Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2012, Article ID 184783, 14 pages
doi:10.1155/2012/184783

Research Article

Selective Forwarding Attacks against Data and ACK Flows in Network Coding and Countermeasures

Yuanyuan Zhang[1,2] and Marine Minier[1]

[1] *CITI laboratory, INSA-Lyon, INRIA, Université de Lyon, 69621 Lyon, France*
[2] *Department of Computer Science and Technology, East China Normal University, No. 500 Dongchuan Road, Shanghai 200241, China*

Correspondence should be addressed to Yuanyuan Zhang, yyjess@gmail.com

Received 27 April 2012; Revised 30 July 2012; Accepted 27 September 2012

Academic Editor: Gildas Avoine

Network coding has attracted the attention of many researchers in security and cryptography. In this paper, a well-known attack *selective forwarding attack* will be studied in network coding systems. While most of the works have been dedicated to the countermeasures against pollution attacks where an attacker modifies intermediate packets, only few works concern selective forwarding attacks on data or acknowledgment (ACK) packets; those last ones are required in network coding. However, selective forwarding attacks stay a real threat in resource constraint networks such as wireless sensor networks, especially when selective forwarding attacks target the acknowledgment (ACK) messages, referred to as *flooding attack*. In the latter model, an adversary can easily create congestion in the network and exhaust all the resources available. The degradation of the QoS (delay, energy) goes beyond the capabilities of cryptographic solutions. In this paper, we first simulate and analyze the effects of selective forwarding attacks on both data flows and ACK flows. We then investigate the security capabilities of multipath acknowledgment in more details than in our original proposal (Zhang et al., 2011).

1. Introduction

Network coding is a very active field of both information theory and networking for information dissemination. It consists in encoding a message into several packets and transmitting those packets in an oriented multicast way through the network to the destination. The intermediate nodes can also combine the received packets. It has been shown that network coding could reach the maximum possible information flow in a network. Network coding is also very interesting for security. Many works have been interested in demonstrating the security capacity of network coding. Two security worlds coexist, and the border is delimited by the adversary capabilities. Network coding can be used to bring secrecy if the adversary eavesdropping capabilities are bounded (see [1–3]). Otherwise, cryptography and security must be used to defeat more powerful adversaries [4–6]. This paper falls in the second class of works related to network coding and security.

In network coding, two information flows are identified: the data flow and the acknowledgment (ACK) flow. Both flows can be targeted by an adversary with different consequences. An adversary attacking the data flow wants to affect the messages produced by different sources and decoded by the destinations. An example of such an attack is *pollution attacks* [6]. Many works have proposed countermeasures against *pollution attacks* [4, 5, 7–10]. Another classical attack on data flow is selective forwarding attack where an adversary drops/delays all or part of the data packets he receives. As shown in [11], this kind of attacks is defeated by network coding due to its intrinsic multipath nature. In this paper, we first show by simulations this result; selective forwarding attacks on the data flow are inefficient when network coding is employed in the network.

Finally, attacks against the ACK flow have less attracted the attention of the security community. It does not mean that threats against the ACK flow are less dangerous than those on the data flow, quite the contrary. Threats against the

ACK flow can be partially defeated by some cryptographic techniques. But it is not enough to prevent attacks against the quality of services (QoS). Attacking the ACK flow can create congestion or exhaust the nodes energy by flooding the network with useless packets. Up to our knowledge, Dong et al. [6, 11] are the only ones referring to attacks against the ACK flow in network coding with the DROP-ACK attack [6]. The threats considered in this paper have all the same consequence: flooding. Unfortunately, the solutions found against flooding in classical networks [12] are all dedicated to TCP and cannot be applied in our context.

In this paper, we first give simulation results concerning the effects on selective forwarding attacks first targeting the data flow and second the ACK flow. From those simulations, we observe that first and as expected selective forwarding attacks targeting the data flow are inefficient when network coding is activated in the network and second that attacks against the ACK flow could be really efficient. We then propose a dedicated mechanism based on multipath routing of ACK packets to discard flooding attacks when the adversary drops or delays the ACK packets. We then provide some results concerning global evaluation of the security of network coding when selective forwarding attacks on data and on ACK flows are combined.

