TECHNISCHE UNIVERSITÄT DRESDEN
Fakultät Informatik

TECHNISCHE BERICHTE
TECHNICAL REPORTS

ISSN 36629-211X

TUD-FI12-07 Mai 2012

Elke Franz, Stefan Pfennig, André Fischer
Institut für Systemarchitektur

Communication Overhead of Network Coding Schemes Secure against Pollution Attacks

Technische Universität Dresden
Fakultät Informatik
01062 Dresden
Germany

Communication Overhead of Network Coding Schemes Secure against Pollution Attacks

Elke Franz, Stefan Pfennig, and André Fischer

TU Dresden, Faculty of Computer Science
01062 Dresden, Germany
{elke.franz|stefan.pfennig|andre.fischer}@tu-dresden.de

Abstract. Network coding is a promising approach for increasing performance of multicast data transmission and reducing energy costs. Of course, it is essential to consider security aspects to ensure a reliable data transmission. Particularly, pollution attacks may have serious impacts in network coding since a single attacker can jam large parts of the network. Therefore, various approaches have been introduced to secure network coding against this type of attack.
However, introducing security increases costs. Even though there are some performance analysis of secure schemes, to our knowledge there are no details whether these schemes are worthwhile to replace routing under the facet of efficiency. Thus, we discuss in this report parameters to assess the efficiency of secure network coding schemes. Using three network graphs, we evaluate parameters focusing on communication overhead for selected schemes. Our results show that there are still benefits in comparison to routing depending on the network topology.

Keywords: network coding, security, efficiency, performance

1 Introduction

The concept of network coding was introduced by Ahlswede et al. [2]. It allows for increasing throughput for multicast transmissions and for saving bandwidth. Particularly, it has been shown that the min-cut max-flow capacity can be achieved in the multicast scenario [2]. The key idea of network coding is that intermediate nodes compute algebraic combinations from packets they receive, in contrast to common routing where packets are just forwarded by the nodes. For an overview on the topic, we refer to [11–13, 27].

While network coding is a promising approach for increasing efficiency of data transmission, it is vulnerable to various attacks. Thus, introducing security mechanisms is a necessity. Within this report, we focus on the question whether such secure network coding schemes still offer benefits in comparison to traditional routing, and which approaches for secure network coding should be preferred depending on the underlying network topology.

Several approaches have been suggested for network coding. The approaches we evaluate in this report are based on random linear network coding (RLNC),

where the nodes randomly and independently select linear network coding coefficients [16]. RLNC allows for implementing a decentralized solution since there is no need for propagating the coefficients to the nodes.

To counteract the vulnerability to attacks, various schemes for secure network coding have been proposed in the literature (e.g., [5, 9, 17, 18, 23, 26]). Most of these approaches aim at providing security against pollution attacks which may have severe impacts on network coding: Even one polluted packet influences all computations performed by subsequent nodes, hence, may prevent the successful decoding of many other packets at the recipients.

Usually, introducing security implies additional costs. Security mechanisms may require additional computations, introduce delays, or increase storage requirements. This fact raises the question whether secure network coding schemes can still provide benefits regarding throughput and bandwidth as intended by network coding. These questions not only influence the time needed for transmitting data packets through a network. An increased effort finally increases energy consumption of the network, a topic that is today of growing importance.

There are two contributions in this report. First, we discuss which parameters are suited for describing the efficiency of secure network coding schemes. To study the influence of the network topology on these parameters, we use three network graphs that allow for varying network parameters. As second contribution, we present first results of the evaluation of selected secure network coding approaches in comparison to RLNC without security and to routing. These first results focus on communication overhead. The network graphs help in clarifying which characteristics of the underlying network increases additional costs. Such results shall help to assess whether secure network coding can provide benefits for a given network topology at all, and which approach should be preferred.

The report is organized as follows. Section 2 gives an overview on existing approaches for network coding schemes secure against pollution attacks and describes the schemes we selected for our evaluations. Section 3 discusses which parameters are suited for evaluating efficiency and describes the assumptions we made for the evaluations. The results of our evaluation are presented and discussed in Section 4. Finally, Section 5 concludes and gives an outlook.

2 Secure Network Coding

2.1 Random Linear Network Coding

The common notation for describing network coding schemes is based on a directed, acyclic graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ consisting of a set of nodes (also called vertices) V and a set of edges E. There is a number of sending nodes $\mathsf{S} \subset \mathsf{V}$, receiving nodes $\mathsf{R} \subset \mathsf{V}$, and forwarding nodes $\mathsf{F} \subset \mathsf{V}$. A forwarding node receives l data packets $\boldsymbol{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,n})$, $i = 1, 2, ..., l$ on its l incoming edges. Each data packet $\boldsymbol{x}_i$ consists of n codewords $x_{i,j} \in \mathbb{F}_q$. The forwarding node randomly selects l coefficients $\alpha_i \in \mathbb{F}_q$ and computes linear combinations

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

Generally, we assume that it computes different combinations for each outgoing edge. When the receiving nodes got sufficient linear independent packets, they can decode by solving the corresponding equation system.

A practical system for implementing these ideas – Practical Network Coding (PNC) – is introduced in [8]. In our evaluation, we refer to this framework. PNC describes a data format that enables receiving nodes to decode without knowing the randomly selected coefficients. The sender divides the data to be sent into portions $\overline{\boldsymbol{p}}_i \in \mathbb{F}_q^m$ of m codewords each. These native data packets are amended by a global encoding vector $(\beta_{i,1}, \beta_{i,2}, ..., \beta_{i,h}) \in \mathbb{F}_q^h$ that reflects the linear operations. Packets that can be combined during transmission establish a generation $\boldsymbol{G}$. Each Packet is tagged with a unique identifier g_{id}. The size of the generation depends on the multicast capacity h.

Thus, we can think of a generation as a matrix of data packets. The sending node produces a generation containing the original, uncombined data $\boldsymbol{P}$ amended by an $h \times h$ identity matrix $\boldsymbol{B}$ that represents the initial global encoding vector:

$$\boldsymbol{G} = \begin{pmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_h \end{pmatrix} = \begin{pmatrix} \beta_{1,1} = 1 \cdots \beta_{1,h} = 0 & \overline{p}_{1,1}\ \overline{p}_{1,2} \cdots \overline{p}_{1,m} \\ \vdots \quad \ddots \quad \vdots & \vdots \quad \vdots \quad \ddots \quad \vdots \\ \beta_{h,1} = 0 \cdots \beta_{h,h} = 1 & \overline{p}_{h,1}\ \overline{p}_{h,2} \cdots \overline{p}_{h,m} \end{pmatrix} \tag{2}$$

The rows of this matrix are the data packets of size $n = h + m$ codewords sent by the source node. During network coding, the data packets are combined as described by Eq. (1). We refer to combined data packets by

$$\boldsymbol{x}_i = (x_{i,1}, x_{i,2}, ..., x_{i,n}) = (\beta_{i,1}, \beta_{i,2}, ..., \beta_{i,h}, p_{i,1}, p_{i,2}, ..., p_{i,m}). \tag{3}$$

The coefficients of the global encoding vector reflect the linear combinations computed by the forwarding nodes.

Successful decoding requires that the sink nodes receive sufficient linear independent combinations, i.e., the rank of the matrix of received data packets must be h. The probability for successful decoding in case of RLNC depends on the field size q; it becomes sufficiently high for a field size of at least $q = 2^8$ [16].

2.2 Attacker Model

In order to be practically usable, security aspects of network coding need to be considered. Confidentiality, integrity, and availability of the messages (i.e., the original data) have to be ensured even in case of intended attacks. For ensuring confidentiality, the attacker must be prevented from getting to know enough linear independent data packets. For ensuring integrity and availability,

a sufficient amount of data packets needs to be available to the recipient so that he can successfully decode the messages. This implies that integrity and availability of these data packets have to be ensured.

Basically, we have to consider passive as well as active attackers. Passive attackers only observe the system (eavesdropping) while active attackers perform specific actions (modification, deletion, or pollution of packets).

