Comparison of Different Secure Network Coding Paradigms Concerning Transmission Efficiency

Stefan Pfennig and Elke Franz
Technische Universität Dresden, Faculty of Computer Science
01062 Dresden, Germany
Email: {stefan.pfennig|elke.franz}@tu-dresden.de

Abstract—Preventing the success of active attacks is of essential importance for network coding since even the infiltration of one single corrupted data packet can jam large parts of the network. The existing approaches for network coding schemes preventing such pollution attacks can be divided into two categories: utilize cryptographic approaches or utilize redundancy similar to error correction coding. Within this paper, we compared both paradigms concerning efficiency of data transmission under various circumstances. Particularly, we considered an attacker of a certain strength as well as the influence of the generation size. The results are helpful for selecting a suitable approach for network coding taking into account both security against pollution attacks and efficiency.

I. INTRODUCTION

Network Coding introduced by Ahlswede et al. [1] is an opportunity to achieve the min-cut-max-flow of a network (multicast capacity) in a multicast scenario. To reach this multicast capacity, intermediate nodes do not solely forward data packets but compute and forward algebraic combinations of these packets.

However, combining packets enables attackers to jam large parts of the network by infiltrating only one so-called polluted packet. A polluted packet does not belong to the linear span of the valid data packets. The problem is the propagation of corrupted packets to all subsequent nodes which might prevent successful decoding at the receiving nodes. Thus, there exist various network coding schemes which aim at limiting the impact of an active attacker. In the following, we refer to these schemes as Secure Network Coding (SNC) schemes.

Most SNC schemes we recently analyzed utilize cryptographic approaches such as homomorphic signatures [2], homomorphic Message Authentication Codes (MACs) [3], homomorphic hashes [4], or checksums [5]. They all have in common that intermediate nodes are able to check the validity of incoming packets. If the validation of a packet fails, it is discarded; otherwise, it can be included when computing the combined outgoing packets. Hence, approaches based on cryptography require that intermediate nodes know keys, hashes, or checksums, and that they can handle the increased requirements on computing power, time, and memory.

Intermediate nodes might not be able to fulfill such increased requirements in every scenario. Network error correction, introduced in [6], works similar to classical coding theory and does not require the intermediate nodes to perform additional tasks. Achieving security against active attackers by utilizing coding theory was studied, e.g., in [7], [8]. In these schemes, intermediate nodes cannot check the validity of data packets they receive, they only need to combine these packets. Thus, a polluted packet will influence all subsequent paths to the receivers. The key idea is that the receivers can interpret the initial polluted packet as an extra packet. The other corrupted data packets are linear combination of valid packets and the extra packet. With the help of some redundancy, it is possible to recover the original data.

Within this paper, we compare SNC schemes based on cryptography (CryptoSNC) and SNC schemes based on introducing redundancy (CodingSNC) concerning the efficiency of data transmission under various circumstances. We particularly focus on the rateless scenario that does not require knowledge about the network topology or the attacker strength. On the one hand, we consider active attackers of a certain strength; on the other hand, we analyze the influence of the generation size for the different approaches. We present experimental results for these scenarios that help in selecting a suitable approach and parameters for network coding taking into account both security against pollution attacks and efficiency.

The paper is organized as follows. In Section II, we describe the network topology, the attacker model, and give an overview on different SNC schemes. Section III illustrates the simulation environment and efficiency metrics; in Section IV, we present the results of our experiments. Section V concludes and gives an outlook on further research.

II. SYSTEM MODEL

A. Network Topology and Attacker Model

The network topology can be described by a directed graph $G = (V, E)$ with nodes (vertices) $V$ and edges $E$. An edge $e = (a, b)$, $e \in E$, $a, b \in V$ implies that node $a$ can send packets to node $b$. The set of sending (source) nodes $S$, forwarding (relay) nodes $F$ and receiving (sink) nodes $R$ is a partition of $V$. To simplify matters, we assume exactly one sender ($|S| = 1$) and unit capacity on each edge.

Within this paper, we use a network model which is parametrized in breadth (number of receiving nodes $|R|$) and in depth (number of hops $k$ from sender $s$ to receivers $R$). The basic network graph consists of $k$ levels of $\ell$ nodes, where each node on one level is connected to each node on the subsequent
level. Thus, the number of outgoing links $\eta$ of each forwarding node is equal to $\ell$.