In Section 2, network coding and selective forwarding attacks are described as well as related works. Section 3 presents our network and adversary models and describes our multipath ACK back strategy to prevent flooding attacks together with some implementation issues. Section 4 gathers all our simulation results concerning selective forwarding attacks and flooding attacks against first classical network coding (without our multipath strategy) and second network coding with our multipath strategy. We finally show that classical network coding is efficient against selective forwarding attacks and that our network coding multipath ACK strategy is efficient against flooding attacks and sum up those results in Section 5.

2. Preliminaries

In this section, we remind the basic elements on network coding and the related work on flooding attacks.

2.1. Network Coding. The seminal work on network coding was done by Ahlswede et al. in [13]. The main aim of network coding is to find optimal information dissemination in a network. It has been shown that network coding can also improve the network resilience against communication failure, for example, erasure, [14, chapter 1]. Wireline and wireless networks can benefit from network coding. For more details on network coding and on the problems solved by this technique, the readers can consult [14–16].

An important topic in network coding is linear codes: packets exchanged by the nodes are linear combinations of the data to be transmitted over a given finite field. Random linear network coding [17] has particularly attracted attention. The coding process is as follows. Let us assume a network viewed as a graph with a source node and some destination nodes. Let us denote $D = (d_1, d_2, \ldots, d_n)$ a data of kn bits viewed as a vector of n fragments $d_i \in \mathbb{F}_{2^k}, i \in [1, n]$. The messages $m_j = h_j \| p_j$ transmitted by the source and the relaying nodes in a scheme using random linear network coding consist in a header h_j and a payload p_j:

$$p_j = \sum_{i=1}^{n} \alpha_{i,j} d_i, \tag{1}$$

where the coefficients $\alpha_{i,j}$ are chosen randomly over $\mathbb{F}_q$ with $q = 2^u$ the favorite choice in the literature. The header h_j contains all the coefficients $\alpha_{i,j}$ which describe the payload:

$$h_j = \left(\alpha_{1,j}, \ldots, \alpha_{n,j}\right). \tag{2}$$

The source and the relaying nodes apply the coding process infinitely until they receive an acknowledgment (ACK) from all destinations. All destinations run the decoding process: a Gaussian elimination or any other methods for solving linear systems of equations (not described here). In network coding, we have an implicit "*data flow*" which transmits data from the sources to the destination and a *feedback/acknowledgment flow* which carries the ACK from the destination to the sources.

Finally, network coding problems are divided into two classes: *intra-flow* and *inter-flow*. Intraflow network coding corresponds to the example described above: a single message and one or several sources. Interflow network coding combines different messages from different sources at the level of intermediate nodes. This problem is also known as source network coding.

Classically, network coding is used with an oriented multicast strategy that could be compared with a partial flooding of information. This partial flooding allows to obtain the maximum possible information flow in the network.

Generally, in most network environments, the mechanism of transmission of the ACK packets usually employs the routing protocol at the lower routing layer by default. This simplified treatment is enough for most of the upper layer transmission demands in most networks such as TCP/IP because the retransmission will compensate the loss of ACK. However, in network coding environments, the source node continues sending encoding packets until it receives an ACK to confirm the correct decoding at the sink node, so it is crucial to guarantee its arrival.

2.2. Classical Attacks against Network Coding. Three attacks are dedicated to network coding in the security literature: *packet pollution attack* [6, 11], *drop-data packets attack* [11] (also known as selective forwarding attack), and DROP-ACK *attack* [6]. In a pollution attack, an adversary injects invalid packets into the data flow. The adversary exploits the capacity of network coding to spread information at his own advantage. The invalid packets are carried through the network to be only discarded by the destination in the best case. The resources, for example, bandwidth, energy used to carry these packets are lost. Such an attack is extremely powerful in resource-constrained networks such as wireless

sensor networks (WSNs). Many papers are devoted to find countermeasures to pollution attacks [4, 5, 7–10].