Potential threats to network coding are discussed, e.g., in [10, 23]. If no uncoded packets are sent through the network, an eavesdropper with limited access to the links cannot threaten confidentiality [5]. However, we cannot exclude stronger attacks with certainty. Particularly, if an attacker is able to control a node he can observe and modify all data packets passing this node.

Nevertheless, confidentiality of messages is not widely discussed in the literature since it is mostly addressed at the upper layers of the system [10]. One example for a scheme protecting the confidentiality of messages is SPOC. In that approach, the originally chosen coefficients are encrypted and only the recipient owning the appropriate keys can decrypt the data [25].

The majority of secure network coding schemes, however, considers pollution attacks that must be addressed at the layer of network coding. Therefore, we also focus on this type of attack. Pollution attacks concern the integrity of data packets since they imply that a node processes data packets that do not belong to the subspace spanned by the original data packets. Such attacks are notably critical because polluted packets influence the result of all subsequent combinations computed by forwarding nodes [10]. Finally, the recipients may not be able to successfully decode the data.

2.3 Network Coding Schemes Secure against Pollution Attacks

Various approaches for securing network coding against pollution attacks have been suggested in the literature. An important distinction is the question when polluted packets can be detected and filtered out: only at the recipient nodes or at the forwarding nodes. The former does not imply additional operations performed by the forwarding nodes. However, the latter provides the advantage that forwarding nodes are able to drop polluted packets so that the influence of pollution attacks is limited. Therefore, we selected schemes corresponding to the latter approach for our evaluation.

Network coding schemes that enable forwarding nodes to detect polluted packets are mainly based on cryptography. That means, we need some secret information that can be used for verifying the validity of received packets. However, known cryptographic solutions cannot be directly applied to network coding. Digital signatures are usual for verifying both the integrity and the source of a message, but since the data packets are modified by forwarding nodes, common digital signatures become invalid after the first hop. The same applies to cryptographic hashes and symmetric authentication. Additionally, symmetric authentication would require a key exchange between the sender and all forwarders and recipients.

To overcome these problems, *homomorphic hashes*, *homomorphic signatures*, and *homomorphic MACs* (Message Authentication Codes) have been suggested for securing network coding against pollution attacks. The homomorphic property of these approaches enables forwarding nodes to compute valid hashes or signatures for combined data packets.

Homomorphic hashes were first suggested by Krohn et al. [20] for securing content distribution. In that paper, the hashes were computed for data blocks encoded by rateless erasure codes. The hash function is based on the Discrete Logarithm problem. Gkantsidis and Rodriguez introduced an approach for applying these homomorphic hashes to network coding [15]. Forwarders as well as receivers need the hash values computed by the sender for verifying the validity of the combined data packets. Gennaro et al. present a further approach based on homomorphic hashes [14]; they suggest a hash function that is computed modulo a composite, similar to the RSA crypto system.

Charles et al. introduced homomorphic signatures for network coding [7]. The proposed signatures are based on Weil pairing in elliptic curve cryptography. Kim at al. also suggest a scheme based on digital signatures [19]; the sender computes and publishes signatures that are used by the nodes to verify the authenticity of received data packets. The security of their scheme is based on the Discrete Logarithm problem. Boneh at al. introduced a scheme that enables forwarding nodes to compute valid signatures for combined data packets [4]. Similar to the scheme proposed by Charles at al. [7], the signatures are computed over elliptic curves using bilinear maps like the Weil pairing. Gennaro et al. suggested an RSA-like scheme that promises less communication overhead and increased computational efficiency. Another example for an RSA-like scheme was recently introduced by Catalano et al. [6]. Boneh and Freeman [3] presented a different construction for homomorphic signatures that bases on lattices.

Agrewal and Boneh introduced two solutions for homomorphic MACs [1]. Their homomorphic MAC scheme enables the recipients to verify the validity of data packets while their broadcast homomorphic MAC scheme additionally enables all forwarding nodes to verify data packets. Yu et al. describe a schemes that allows for efficient verification of MACs [28]; however, their schemes implies significantly enlarged bandwidth overhead since all MACs received by forwarding nodes are appended to the data packets. Generally, symmetric authentication requires the aforementioned prior exchange of symmetric keys.

Another approach for securing network coding against pollution attacks is the delayed delivery of information necessary for verifying the validity of received data packets. This *time asymmetry* was introduced in the TESLA protocol [24], a broadcast authentication protocol with delayed key release.

An approach that utilizes time asymmetry was introduced by Dong et al. [9]. In that scheme, the sender periodically computes checksums for the data packets sent before; the forwarders and recipients can verify the validity of the data packets by means of these check sums.

There are also schemes that combine time asymmetry and homomorphic MACs. These schemes utilize the time asymmetry for exchanging symmetric

keys. One example is the RIPPLE, introduced by Li et al. [22]. The name stems from the fact that data packets pause at intermediate nodes for key disclosure and verification. The TESLA-like scheme introduced by Le et al. [21] aims at both identification of polluted packets and identification of the attacker in network coding based-P2P systems. However, it still requires key exchange and additionally the existence of a central authority for identification of attackers. Wang proposed a further TESLA-like scheme [26]. In that scheme, the number of MACs depends on the maximum number of hops from sender to recipient.

Generally, schemes based on asymmetric cryptography require more computational effort while TESLA-like schemes increase delay and communication overhead. However, the actual costs depend on the underlying network graph and the communication requirements. Hence, the dependence of the additional costs on parameters describing the network should be known to decide which approach should be preferred in a concrete communication scenario.

Within this report, we provide first results for the selected secure network coding schemes described in the following section. To compare the performance of the different approaches we selected one example for each of them (homomorphic hashes, homomorphic MACs, time asymmetry, and time asymmetry combined with homomorphic MACs).

2.4 Schemes selected for Evaluation

In order to achieve a similar presentation, we slightly adapt the description of the schemes given in the cited articles to the description introduced in Sect. 2.1.

Homomorphic Hashes [15]. The first scheme we selected uses a hash function h to enable recognizing polluted packets. The hash function includes exponentiation modulo a prime r of size 1024 bits, hence, the size of the hash values is 1024 bits. The encoding operations are computed in $\mathbb{Z}_q$, whereas q is a prime of 256 bit and $q|(r-1)$. Thus, the size of the code words is about 256 bits.

Given a vector $\boldsymbol{g}$ of m random elements of $\{u \in \mathbb{Z}_q \mid <u>= \mathbb{Z}_q\}$, a hash value for a data packet $\boldsymbol{p}_i$ can be computed as follows :

$$\mathrm{h}(\boldsymbol{p}_i) = \prod_{j=1}^{m} g_j^{p_{i,j}} \bmod r.$$

The sender computes for each native data packet $\overline{\boldsymbol{p}}_i$ a hash value $\mathrm{h}(\overline{\boldsymbol{p}}_i)$. Since the hash values are homomorphic, forwarding nodes can verify the validity of data packets $\boldsymbol{x}_i = (\boldsymbol{\beta_i}, \boldsymbol{p}_i)$ by comparing the hash of these data packets to the linear combination of the hashes delivered by the sender:

$$\mathrm{h}(\boldsymbol{p}_i) \stackrel{?}{=} \mathrm{h}(\overline{\boldsymbol{p}}_1)^{\beta_{i,1}} \cdot \mathrm{h}(\overline{\boldsymbol{p}}_2)^{\beta_{i,2}} \cdot \ldots \cdot \mathrm{h}(\overline{\boldsymbol{p}}_h)^{\beta_{i,h}} \bmod r.$$

Hence, the hashes $\mathrm{h}(\overline{\boldsymbol{p}}_i)$ must be known to the forwarding nodes. Since the context of the paper is content distribution, the authors assume that the nodes download the hash values when they join the system. For our analysis, we assume that the sender broadcasts the hash values before transmission of the data

packets. To ensure authenticity of the hashes, they need to be digitally signed. The structure of the data packets does not need to be changed.