This basic network graph where each level is fully connected ($\eta = \ell$) to the next level can be restricted in addition. We can limit $\eta$ to decrease the number of send operations in the network. To stay balanced, we require that the number of outgoing links of each forwarding node on level $\kappa$ equals the number of incoming links of each node on level $\kappa + 1$. Fig. 1 presents an example for $\eta = 2$ and $k = 3$. Forwarding nodes with one incoming edge only forward the data packets they receive; only forwarders with more than one incoming edge actually perform network coding.

Since we focus on schemes preventing the success of pollution attacks, we consider only active attackers. The attacker can be described by the number of links $|E'|$ or nodes $|V'|$ he controls and by his computational power. Basically, we can distinguish two types of active attackers: Outsiders who can only jam links $E' \subseteq E$ and insiders who can control nodes $V' \subset V$ and, hence, can control all outgoing links of these corrupted nodes. Within our first experiments, we focused on outsiders. Thereby, an outsider was simulated by a probability $p$ which determine whether a link $e \in E$ is corrupted. Additionally, we assume that an attacker cannot control the outgoing links of the sender since such an attack has the same influence for both CodingSNC and CryptoSNC.

B. CodingSNC vs. CryptoSNC

Jaggi et al. have shown in [7] that the impact of a limited attacker who controls $z$ links in a network with multicast capacity $C$ can be limited to the optimal rate $C - z$. In case of a computationally unbounded attacker, a secret channel is needed in order to reach this optimal rate [8].

An implementation of CodingSNC is introduced in [7]. The data to be transmitted is represented by a matrix $X = [IM]$ that consists of $h$ packets of $n$ bytes each; $I$ is an identity matrix and $M$ the actual message. The secret necessary to decode even in case of a polluted packet is an $n \times C$ parity check matrix $P$. Based on this parity check matrix, a hash matrix $H = XP$ of size $h \times C$ is computed. A possibility for choosing the hash matrix $H$ independently from $X$ is introduced in [8]. In that algorithm, $X$ is expanded with a “modification matrix” $L$ to satisfy $H = XP = [IML]P$.

The forwarding nodes $f \in F$ apply usual random linear network coding. They receive $l$ data packets $x_i = (x_{i,1}, x_{i,2}, ..., x_{i,n})$, $x_{i,j} \in \mathbb{F}_q$, select $l$ random coefficients $\alpha_i \in \mathbb{F}_q$, and compute linear combined packets $y$ for each outgoing link

$$y = \sum_{i=1}^{l} \alpha_i x_i.$$  

Only the receivers $R$ need the secret. Each of them computes $S = YP - \hat{T} H$, where $S$ is the syndrome matrix, $Y$ is the matrix of received packets and $\hat{T}$ represents the network transformation, i.e., $\hat{T}$ corresponds to the $I$ in the original data matrix $X$.

If $S$ is an all-zero matrix, i.e., the rank $\text{rk}(S) = 0$, there is (with high probability) no corrupted packet in $Y$. In case of $\text{rk}(S) > 0$, there are $\text{rk}(S)$ corrupted packets. For multicast capacity $C$ and only $C - z$ independent data packets in $X$, there are $r$ redundant packets. Thus, the receiver can deduct up to $z$ corrupted packets from $Y$ which means that he can successfully decode all payload data as long as $\text{rk}(S) \leq z$. Since a polluted data packet cannot be discarded by forwarders, it is included in the subsequent processing. However, linear combinations that contain the polluted data packet do not further increase $\text{rk}(S)$.

The detection of polluted packets works differently for CryptoSNC. In contrast to CodingSNC, forwarding nodes $F$ are in most cases also able to check the validity of received data packets by means of some secret information. Hence, they can discard corrupted packets and use only valid packets for computing linear combinations according to (1).

Consequently, a polluted packet influences decodability only if it decreases the overall multicast capacity $C$. For example, this would be the case if the attacker controls a direct link to a receiver. To ensure decodability independent of the position of the attacker, we also need redundant data packets. Similar to CodingSNC, we limit the payload packets to $h = C - z$ and add $z$ redundant packets which actually are linear combinations of the payload packets. Hence, we can guarantee the decodability of $\hat{Y}$ for up to $z$ corrupted packets.

C. Predetermined Redundancy vs. RatelessSNC

We assume an implementation for network coding according to Practical Network Coding [9], i.e., data to be sent is organized in generations of size $h \times n$ symbols, and each packet contains a global encoding vector (GEV) of $h$ symbols. Thus, each generation contains $h$ linear independent data packets.