Selective forwarding attack is a well-known and very harmful attack in wireless multihop networks for example described in [18]. In a selective forwarding attack, a compromised node refuses to forward some of the packets in its outstanding buffer, such as control information or data packets in order to cut off the packets propagation. An extreme example of this attack is a two-step attack where first a malicious node attracts most of the local traffic using, for example, false neighbors information, and then the malicious node completely suppresses the received packets transmission provoking what is usually known as a *black hole attack*. Selective forwarding attacks will not always happen on the data flow but also on controlling packets such as HELLO packets or acknowledgment packets. When it is applied on ACK, we talk here about *flooding attacks*.

Selective forwarding attacks have been studied [11] in the context of network coding where the adversary drops or delays packets of the data flow. By its intrinsic nature, network coding process uses several routes to transmit a message, and the consequences of this attack will be essentially to introduce a delay as shown in [11] but not to prevent the data to reach the destination. Some additional methods [19] coming from the routing world can also help improving the damaged throughput and to decrease the delay.

A DROP-ACK attack [6, 11], or flooding attacks as it is referred throughout the paper, targets the ACK flow. Everything happens after a destination successfully decodes D and starts to forward an ACK. Attacking the ACK flow can be particularly interesting for the adversary: preventing the ACK to reach the source can increase the congestion in the network, prevent a given source to transmit new information, or exhaust the energy of all nodes forwarding the packets (see Figure 1).

From the perspective of the classical man-in-the-middle adversary model, three attack strategies are possible against the ACK flow: injecting/modifying ACK, dropping/removing ACK, and delaying ACK. The last two attacks are the ones leading to a flooding attack.

(a) Injecting/Modifying ACK. Charlie attempts to forge an ACK packet and sends it to Alice. She can believe that Bob has received enough information to recover D. Such attacks can be prevented by a proper use of cryptography, that is, by using a message authentication code (MAC) [20–22] and key distribution [23].

(b) Dropping/Removing ACK. Charlie is seen as a black hole attacker by destroying any ACK packet. Charlie can also just modify the ACK delivery path to prevent the packet to reach Alice. As a result, Alice continues indefinitely sending encoded packets to Bob and so wastes resources. This attack is very difficult to detect.

(c) Delaying ACK. In this case, instead of dropping the ACK packet, Charlie has just to delay the delivery of the packet. This behavior is difficult to distinguish from the selfishness behavior when nodes want to reduce their own energy consumption. As a result, this attack increases the time needed to pass to the next set of packets, and it implies a node energy waste and additional transmission delay.

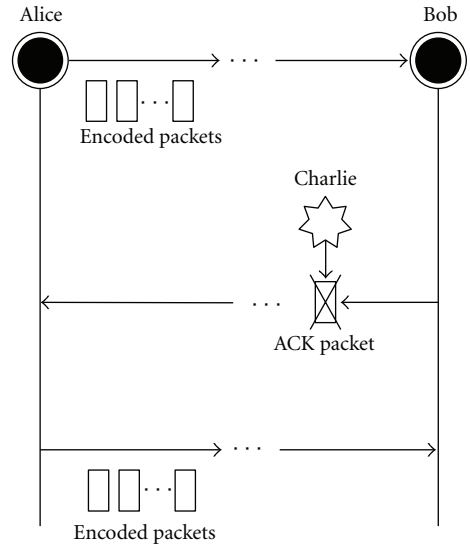


Figure 1: An example of flooding attack. Alice is the source who attempts to send encoded packets to the destination, Bob. Bob is supposed to forward an ACK to Alice once he successfully decodes a message. The adversary, Charlie, drops or delays the ACK: Alice never stops to transmit packets to Bob.