DART [9]. As second scheme, we selected the TESLA-like scheme DART that is based on delayed checksum delivery. The sender periodically computes and disseminates a signed checksum packet consisting of the checksum $\mathrm{chk}_s(\boldsymbol{G})$, a seed s, and a timestamp t for the current generation $\boldsymbol{G} = \boldsymbol{B}|\boldsymbol{P}$. Given a predetermined security parameter c, the sender generates a pseudo-random $h \times c$ matrix $\boldsymbol{H}_s$ using the seed s and a publicly known function f. That matrix is used for computing the checksum:

$$\mathrm{chk}_s(\boldsymbol{G}) = \boldsymbol{P}\boldsymbol{H}_s.$$

Each node maintains two buffers: verified_set and unverified_set. After receiving a checksum packet, the node first checks its authenticity. If the verification succeeded, the node re-broadcasts the checksum packet to its neighbors and then checks packets in unverified_set it has received before the checksum was generated. To verify the data packets, the node also generates $\boldsymbol{H}_s$ using the seed s that is contained in the checksum packet. Then the node checks if the product of global encoding vector and checksum equals the product of encoded data and random matrix:

$$\boldsymbol{\beta}_i \mathrm{chk}_s(\boldsymbol{G}) \stackrel{?}{=} \boldsymbol{p}_i \boldsymbol{H}_s.$$

Invalid packets are discarded; successfully checked packets are transferred to verified_set and will be used for computing linear combinations. Since each node needs for verification a new checksum packet, the number of checksum packets the sender has to generate depends on the number of hops to the recipients.

For increasing efficiency, the authors suggest batch verification. Furthermore, to reduce the introduced delays, pipelining is suggested in which several generations are sent and processed concurrently. A variant of the basic scheme is EDART, also introduced in [9]. In EDART, nodes do not always verify packets but optimistically forward packets to increase throughput. For our evaluations, we consider the basic scheme.

Scheme according to Wang [26]. The third scheme we selected for evaluating additional costs utilizes symmetric authentication – homomorphic MACs (Message Authentication Codes) – and time asymmetry for delayed key release. The number of MACs per data packet depends on the number of hops. The MACs are integrated into the data packets.

Verification of the MACs requires knowledge of the corresponding keys. For each generation with identifier g_{id}, the sender generates a chain of seed values b^i that are the basis for computing these keys. It first selects a seed s and then computes $b^u = \mathrm{h_1}^u(s, g_{id})$ with $u = 1, 2, \ldots, k+1$ (k: maximum number of hops to the recipients). By means of these values, the sender generates the actual keys $a_j^u = \mathrm{h}_2(j, b^u)$ ($j = 1, 2, \ldots,$ at maximum $n + k$ values) with $\boldsymbol{a}^u =$

$(a_1^u, a_2^u, \ldots, a_n^u)$. Here, h_1 and h_2 are pseudo-random functions with ${\mathrm{h}_1}^u(\cdot) = \mathrm{h}_1(\mathrm{h}_1(\cdots \mathrm{h}_1(\cdot)))$.

For each data packet $\boldsymbol{x}_i$ of a generation, the sending node computes k MACs as follows:

$$\mathrm{MAC}_1(\boldsymbol{x}_i) = \boldsymbol{a}^1 \cdot \boldsymbol{x}_i,$$
$$\mathrm{MAC}_u(\boldsymbol{x}_i) = \boldsymbol{a}^u \cdot \boldsymbol{x}_i + \sum_{j=1}^{u-1} a_{n+j}^u \mathrm{MAC}_j(\boldsymbol{x}_i).$$

The final value b^{k+1} of the chain is necessary for checking the validity of subsequently sent values b^u; therefore, the sender first digitally signs this value and broadcasts it to the involved nodes at time $t = 0$. In the next time slot, the sender sends b^k. When the forwarding nodes get this value, they can check its validity by means of the publicly known function h_1. According to the TESLA scheme, the seed values are distributed to the verifying nodes after the data packets arrived. To prevent that an attacker can compute upcoming seed values, the code word length is increased to 128 bits.

Generally, a node r hops away from the sender waits for b^{t-r+1} and verifies its validity by checking whether ${\mathrm{h}_1}^r(b^{k-r+1}) = b^{k+1}$. If the test was successful, the node verifies MAC_{k-r+1} of the received messages by computing these MACs on its own and comparing the computed MACs to the MACs contained in the received data packets. Since the node knows b^{k-r+1}, it can compute a_j^{k-1+r} (with $j = 1, 2, \ldots, n + k - r$) needed for computing the MACs.

Due to the homomorphic property of the MACs, nodes can compute valid MACs for the combined packets by combining the received MACs using the randomly selected coefficients α_i. Each node has to compute one MAC less than received. Consequently, if the path of a data packet from sender to recipient contains the maximum number of k hops, the data packet includes only one MAC when it arrives at the recipient.

RSA-based scheme [14] The last scheme we analyzed uses the homomorphic property of the simple RSA signature scheme. The sender generates a key pair using the modulus N which is the product of two safe primes. This pair consists of a public key (N,e,$g_1 \ldots g_m$) and a private key d , where $ed = 1 \mod \phi(N)$ and each $<g_i> = \mathcal{QR}_N$. Then the sender generates a signature for each data packet $\boldsymbol{x}_i$ using his private signature key d and a publicly known hash function h and integrates it into the particular packet. Setting $f_i = \mathrm{h}(i, g_{id})$, the signature is computed by:

$$\mathrm{Nsig}(\boldsymbol{x_i}) = \left(\prod_{j=1}^{h} f_j^{\beta_{i,j}} \prod_{j=1}^{m} g_j^{p_{i,j}} \right)^d \mod N.$$

Each node is able to verify signatures with the use of the public test key e by checking:

$$\mathrm{Nsig}(\boldsymbol{x}_i)^e \stackrel{?}{=} \prod_{j=1}^{h} f_j^{\beta_{i,j}} \prod_{j=1}^{m} g_j^{p_{i,j}} \bmod N.$$

Because of the homomorphic property, forwarders can compute valid signatures for a combination of ℓ verified packets by multiplying the signatures raised to the power of the local coefficients α_i:

$$\mathrm{Nsig}(\boldsymbol{x}_j) = \prod_{i=1}^{\ell} \mathrm{Nsig}(\boldsymbol{x}_i)^{\alpha_i} \bmod N.$$

All operations concerning the signature are done modulo the composite number N. Hence the size of the signatures is constant, e.g., 1024 bit.

In contrast to the schemes described above, the codewords are arbitrary integers instead of elements of a finite field. Thus, each $x_{i,j}$ will grow by a certain amount of bits depending on the number of hops k to the recipient and the number of ingoing edges ℓ of each forwarding node involved in the transmission, e.g. 10 bits per hop with $\ell = 4$. For optimization of communication overhead, we set $m = 1$, i.e., we have only one large $p_{i,1}$ per packet $\boldsymbol{x}_i$. However, this setting implies higher computational costs. A larger m increases communication overhead while decreasing computational costs.

3 Assumptions for the Analysis

3.1 Parameters for the Evaluation of the Schemes

The major concern of our comparison is to answer the question whether secure network coding schemes can still provide benefits in comparison to traditional routing. On one hand, we can compare the performance of the secure schemes to the performance of schemes without security mechanisms. On the other hand, it is reasonable to evaluate additional costs implied by introducing security. It is necessary to define suitable parameters reflecting performance and additional costs. Thereby, we are interested in results that describe the dependence of the selected parameters on the characteristics of the underlying network so that it is possible to assess

1. whether secure network coding can offer benefits in comparison to routing for a given data flow, and if this is the case,
2. which approach for secure network coding should be preferred for the given network.

Consequently, we focus on parameters that can be analyzed on a rather abstract level without knowing technical details of the system components.

Performance is usually described by parameters like throughput or delay. To allow for a theoretical analysis, we evaluate the **number of ticks** (time slices) necessary to deliver the messages to the intended recipients, a parameter

that should highly correlate with the real delay and, hence, roughly reflects the throughput within the network. To simplify matters, we assume that each node can receive on all incoming edges and send on all outgoing edges at the same time. Furthermore, we assume that both transmission of a packet via one link and processing of packets by the nodes need one tick. Thus, the results describe the minimum introduced delay.