Basically, the generation size $h$ can be set to the multicast capacity $C$ since this is the maximum number of data packets that can be transferred to the recipients in one transmission round. Lowering the generation size $h$ is equivalent to explicitly consider $z$ redundant packets per transmission round. We assume that retransmission of the whole generation is necessary if decoding was not possible at all recipients. Since CodingSNC is targeted to an attacker of a given strength, CryptoSNC is likely to provide better results in this scenario.
In practice, it is not realistic to assume knowledge about the network topology and the attacker strength. For operating without such knowledge, rateless coding was applied for network coding. The sender sends data packets from the current generation until he gets an acknowledgment from the recipients which implies that they store data packets of one generation (e.g. [10], [11]). Recently, the application of CodingSNC in the rateless scenario was also introduced [12].

The rateless approach is advantageous in cases of errors – since there is no need to resend the whole generation, the sending overhead decreases. However, the latencies increase. In the rateless scenario, the analyzed approaches are referred to as CryptoRSNC and CodingRSNC, respectively.

III. EVALUATION

A. Simulation of the Approaches

For the evaluation of the different approaches, we simulated CodingSNC, CryptoSNC, CodingRSNC, and CryptoRSNC using Sage [13] without implementing complete algorithms. Rather, we focused on the question whether successful decoding at the receiving nodes is possible for a given scenario. Simulation of CodingSNC means that receiving nodes compute the syndrome matrix $S$ and $\text{rk}(S)$. Decoding the matrix of received packets $Y$ is possible only if $\text{rk}(Y) - \text{rk}(S) = h$ after one transmission round, or in other words, if $\text{rk}(S) \leq z$.

Simulating CryptoSNC means that forwarding and receiving nodes detect and discard polluted data packets. Hence, the matrix of received data packets $Y$ contains only valid data packets. Successful decoding is possible for the receiving nodes if $\text{rk}(Y) = h$ after one transmission round.

In case of the rateless scenario, the matrix of received data packets $Y$ is filled over a number of transmission rounds. Decoding is possible if $h$ linear independent data packets have been received. Again, intermediate nodes filter out polluted data packets only in case of CryptoRSNC. For CodingRSNC, decoding is possible if $\text{rk}(Y) - \text{rk}(S) = h$ i.e., the number of linear independent packets minus the number of corrupted packets should match the generation size.

Parameters for all simulation runs are the network topology based on the model introduced in Sec. II, the generation size $h$, and the number of redundant packets $z$. The network topology, defined by $k$, $\ell$, and $\eta$, determines the multicast capacity $C$ that represents the maximum of $h$ for CodingSNC and CryptoSNC.

Since we assume a random attacker model, the results are influenced by this random value, we have to repeat the simulation runs for each scenario multiple times and average the computed efficiency parameters to get meaningful results.

B. Efficiency Metrics

There are various possibilities to measure efficiency depending on the application. Within our experiments, we focused on the question whether all intended recipients can decode all data packets sent. As a measure, we computed the transmission efficiency that is generally the ratio of data sent by the sender to the sum of decodable data of successful transmission round(s). There are different possibilities for defining the data sent by the sender which implies different efficiency metrics.

First, we evaluate the efficiency of data transmission referring to data packets. In one transmission round, the sender sends data packets over all outgoing edges $(s,f_{1,i})$ for $i = 1, 2, ..., \ell$.

Let $\text{in}(v)$ be a function that delivers the matrix of all (valid) input packets for node $v \in V$, $|\text{out}(v)|$ the number of all output packets of $v \in V$, and $\text{dec}(Y)$ be a function such that

$$\text{dec}(Y) = \begin{cases} h, & \text{if } \text{dec}(Y) = h \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

We consider a transmission round to be successful only if all receivers $r \in R$ can decode. Hence, we can define the packet transmission efficiency $E_p$ by

$$E_p = \frac{\min_{r \in R} \text{dec}(\text{in}(r)) \cdot |R|}{|\text{out}(s)|} \quad (3)$$

As an example, consider a network with $\eta = \ell = 4$. If $s$ sends $h = C = 4$ packets and only 3 of the 4 receivers can decode, $\min_{r \in R} \text{dec}(\text{in}(r)) = 0$, hence, $E_p = 0$. If all receivers can decode, $E_p = \frac{4}{4} = 4 \approx 100\%$. That means, in case of a successful multicast transmission, we always get more than 100% which is reasonable in comparison to routing the same amount of data by means of unicast transmissions.