3. General Assumptions, Implementation Aspects, and Our Proposal

In this section, we provide all the hypotheses made concerning the network, the adversary models, and the implementation of the network coding process. We also provide in Section 3.3 our multipath ACK back strategy to discard flooding attacks.

3.1. General Network Assumptions and Adversary Models. In our proposal, we focus on large-scale static wireless sensor networks as case study with two types of nodes: low-power sensor nodes and a single collecting point which we call the sink.

In our approach, all low-power sensor nodes are exactly the same. In our implementation, we use a general multistream unicast scenario as a network coding mechanism. Every sensor node has 100 raw messages to be encoded and delivered to a single destination which is the sink. So, from this point, we talk about the destination or the sink without distinction. A source sensor node continuously sends one encoded packet per second until it receives the ACK from the sink, and then it starts sending the encoded packets of the next raw message. Meanwhile, all the sensor nodes also play the role of forwarding nodes in the network. The encoded packets are computed using XOR network coding [24]. XOR network coding is a special case of linear

network coding where the coefficients $\alpha_{i,j}$ belong to $\mathbb{F}_2$. Because the coefficients are chosen between 0 and 1, the decoding procedure is much simpler. The destination nodes add the received linear combinations until they recover a single message slice. Repeating the procedure, all slices will be calculated and the original message comes out. In this work, the original message is cut into 10 slices and encoded by XOR network coding method.

In this paper, we assume that the adversary goal is to selectively drop packets in two flows, data, and ACK flows after a communication has begun in the network between a source node and the sink. We also assume that the adversary is an insider; that is, it can capture and corrupt sensor nodes, and then he launches those selective forwarding attacks from those compromised nodes. For the sake of simplicity and as previously mentioned, we distinguish these two attacks, on data and ACK flows, by naming them, respectively, *selective forwarding attack* and *flooding attack*.

Our security goal is to prevent selective forwarding attack depressing the performances of network coding. Specifically, we want to be able to preserve a high probability of successful decoding, to prevent *selective forwarding attack* and *flooding attack* from prolonging the average message decoding time, *flooding attack* from wasting the energy of the network (i.e., the energy cost must stay reasonable), preventing a network coding session from finishing (i.e., to decrease the average decoding time consumption).

3.2. Implementation Aspects. Classically, network coding is implemented using an oriented multicast as routing protocol. However, even if this method guarantees the maximum flow in the network, it is very expensive in terms of energy when considering constrained networks such as sensor networks. To preserve the diversified nature of the neighbors choice of network coding and to limit the energy consumption, we first have based all our implementations at the routing layer level and we decided to use a random version [25] of the gradient-based routing (GBR) protocol [26]—a multihop and multistream unicast routing protocol—underneath the network coding. The choice of the random GBR, as explained in [25], allows to maintain the diversified nature of the next hop neighbor required by network coding and also allows to create at the end of a multipath routing protocol useful for network coding. In all the simulations provided in this paper, we have made those implementation choices for network coding.

3.2.1. Gradient-Based Routing (GBR). GBR was first proposed in [26]. It uses a natural gradient as a metric to forward the query towards source. The metric can be regarded as physical distance, hops, or others. In this work, a query is forwarded based on the hop gradient in the sensor nodes. A node forwards the query to its neighbors including its information level about the queries. After a certain period, every sensor node builds up a *gradient table* (GTable) which indicates the distance to its sinks.

When a source node outwards a packet, it chooses a nexthop node which has the smallest gradient in GTable. Thus, each forwarder node will choose their nexthop in the same way. Finally, the path from source to sink is established ideally.