Various parameters can be evaluated to describe additional costs and, finally, additional energy required for secure data transmission. Describing absolute energy consumption requires detailed knowledge about the system, e.g., about the implementation of various operations or about the energy requirements of the system components. Generally, additional costs can be described by

- additional operations to be performed by the nodes,
- memory overhead, and
- communication overhead.

Additional operations clearly increase energy consumption. Memory overhead also implies additional operations for accessing the memory. We did not consider these issues since they strongly depend on technical conditions and we focus on a theoretical analysis that allows answering the general questions given above. Additional operations and memory accesses might influence the time needed by the nodes for processing data. Due to simplicity, we assume that nodes have enough computing power and memory so that neither additional operations nor memory accesses introduce additional delays. Within our evaluation, we focused on *communication overhead* introduced by security mechanisms. Generally, we assumed a predetermined size per packet for all schemes under investigation. IP addresses and other header information are not considered since they are equal for all schemes. We evaluated three parameters describing communication overhead referring to the transmission of a single generation $\boldsymbol{G}$.

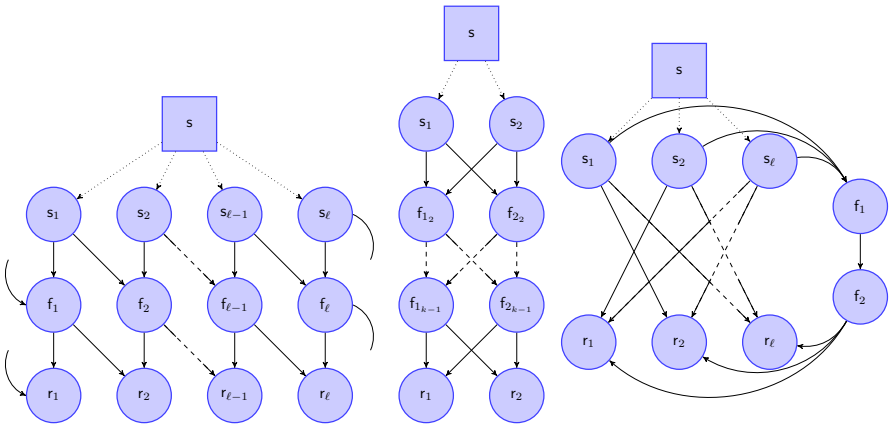
As first parameter, we evaluated the **maximum relative payload**. This parameter helps to assess the amount of additional data introduced by secure network coding schemes to allow authentication of data packets – checksums, digital signatures, MACs, etc. This additional data needs to be included within the data packets or has to be sent in extra data packets what reduces the available payload. Thus, the maximum relative payload allows for coarsely estimating how many generations are necessary to transmit a given amount of data. Considering in addition the number of ticks necessary for transmitting one generation allows roughly assessing the delay for transmitting that amount of data. We want to point out that the processing of multiple generations needs to be considered for such an estimation, e.g., applying pipelining as suggested in [9].

However, the actual amount of data packets to be sent by the sending node may be larger since it depends on the underlying network graph. For example, all nodes involved in data transmission need the authentication information for checking the validity of received data packets. Thus, data packets containing authentication information may be sent several times. Hence, we evaluated as second parameter the **actual relative payload**, a value that reflects the actual network load initiated by the sending node.

As a third parameter, we determined the **send operations** necessary for transmitting all the data packets to the recipients. This parameter gives an impression of the overall effort in the whole network while the parameters referring to the relative payload just considered the applied load for the sending node. In case of constant packet size the amount of data sent in the whole network for the transaction is linearly dependent on the number of send operations. Thus, this parameter should roughly correspond to the overall network load.

3.2 Network Topology

The parameters introduced above describe the efficiency of the selected schemes depending on the underlying network topology. Thus, it seems to be reasonable to use network topologies for the evaluation that are suitable to study the influence of relevant network properties. Particularly, we considered the number of nodes involved in a data transmission and the number of hops to the recipients. For our evaluation we used the network models depicted in Fig. 1.



(a) Model 1 (b) Model 2 (c) Model 3

Fig. 1: Considered network topologies.

According to the definition in Sect. 2.1, sending nodes s_i are nodes that only send data but do not compute linear combinations. Generally, we assume that a large file should be transmitted, so we introduce a virtual source node s that has the task to distribute data to the sending nodes s_i. For the analysis we restrict our focus on one generation $\boldsymbol{G}$ whose size depends on the broadcast capacity h. All edges are assumed to be equal and to have unit capacity, so h is determined by the min-cut of G. Furthermore, we assume that all receiving nodes shall get all messages contained in one generation. Nodes that have the same distance from the sending nodes are considered to be on the same level.

Model 1 is intended to study the influence of the number of nodes involved in transmission. The size of a generation is 2 for this example, thus, the virtual source node distributes 2 packets alternating to the sending nodes s_i for $1 \leq i \leq \ell$ (ℓ even). Each node has 2 direct connections to nodes on the next level. Thus, each forwarding node f_i for $1 \leq i \leq \ell$ should get 2 different packets.

Model 2 allows for evaluating the impact of the number of hops k to the recipients. The virtual source node distributes 2 different packets to the sending nodes s_1 and s_2. Every node can communicate to every node on the next level. The number of forwarding nodes increases with a growing number of hops k.

Model 3 was originally introduced by Fragouli et al. [12] to demonstrate possible benefits of network coding. We analyzed how secure network schemes perform with that graph. In contrast to the scheme introduced in [12], we again assume that all recipients should get all messages to have comparable conditions. The virtual source node sends a total of ℓ packets $\boldsymbol{x}_i$ to the sending nodes s_i for $1 \leq i \leq \ell$. Now every receiver r_i is interested in getting all original messages $\boldsymbol{p}_i$ for $1 \leq i \leq \ell$. There is no direct communication possible between s_i and r_j for $i = j$. The only obvious way is to communicate via the two forwarder nodes f_1 and f_2, the link between these nodes establishes a kind of bypass.

Furthermore, we have to analyze the probability that the receiver is able to decode all packets. This probability depends on the underlying finite field $\mathbb{F}_q$ and in some cases on the topology of the network. In general the probability is about $(q-1)/q$, which is also true for model 1 and 2. Indeed, it is essential for model 2 that every node sends different packets on its outgoing edges, otherwise the decoding probability decreases with rising k. In model 3, only node f_1 performs network coding. Hence, it can be assured that all recipients can successfully decode all packets disregarding transmission errors.

3.3 Packet Format

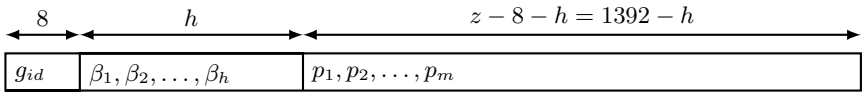
As far as possible, we used similar conditions for evaluating the schemes considering the different network graphs. We always assumed a packet size of $z = 1400$ byte. Furthermore, we determined the selected parameters for transmission of one generation disregarding the actual payload size. Generations are not relevant for routing, here we assumed that h data packets should be transmitted to each receiver.

Figure 2 provides an overview on the packet formats of all selected schemes. The size of the data fields used for the evaluation is given in bytes.

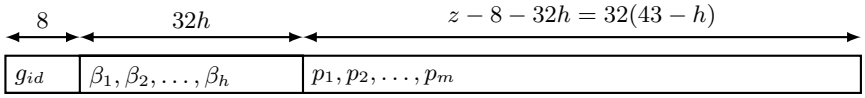
Since we use PNC as basis for our evaluations, the data packets of all schemes generally contain a generation number g_{id}, an encoding vector $\boldsymbol{\beta}_i$, and the vector $\boldsymbol{p}_i$. The number of coefficients β_i contained in the encoding vector depends on the size h of the generation and is thus conditioned by the network topology (Sect. 2.1). Additional data fields are given by the particular scheme; the size of data fields that need to be included in the data packets determines the size of $\boldsymbol{p}_i$ and, hence, the possible payload.