In case of the rateless scenarios, we count the transmission rounds needed for transmission of one generation to all receivers $r \in R$ and divide this number by the generation size $h$. In the first place, this seems to be a different definition, but we have to consider that it is necessary to average the transmission efficiencies for a large number of repetitions to get meaningful results. For example, we get the same packet transmission efficiency if either all messages are decodable by all receivers every second generation in CryptoSNC or all messages are decodable after two rounds for CryptoRSNC.

Another possibility to describe the data sent by the sender is to refer to the payload of the data packets. This approach allows for a fairer comparison of different algorithms for CryptoSNC and CodingSNC since their communication overhead may be rather divergent. If we relate the efficiency to the payload, we have to define the payload size and the field size $q$ as further parameters for the simulation. For a fixed packet size, we can estimate the payload per packet of a certain SNC scheme in dependency on the generation size and on the communication overhead of that scheme according to [14]. Multiplying the payload $r$ with the packet transmission efficiency $E_p$ yields the data transmission efficiency $E_d$:

$$E_d = E_p \ast r \quad (4)$$

IV. DISCUSSION OF THE RESULTS

A. CodingSNC vs. CryptoSNC

For first experiments, we used the fully meshed network (Sec. II) with $k = 3, \eta = \ell = 4$ and an outside attacker who can corrupt links with a probability $p = 0.1$. For the given network, there are 32 links between the 3 levels that can be
corrupted. Hence, there are 3.2 corrupted links on average in the given network. Since that network has a multicast capacity of \( C = 4 \), \( h \) can be at maximum 4, too. Consequently, there are four different settings for the simulations.

We measured the rank of the received (valid) packets \( \text{rk}(\mathbf{Y}) \) or the rank of all received packets minus the rank of the syndrome matrix \( \text{rk}(\mathbf{Y}) - \text{rk}(\mathbf{S}) \), respectively, for 1000 repetitions. Hence, we derived the probability that \( \mathbf{Y} \) is decodable for all possible values of \( z \).

C. Data Transmission Efficiency

As expected, a larger generation size implies higher efficiency for both approaches. However, the increase of efficiency is not linear and not boundless. Additionally, delay and processing time increase and there are also a higher computational effort, memory requirements, etc. Finally, a higher generation size also implies an increase of the GEV which decreases the payload per packet if we assume a limited packet size.

To substantiate these assumptions, we analyzed a network with \( \eta = 2, k = 3, \ell = 4 \) (Fig. 1) and measured \( E_p \) and the time \( t \) needed for computing (Fig. 4).

Of course, these diagrams depend on the network topology; however, they visualize that the increase in time \( t \) grows faster than linear, whereas the increase in transmission efficiency \( E_p \) decelerates and converges to a horizontal asymptote. Consequently, a suitable value of \( h \) has to be defined as a trade-off between \( E_p \) and other parameters like the time \( t \).

C. Data Transmission Efficiency

As expected, a larger generation size implies higher efficiency for both approaches. However, the increase of efficiency is not linear and not boundless. Additionally, delay and processing time increase and there are also a higher computational effort, memory requirements, etc. Finally, a higher generation size also implies an increase of the GEV which decreases the payload per packet if we assume a limited packet size.

To substantiate these assumptions, we analyzed a network with \( \eta = 2, k = 3, \ell = 4 \) (Fig. 1) and measured \( E_p \) and the time \( t \) needed for computing (Fig. 4).

Of course, these diagrams depend on the network topology; however, they visualize that the increase in time \( t \) grows faster than linear, whereas the increase in transmission efficiency \( E_p \) decelerates and converges to a horizontal asymptote. Consequently, a suitable value of \( h \) has to be defined as a trade-off between \( E_p \) and other parameters like the time \( t \).

C. Data Transmission Efficiency

As expected, a larger generation size implies higher efficiency for both approaches. However, the increase of efficiency is not linear and not boundless. Additionally, delay and processing time increase and there are also a higher computational effort, memory requirements, etc. Finally, a higher generation size also implies an increase of the GEV which decreases the payload per packet if we assume a limited packet size.

To substantiate these assumptions, we analyzed a network with \( \eta = 2, k = 3, \ell = 4 \) (Fig. 1) and measured \( E_p \) and the time \( t \) needed for computing (Fig. 4).

Of course, these diagrams depend on the network topology; however, they visualize that the increase in time \( t \) grows faster than linear, whereas the increase in transmission efficiency \( E_p \) decelerates and converges to a horizontal asymptote. Consequently, a suitable value of \( h \) has to be defined as a trade-off between \( E_p \) and other parameters like the time \( t \).