3.2.2. Random GBR. As the network coding process is only efficient if many forwarders combine/forward the encoded packets, we need to modify the original GBR proposal from single path routing to multipath routing from the source to the sink. To do so, we use [25] where the original version of GBR is randomized. This mechanism works as follows: when a source node outwards a packet, it randomly chooses a nexthop node which has a smaller gradient than him in GTable. So, at each packet sent, the choice for the source node for the next hop is randomly made leading to generate multipath routing as soon as many packets are sent which is the case for the network coding process. In the same way, each forwarder node will choose their own nexthop nodes in the same manner (at each new packet, the next hop is randomly chosen leading to create multipath when the network coding process is used). Notice that, we only allow the packets generated from the same data flow to belong to the encoding process. Each packet traversing through the network will record its path for future use because when the sink has correctly decoded the message, then it sends back through the shortest single path the ACK message. Finally, we will have multipath GBR protocol.

3.3. Our Multipath ACK Strategy against Flooding Attacks. In this section, we describe our multipath ACK scheme strategy and how we have implemented this scheme for the simulations presented in Section 4.

The algorithm we propose to prevent flooding attacks in the network is really simple.

(i) The source node Alice wants to send the data D to Bob. First, she encodes D into a certain number of m_j messages as explained in Section 2.1, and then she sends to r_1 of her neighbors the encoded packets m_j for $j = 1, \ldots$ until she receives an ACK packet.

(ii) Each of the forwarders (i.e., intermediate nodes) forwards and/or combines the received packets m_j sent by Alice to r_2 of its neighbors (note that the process for a forwarder to encode intermediate packets is the same as the one previously described) until the packets reach the sink Bob.

(iii) The sink, after having received at least n encoded packets, begins to try to decode the message D. When Bob receives a sufficient amount of data, he decodes D and sends the ACK packet through p different routes. Those p routes are selected among all the routes received by the sink: each packet m_j brings with it all the intermediate nodes from the source to the destination.

(iv) As soon as the source Alice has received one ACK packet, she stops sending combination of data of D.

The principle of this algorithm is rather simple; however, its implementation is more tricky and depends on the way

the network coding process is performed. In our case, as the network coding is implemented with the help of the random GBR protocol, we derive multipath from it for the ACK flow.

As previously defined, each sensor in the network continuously transmits encoded packets according to network coding scheme. Each encoded packet could choose several nexthop nodes by random GBR protocol. The forwarding nodes generate new encoded packets from the packets buffers and then forward to next-hops.

When the sink collects enough encoded packets of the same data flow, the data flow will be successfully decoded and recovered. Then, the sink must send back an ACK to the source to notify it to stop sending more encoded packets. Using random GBR, we can obtain several paths from the source to the sink. In random GBR, every packet records its route. So, when it arrives at the sink, the route is stored for ACK backsending. The sink maintains a routing table of distinct candidate ACK paths collected from incoming packets. Meanwhile, these paths also satisfy the condition of "the least hop counts" from the sink to the source. Therefore, the sink has many paths to send back ACK; thus the opportunity of ACK being blocked by flooding attackers is reduced.

Multipath ACK scheme is supposed to provide more opportunity to avoid the hijacking of ACK on the paths. The sink is able to choose more than one path from the candidate paths to send ACK.

4. Simulation Results without and with the Multipath ACK Strategy

In this section, we present all our simulation results concerning selective forwarding attacks and flooding attacks, first against classical network coding (without our multipath ACK strategy) and second using our solution after having shortly introduced our simulation environment.

4.1. Simulation Assumptions. All the simulations performed in this paper are carried out using the simulator WSNet [27], an event-driven network and physical layer simulator.

Our simulation results are observed in several scenarios. The result of each scenario is averaged on 20 times simulations run with n sensor nodes, where $n \in [50, 200]$ randomly distributed over a square field of 100 m by 100 m. Each sensor node has a radio range equal to 20 m. We assume that energy consumption of transmitting a packet is twice that of receiving a packet, and each sensor does not expire during the simulation duration time.

In this work, the negative influence by packet loss rate caused by signal degradation or collision in MAC layer is not taken into account, which implies that the source nodes do not retransmit the lost encoded packets but just continue sending encoding packets until the ACK arrives from the sink. The simulation duration time is 150 s. Packet transmission rate at each sensor node is one packet per second.