Chou et al. suggest using one or two bytes for the generation identifier g_{id} [8]. We want to point out that they assume a generation size of $h = 50$. However, the

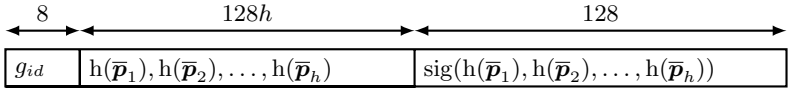
PNC ($q = 2^8$)
data packets:



Homomorphic Hashes ($q \approx 2^{256}$)
data packets:

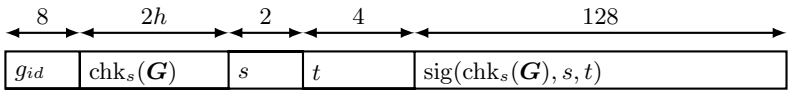


additionally required: hash-packet broadcasted before transmission

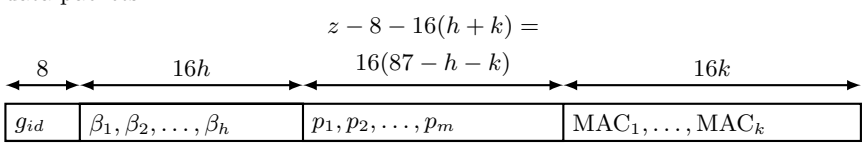


DART ($q = 2^8$)
data packets: identical to PNC

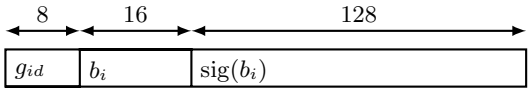
additionally required: checksum-packets, which are periodically broadcasted



WANG ($q = 2^{128}$)
data packets:



additionally required: $k + 1$ seed value packets (signature only for b_{k+1} necessary)



RSA-based scheme (all codewords β_i and p_1 are integers and increases in size)

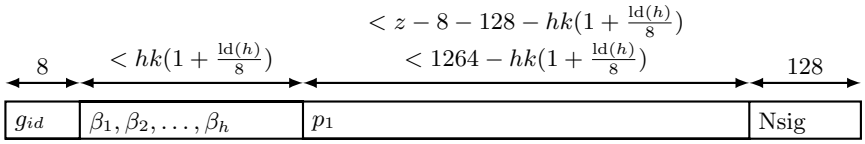


Fig. 2: Packet formats and size (in bytes) assuming a packet size of $z = 1400$ byte.

network models evaluated in our report have a considerably smaller generation size what implies that more generations have to be distinguished in the network. We set the size of g_{id} to 8 bytes to get a real unique identifier. Moreover, unique identifiers are important for security, e.g., regarding replay attacks. Security was not explicitly considered in [8]. Of course, security need to be discussed in more detail, but this is out of scope of this report.

DART includes the security parameter c that controls the size of the random matrix $\boldsymbol{H}_s$. According to [9], we used $c = 2$ in our evaluations. The seed s and the timestamp t should have a size of 2 respectively 4 bytes. Furthermore, we assume the size of a digital signature to be 128 bytes for all schemes. Higher security requirements will imply longer signatures.

Due to the increasing size of the integers in the RSA-based scheme, we cannot precisely assess the size of the codewords for that scheme. In the evaluation, we worked with an upper bound since the maximum length need to be considered by the sender. We like to point out that we used the min-cut h as estimation for the incoming edges l to get an upper bound independent from the detailed network graph. This substitution will work for all presented network graphs but not necessarily for all existing graphs.

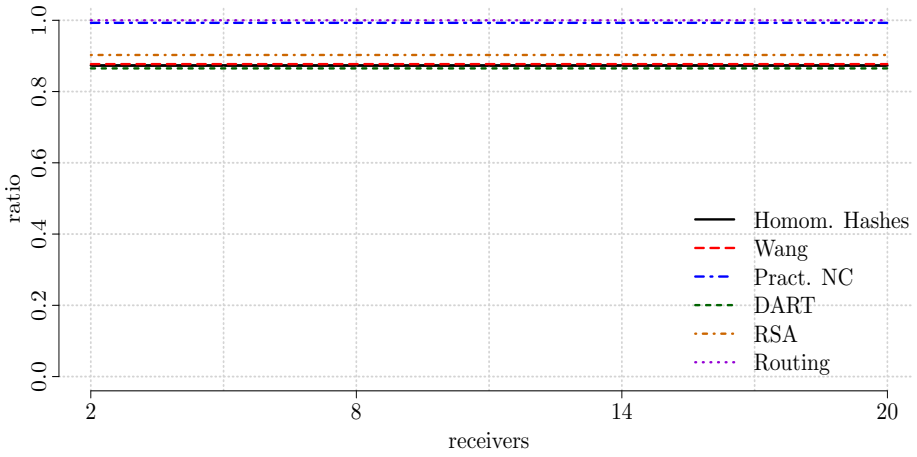
4 Evaluation and Results

4.1 Results of our Theoretical Analysis

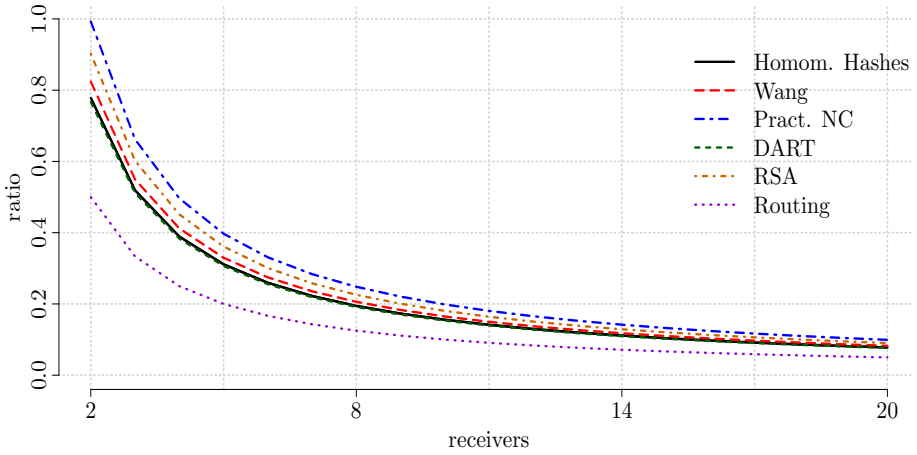
The diagrams show results for the selected secure network coding schemes compared to network coding without security (PNC) and to routing. For the latter, we assume that each packet has only one network destination. Hence, a packet has to be sent several times if there are multiple recipients. The results for the network models shown in Fig. 1 are discussed in the following.

Model 1. The constant generation size of model 1 implies a constant size of the encoding vector. Thus, the maximum relative payload is also constant for all schemes (Fig. 3a). The maximum relative payload of the Wang scheme further depends on the number of hops since this value determines the number of MACs to be attached, however, the number of hops is also constant for model 1. Since routing does not require to include additional data, it achieves a maximum relative payload of 1.0. PNC implies the introduction of the global encoding vector and therewith a loss of only h codewords per packet. Hence, it also achieves a high maximum relative payload of 0.99. Due to the largest field size, Homomorphic Hashes achieve the worst maximum relative payload.

The actual relative payload depends on the number of receiving nodes since more data need to be sent (Fig. 3b). This parameter reflects the advantages of network coding in comparison to routing. Even if the additional data required by the secure network coding schemes decrease the actual relative payload, the schemes are still better than routing.