C. Data Transmission Efficiency

As expected, a larger generation size implies higher efficiency for both approaches. However, the increase of efficiency is not linear and not boundless. Additionally, delay and processing time increase and there are also a higher computational effort, memory requirements, etc. Finally, a higher generation size also implies an increase of the GEV which decreases the payload per packet if we assume a limited packet size.

To substantiate these assumptions, we analyzed a network with \( \eta = 2, k = 3, \ell = 4 \) (Fig. 1) and measured \( E_p \) and the time \( t \) needed for computing (Fig. 4).

Of course, these diagrams depend on the network topology; however, they visualize that the increase in time \( t \) grows faster than linear, whereas the increase in transmission efficiency \( E_p \) decelerates and converges to a horizontal asymptote. Consequently, a suitable value of \( h \) has to be defined as a trade-off between \( E_p \) and other parameters like the time \( t \).

C. Data Transmission Efficiency

As expected, a larger generation size implies higher efficiency for both approaches. However, the increase of efficiency is not linear and not boundless. Additionally, delay and processing time increase and there are also a higher computational effort, memory requirements, etc. Finally, a higher generation size also implies an increase of the GEV which decreases the payload per packet if we assume a limited packet size.

To substantiate these assumptions, we analyzed a network with \( \eta = 2, k = 3, \ell = 4 \) (Fig. 1) and measured \( E_p \) and the time \( t \) needed for computing (Fig. 4).

Of course, these diagrams depend on the network topology; however, they visualize that the increase in time \( t \) grows faster than linear, whereas the increase in transmission efficiency \( E_p \) decelerates and converges to a horizontal asymptote. Consequently, a suitable value of \( h \) has to be defined as a trade-off between \( E_p \) and other parameters like the time \( t \).
The curve denoted as raw corresponds to $E_p$. Since $E_p$ does not consider the actual payload per packet, this value is generally larger than $E_d$. The selected CryptoRSNC schemes are based on Practical Network Coding (PNC, [9]), i.e., each packet contains a GEV with $h$ coefficients. Hence, there is a payload of $1400 - h$ bytes for PNC. The maximum efficiency can be achieved for $h \approx 40$. Regardless of the actual efficiency value for the CryptoRSNC schemes, we can assert that all these schemes have a maximum value of $E_d$ for $h < 40$, because the overhead of PNC is a lower bound for all CryptoRSNC. Maximum values for $E_d$ have been achieved for DART [5] at $h \approx 25$, for the RSA-based scheme RSA [2] at $h \approx 16$, for HMAC [3] at $h \approx 8$, and for HH [4] at $h \approx 2$. The decreased efficiency is caused by additional data to be sent as well as by the necessity to increase the field size for some of the schemes.

In this evaluation, the RSA-based scheme would yield best efficiency. However, we want to point out that $E_d$ depends on the topology itself [15]. The key aspect is, that there always exists an optimal generation size $h$ for a given network.

V. Conclusion

Within the experiments presented in this paper, we compared CodingSNC and CryptoSNC to give some answers to the question which approach should be preferred to strengthen network coding schemes against pollution attacks regarding efficiency. There are various factors that have to be considered; we worked with a basic network topology, a varying generation size, and an outside attacker. For a comparison of the approaches, we evaluated the transmission efficiency.

Just considering the overhead caused by the approaches, CodingSNC seems to be preferable since there is no need to transfer secret information to forwarding nodes, and there is no computational overhead for the forwarders. CryptoSNC requires additional overhead for key management, but CodingSNC also necessitates a secret channel or a shared secret.

However, CryptoSNC outperforms CodingSNC if we assume an outside attacker who can control links with a certain probability. To conclude, CodingSNC will maximally perform as good as CryptoSNC, but mostly worse than CryptoSNC. Thus, CryptoSNC should be preferred if possible.

Furthermore, our experiments delivered concrete results regarding the question to which degree a larger generation size improves efficiency for rateless SNC. Nevertheless, we want to point out that the increase of efficiency is limited. The higher the generation size, the lower the gain in efficiency. Simultaneously, increasing the generation size causes a rapid increment of grow for delay and required computing time. Taking into account the data transmission efficiency $E_d$, best results were achieved for manageable generation sizes $h < 40$ given a fixed packet size of 1400 bytes.

Future work is necessary to study the influence of additional attacker models including also inside attackers. Further, we will also investigate other network topologies in order to find the best suited SNC scheme as well as best suited settings for parameters like the generation size $h$. Thereby, we also have to consider other performance metrics in addition to the transmission efficiency. Finally, we are working on the question how to design the network flow to minimize the impact of an attacker.

References