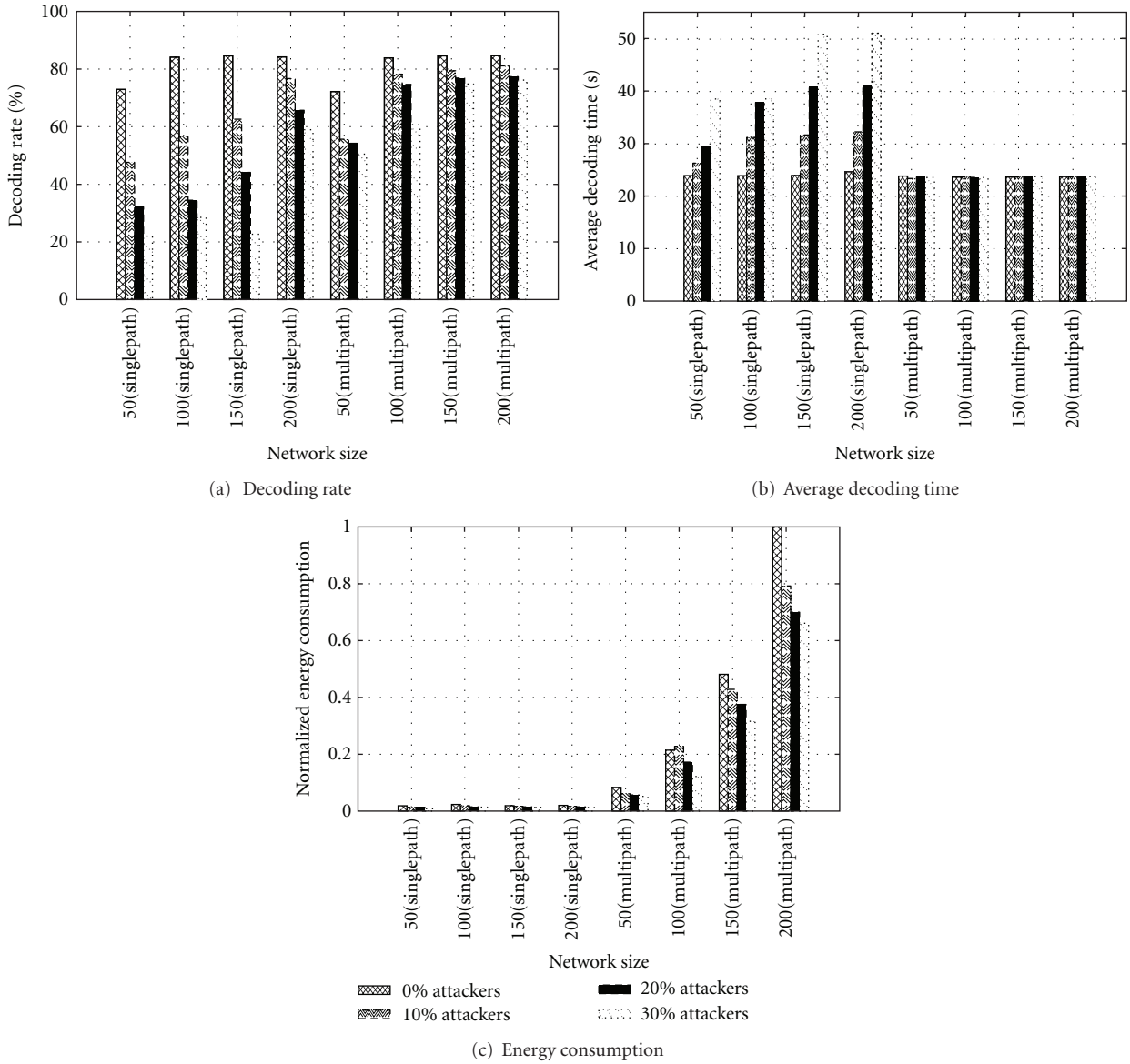
(a) Adversary Strategy. Our adversary is specialized on dropping/removing all data packets and/or all ACK packets passing through him. To do so, he compromises nodes in the network. We assume that he chooses randomly the nodes to compromise. Our adversary is not really clever in the sense that he does not take into account his position in the network. In our simulation, the number of compromised nodes is between 10% and 30% of the total.