(a) Maximum relative payload

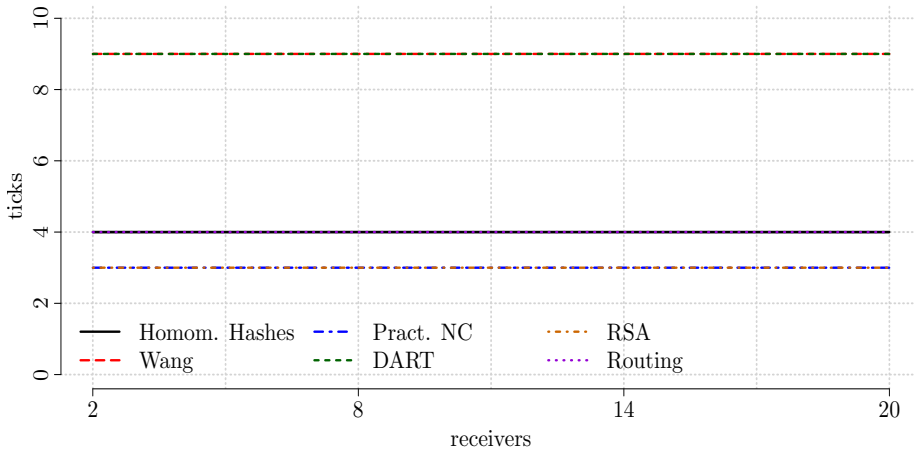


(b) Actual relative payload

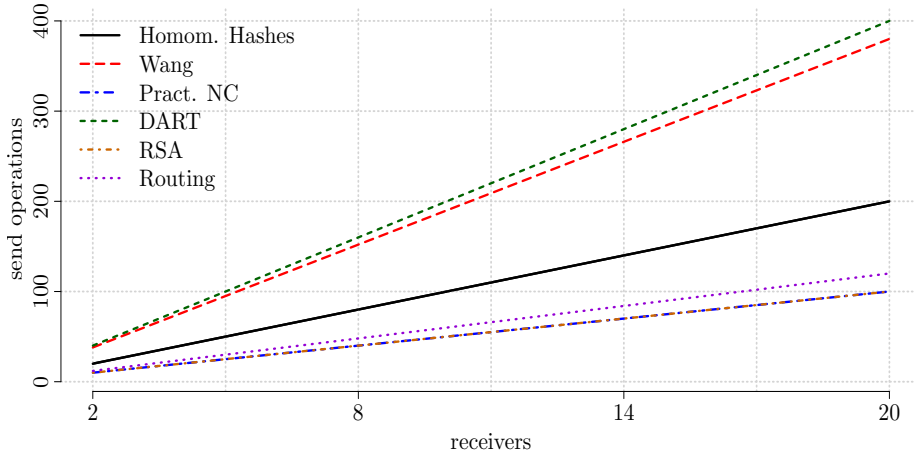
Fig. 3: Results for model 1: Transmitting two packets.

The number of ticks until all data packets are transmitted to the recipients is also constant for all schemes (Fig. 4a). PNC shows the benefit of network coding, but the RSA-based scheme achieves the same good results here. Schemes that utilize time asymmetry need of course more ticks for the transmission.

The time asymmetry also increases the number of send operations in the whole network since the data necessary for verifying the data packets need to be sent to the nodes involved in transmission (Fig. 4b). Schemes without time asymmetry are much better, in the best case, they require less network load than routing.



(a) Number of ticks



(b) Send operations

Fig. 4: Results for model 1: Transmitting two packets.

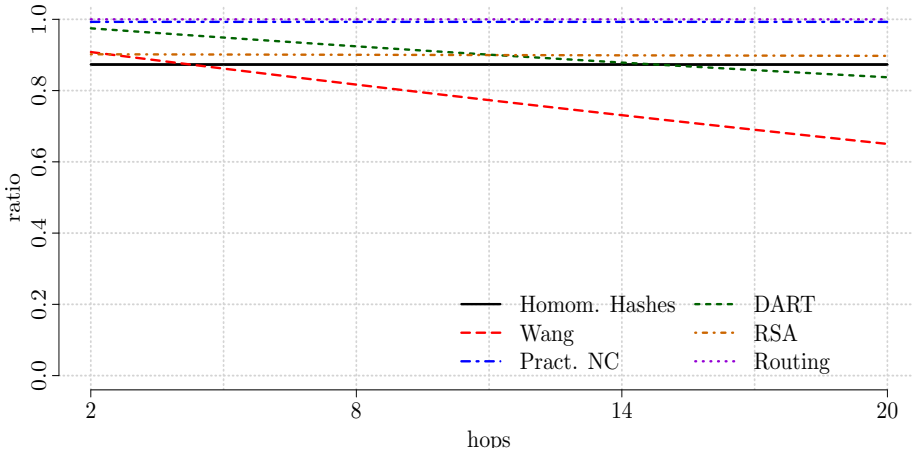
The diagrams only consider the transmission of a single generation. Given the maximum payload of one generation, it is possible to compute the number of generations needed for transmitting a given amount of data. For example, the transmission of a file of 1 GB requires sending 357 143 "generations" (i.e., $h = 2$ data packets) for routing, 359 713 (+0.7%) for PNC and DART each, 363 373 (+1.7%) for Homomorphic Hashes, 381 098 (+6.7%) for Wang, and 396 511 (+11%) for RSA.

Model 2. The generation size for this model is also constant. Thus, the size of the encoding vector is constant. However, the number of hops increases which

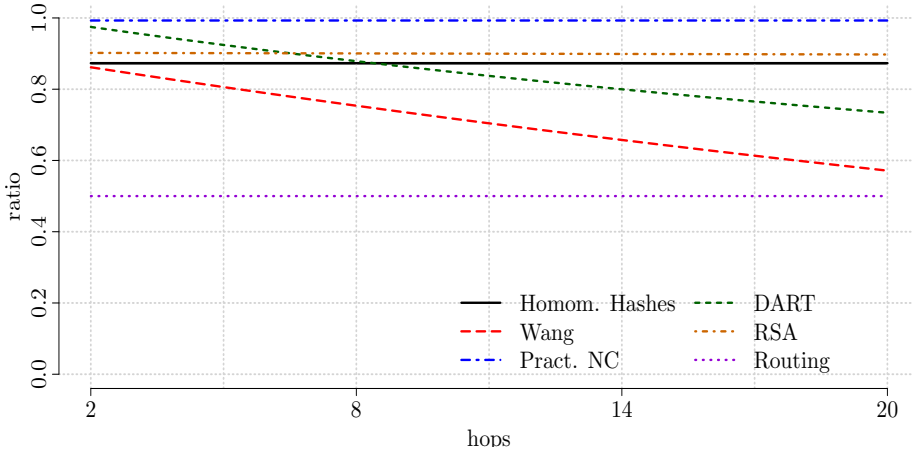
also implies that the number of forwarding nodes increases. Hence, the relative payload of schemes that require sending authentication information decreases with an increasing number of hops (Fig. 5a and 5b). The influence is especially strong for the Wang scheme since the number of MACs depends on the number of hops.

The number of ticks equally increases for all schemes without time asymmetry (Fig. 6a). Again, the influence is significant for the Wang scheme.

The results for the last parameter are similar except that DART yields the worst results due to the need to broadcast the checksums (Fig. 6b).

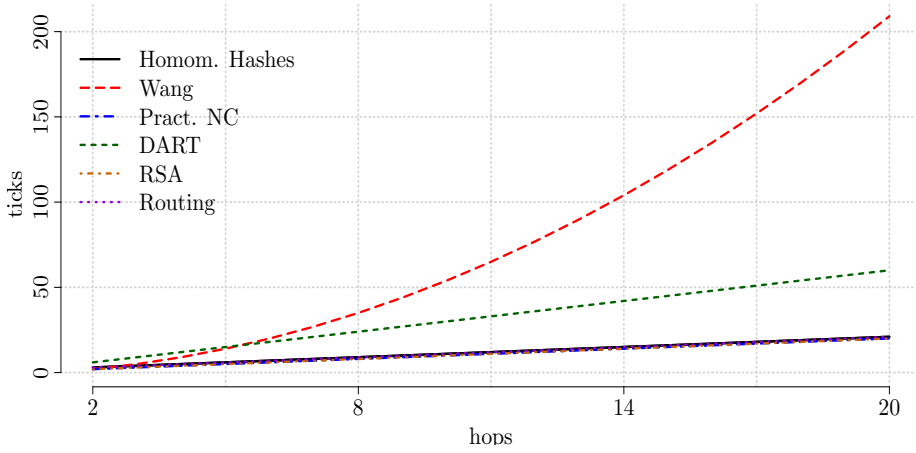


(a) Maximum relative payload

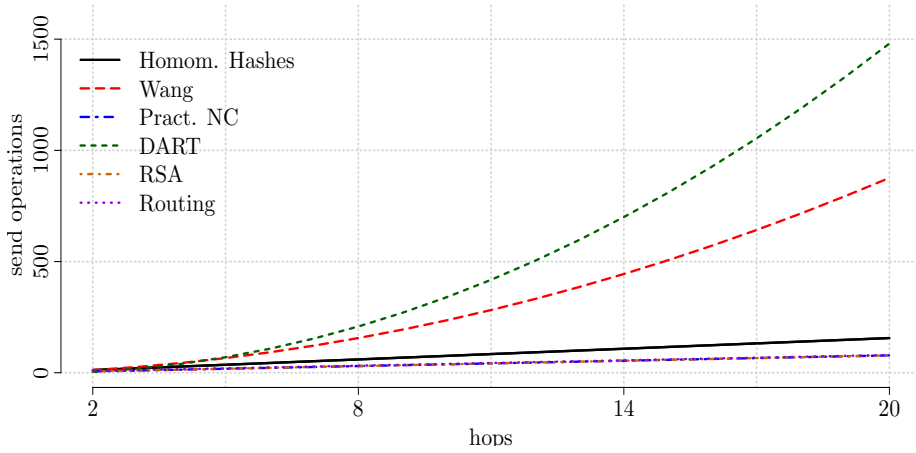


(b) Actual relative payload

Fig. 5: Results for model 2: Transmitting two packets.



(a) Number of ticks



(b) Send operation

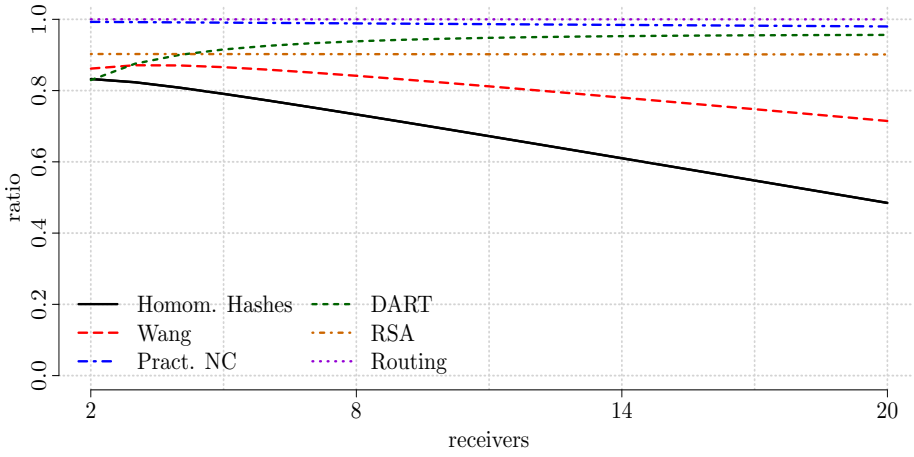
Fig. 6: Results for model 2: Transmitting two packets.

Model 3. In contrast to the other models, the generation size increases for this model. Hence, the maximum relative payload of all network coding schemes at least slightly decreases (Fig. 7a). In case of a larger field size, the influence is stronger. Contrary to expectations, the maximum relative payload increases for DART for a small number of nodes. The reason is that the digital signature has a stronger influence on the relative payload if there are only few packets in a generation.

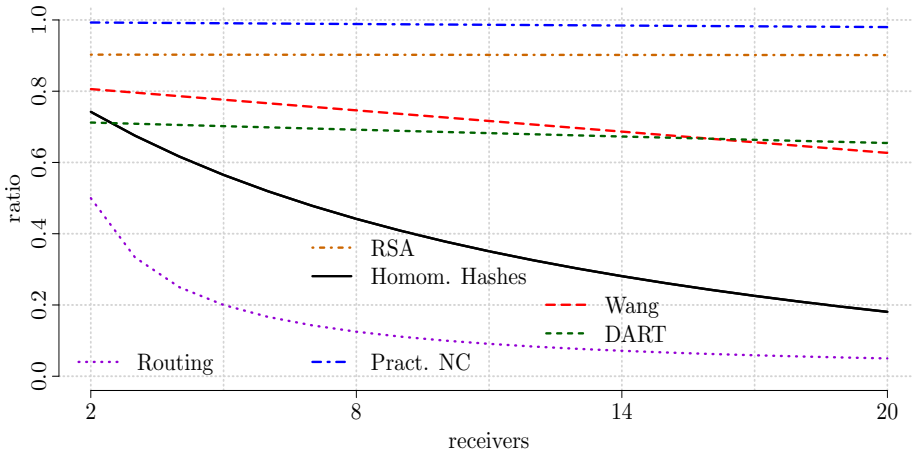
Model 3 was introduced in the literature to illustrate the potential benefits of network coding. This is especially reflected by the actual relative payload (Fig. 7b). Even the secure network coding scheme that yields the worst parameter

outperforms routing regarding this parameter. If there is a bottleneck in the network, we can expect that network coding provides advantages.

Due to the bottleneck in this network graph, even the network coding schemes based on time asymmetry outperform routing regarding the number of ticks if there are more than 10 recipients (Fig. 8a). For Homomorphic Hashes, the number of ticks jumps after a certain increase of the number of recipients. The reason is the necessity to send the hashes of the original data at the beginning. Due to the size of the hashes, there can be at maximum 9 hashes plus signature in a data packet. If the number of packets per generation exceeds this number, an additional data packet needs to be sent.

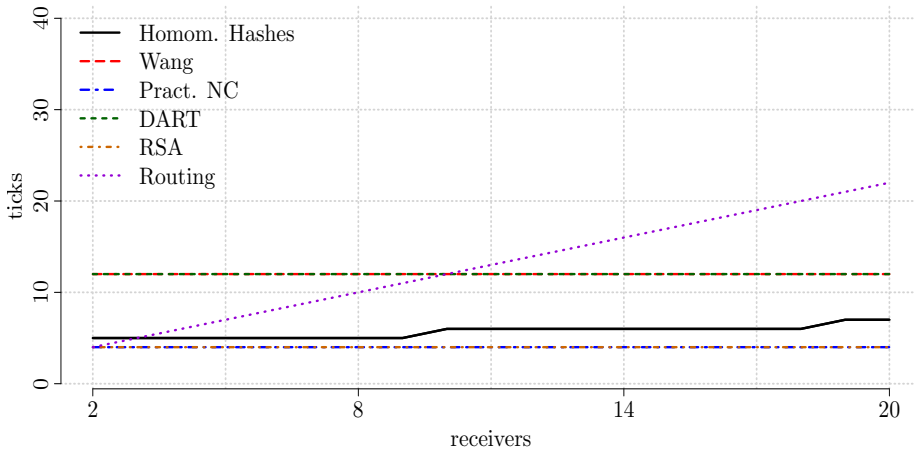


(a) Maximum relative payload

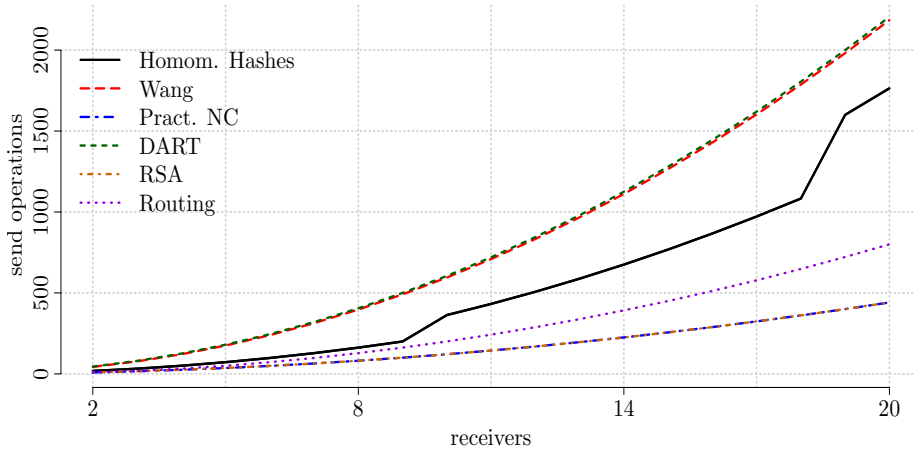


(b) Actual relative payload

Fig. 7: Results for model 3: Transmitting ℓ packets.