(b) Metric. We focus essentially on evaluating the *average probability of successfully decoded messages*. This event occurs when the decoding process is successful for a given message D and when the source node stops forwarding encoded packets for this message; that is, the source receives the ACK. The *decoding rate* denotes this event, that is, the proportion of successfully decoded packets. The *average decoding time* represents the time interval, at the source node, between the moment where a raw message is generated and an ACK packet is received. The *energy consumption* represents the gain in terms of energy between the most expensive solution and the considered solution (a scale between 0 and 1).

4.2. Attacks under Study with Classical Network Coding. In this part, we give simulation results concerning the way the network coding reacts when confronting to first selective forwarding attacks and second flooding attacks when only single ACK path is considered. For comparison purpose, we also give the results for the dummy example "single path network coding strategy" which means that the network coding process works on a single path using classical GBR. In Section 4.2.1 we give the results concerning selective forwarding attacks whereas in Section 4.2.2 we give results concerning flooding attacks. In Section 4.2.3 we give the results concerning the combination of the two previous attacks.

4.2.1. Analysis for Selective Forwarding Attacks. We sum up in Figure 2 the simulation results when the network is confronted to selective forwarding attackers (from 0% to 30% of attackers), considering both network coding used with a single path (i.e., classical GBR) and network coding used with multipath (random GBR). Note that network coding with single path is only a case study which is not really interesting in concrete applications of network coding.

First, it is important to notice that the decoding rate never reaches 100% even when there is no attacker in the network. This is due to the way the simulations are processed: the simulation time is bounded and the simulations stop when the network still works. We do not wait for the successful decoding of all packets. So, all decodings are not completed; this is why the decoding rate never reaches 100%. This fact is more visible on small networks because less packets are sent in the network, leading to reduce the proportion of well-decoded packets (in the sense of our metric). Moreover, XOR network coding is not always a solution for large networks where operations on bigger finite fields are more efficient. Indeed, the number of packets that must be sent in XOR network coding must be more