(a) Number of ticks



(b) Send operation

Fig. 8: Results for model 3: Transmitting ℓ packets.

The same reason causes jumps regarding the number of send operations for Homomorphic Hashes (Fig. 8b). The RSA-based scheme does not imply additional send operations in comparison to PNC, thus, it outperforms routing for this model.

4.2 Discussion of the Results

Generally, we can summarize two basic results that confirm our assumptions: First, network coding (PNC) outperforms routing in terms of throughput (number of ticks), network load (number of send operations), and actual relative

payload. Second, introducing security increases costs – secure network coding schemes yield at best the same results as PNC, but no better results. However, a closer look at the results reveals that secure network coding schemes may be still better than routing. As example, consider model 3 with 12 recipients. Best results for the evaluated secure network coding schemes regarding routing are: actual relative payload 391 % - 1082 %, number of ticks 28.6 % - 85.7 %, and send operations 58.7 % (RSA-based scheme).

For the maximum relative payload, routing always delivers the best results due to the fact that network coding schemes always require to contain some additional data. However, this parameter is rather theoretical since it does not take into account the underlying network topology. Thus, we focus on the other parameters in this concluding discussion.

Schemes that require including additional data in the data packets decrease the actual relative ratio. This influence is especially strong if the alphabet size needs to be increased, e.g., in the scheme according to Wang [26]. Regarding the number of ticks that represents the delay for transmitting messages and the number of send operations that influences the energy consumption, schemes with no time asymmetry are clearly better. For schemes with time asymmetry, the number of nodes involved in data transmission as well as the number of hops have a significant influence on these parameters.

So far, we got best results for the RSA-based scheme [14] and we expect that we would get similar results regarding the evaluated parameters for other schemes that do not utilize time asymmetry. However, we want to point out that we worked with a setting that reduces the communication overhead (Sect. 2.4).

5 Summary and Outlook

Our results show that secure network coding can still provide benefits regarding communication overhead in comparison to routing. However, we want to point out that the results presented in this report are not sufficient to completely assess the efficiency of secure network coding schemes. Particularly, we solely focused on parameters describing communication overhead. A comprehensive comparison of secure network coding schemes regarding their efficiency calls for considering *all* efficiency parameters sketched in Sect. 3.

Moreover, answering the question whether secure network coding is beneficial at all and which approach should be preferred requires to analyze the given network and communication requirements. For example, it is necessary to determine what requires most energy considering the technical conditions of the network given – more computations, more sending operations, or whatsoever. Enhancing the evaluations by considering more parameters as well as dependencies on technical conditions are topics of future work.

Next steps will also include simulation runs. We are currently working on a network coding simulator based on the NS3 framework[1]. This simulator will allow to consider various communication scenarios and multiple data flows.

[1] http://www.nsnam.org/

Acknowledgement. This work is supported by the German Research Foundation (DFG) in the Collaborative Research Center 912 "Highly Adaptive Energy-Efficient Computing". We wish to thank Sebastian Clauß, Sabrina Gerbracht, Eduard Jorswieck, Christian Scheunert, Dagmar Schönfeld for their constructive comments.

References

1. S. Agrawal and D. Boneh. Homomorphic MACs: MAC-based integrity for network coding. In *Applied Cryptographyand Network Security*, pages 292 – 305, 2009.
2. R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Trans. on Information Theory*, 46(4):1204–1216, 2000.
3. D. Boneh and D. M. Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In *Proc. of Public Key Cryptography (PKS 2011)*, volume LNCS 6571, pages 1 – 16. Springer, 2011.
4. D. Boneh, D. M. Freeman, J. Katz, and B. Waters. Signing a linear subspace: Signature schemes for network coding. In *Public Key Cryptography*, pages 68–87, 2009.
5. N. Cai and R. W. Yeung. Secure Network Coding. In *Proc. IEEE Int. Symp. on Information Theory*, 2002.
6. D. Catalano, D. Fiore, and B. Warinschi. Efficient network coding signatures in the standard model. In *Proc. of PKC 2012*, volume 7293 of *LNCS*, pages 680 – 696, 2012.
7. D. Charles, K. Jain, and K. Lauter. Signatures for Network Coding. In *Proc. Conf. on Information Sciences and Systems*, 2006.
8. P. A. Chou, Y. Wu, and K. Jain. Practical network coding. In *Proc. Annual Allerton Conference on Communication, Control, and Computing*, 2003.
9. J. Dong, R. Curtmola, and C. Nita-Rotaru. Practical defenses against pollution attacks in intra-flow network coding for wireless mesh networks. In *Proc. WiSec*, 2009.
10. J. Dong, R. Curtmola, and C. Nita-Rotaru. Secure network coding for wireless mesh networks: Threats, challenges, and directions. *Computer Communications*, 32:1790–1801, 2009.
11. C. Fragouli, J.-Y. L. Boudec, and J. Widmer. Network coding: An instant primer. *SIGCOMM Computer Communication Review*, 36:63–68, 2006.
12. C. Fragouli and E. Soljanin. *Network Coding Applications*. Now publishers, 2007.
13. C. Fragouli and E. Soljanin. *Network Coding Fundamentals*. Now publishers, 2007.
14. R. Gennaro, J. Katz, H. Krawczyk, and T. Rabin. Secure network coding over the integers. In *Proc. PKC 2010*, pages 142–160, 2010.
15. C. Gkantsidis and P. R. Rodriguez. Cooperative Security for Network Coding File Distribution. In *Proc. IEEE Int. Conf. on Computer Communications*, 2006.
16. T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros. The benefits of coding over routing in a randomized setting. In *Proc. of the IEEE International Symposium on Information theory*, 2003.
17. 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 Information Theory*, 54(6):2798–2803, 2008.
18. S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Médard. Resilient network coding in the presence of byzantine adversaries. In *in Proc. 26th Annual IEEE Conf. on Computer Commun., INFOCOM*, pages 616–624, 2007.

19. M. Kim, L. Lima, F. Zhao, J. Barros, M. Mdard, R. Koetter, T. Kalker, and K. J. Han. On counteracting byzantine attacks in network coded peer-to-peer networks. *IEEE Journal on Selected Areas in Communications*, 28(5):692–702, 2010.
20. M. Krohn, M. Freedman, and D. Mazières. On-the-Fly Verification of Rateless Erasure Codes for Efficient Content Distribution. In *Proc. IEEE Symp. on Security and Privacy*, 2004.
21. A. Le and A. Markopoulou. TESLA-based defense against pollution attacks in p2p systems with network coding. In *International Symposium on Network Coding (NetCod)*, 2011.
22. Y. Li, H. Yao, M. Chen, S. Jaggi, and A. Rosen. RIPPLE authentication for network coding. In *Proc. of INFOCOM'10*, pages 2258 – 2266, 2010.
23. L. Lima, J. P. Vilela, P. F. Oliveira, and J. Barros. Network coding security: Attacks and countermeasures. *CoRR*, abs/0809.1366, 2008.
24. A. Perrig, R. Canetti, J. D. Tygar, and D. Song. The TESLA broadcast authentication protocol. *RSA CryptoBytes*, 5(2):2–13, Summer/Fall 2002.
25. J. P. Vilela, L. Lima, and J. Barros. Lightweight security for network coding. In *Proc. IEEE Int. Conf. on Communications*, 2008.
26. Y. Wang. Insecure "provably secure network coding" and homomorphic authentication schemes for network coding. IACR Eprint archive, 2010.
27. R. W. Yeung. *Information Theory and Network Coding*. Springer Publishing Company, Incorporated, 2008.
28. Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan. An efficient scheme for securing XOR network coding against pollution attacks. In *Proc. IEEE INFOCOM*, 2009.