(a) Decoding rate

(b) Average decoding time

(c) Energy consumption

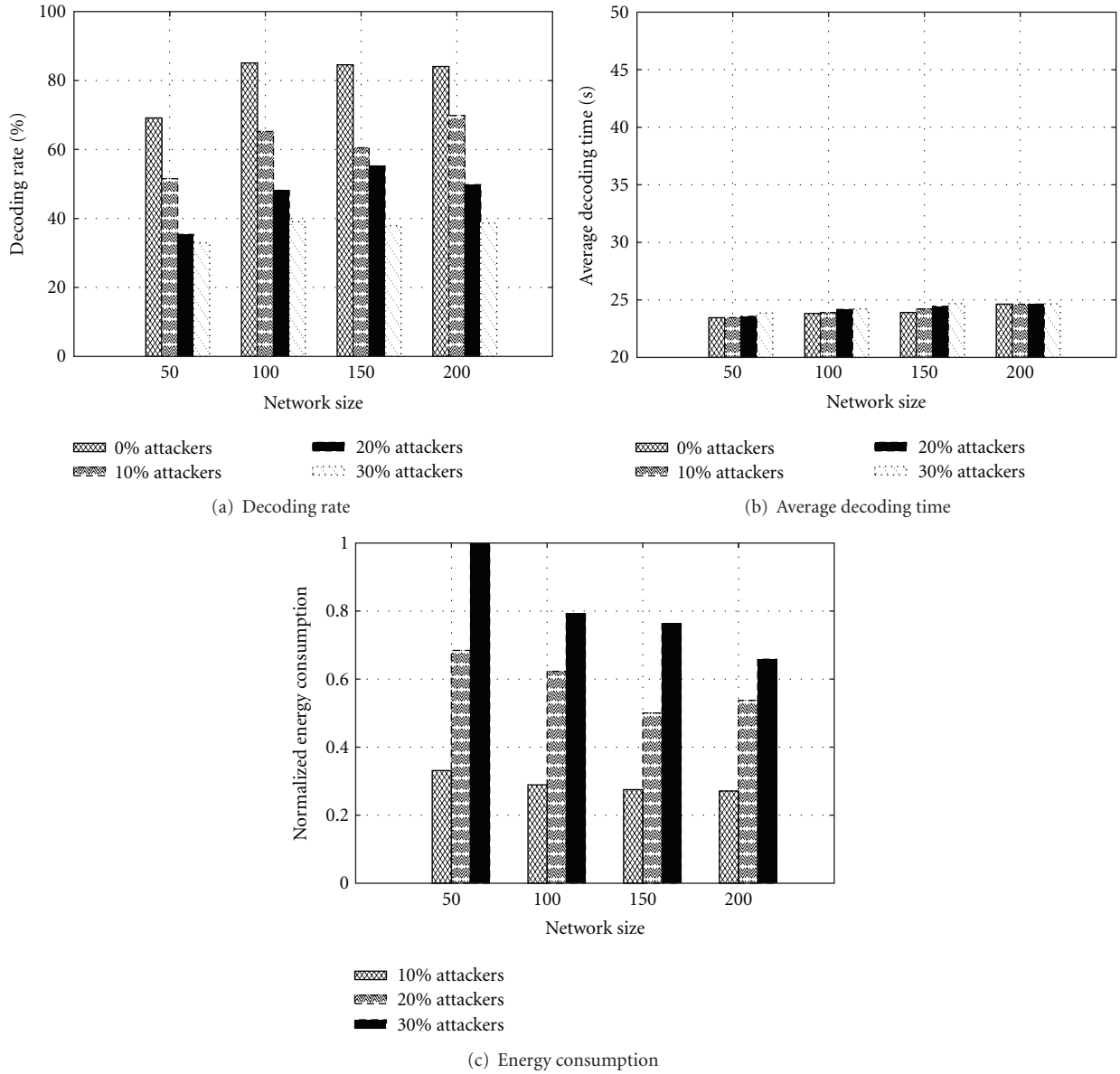
Figure 2: Performance results when single path network coding and multipath network coding are confronting to selective forwarding attackers.

important than in other cases to guarantee a correct decoding at the destination (as shown in [28]). However, we compare the different results performed in the same conditions.

So, we observe in Figure 2(a) that the decoding rate drastically decreases for the single path case when the number of attackers increases whatever the size of the network. For example, whereas the decoding rate is more than 80% when no attackers are present in a 150 nodes network, the decoding rate decreases to about 40% when 20% of the nodes are compromised and down to around 20% when 30% of the nodes are compromised. The degradation is clearly less important when the multipath strategy is used (the worst case is observed for a 50 nodes network where the decoding rate passes from 70% with 0% of attackers down to around 50% when 30% of attackers are present in the network). And larger the network is, less the degradation is important (this remark also holds for the single path case). This is due to the previous remark concerning the bounded simulation time and because, in a larger network, the opportunities of finding more paths are greater.

The average decoding times presented in Figure 2(b) clearly increase in all cases when considering single path GBR whereas the average decoding time (equal to 24 seconds) stays about the same in all cases when considering multipath scheme. This means that when multipath strategy is enabled in a sufficiently dense network, it erases all the negative effects brought by the selective forwarding attackers and makes the average time approaching the ideal value when no attackers are present in the network.

When looking at energy consumption results presented in Figure 2(c), we define the norm value equal to 1 as the biggest energy consumption which is the multipath



(a) Decoding rate

(b) Average decoding time

(c) Energy consumption

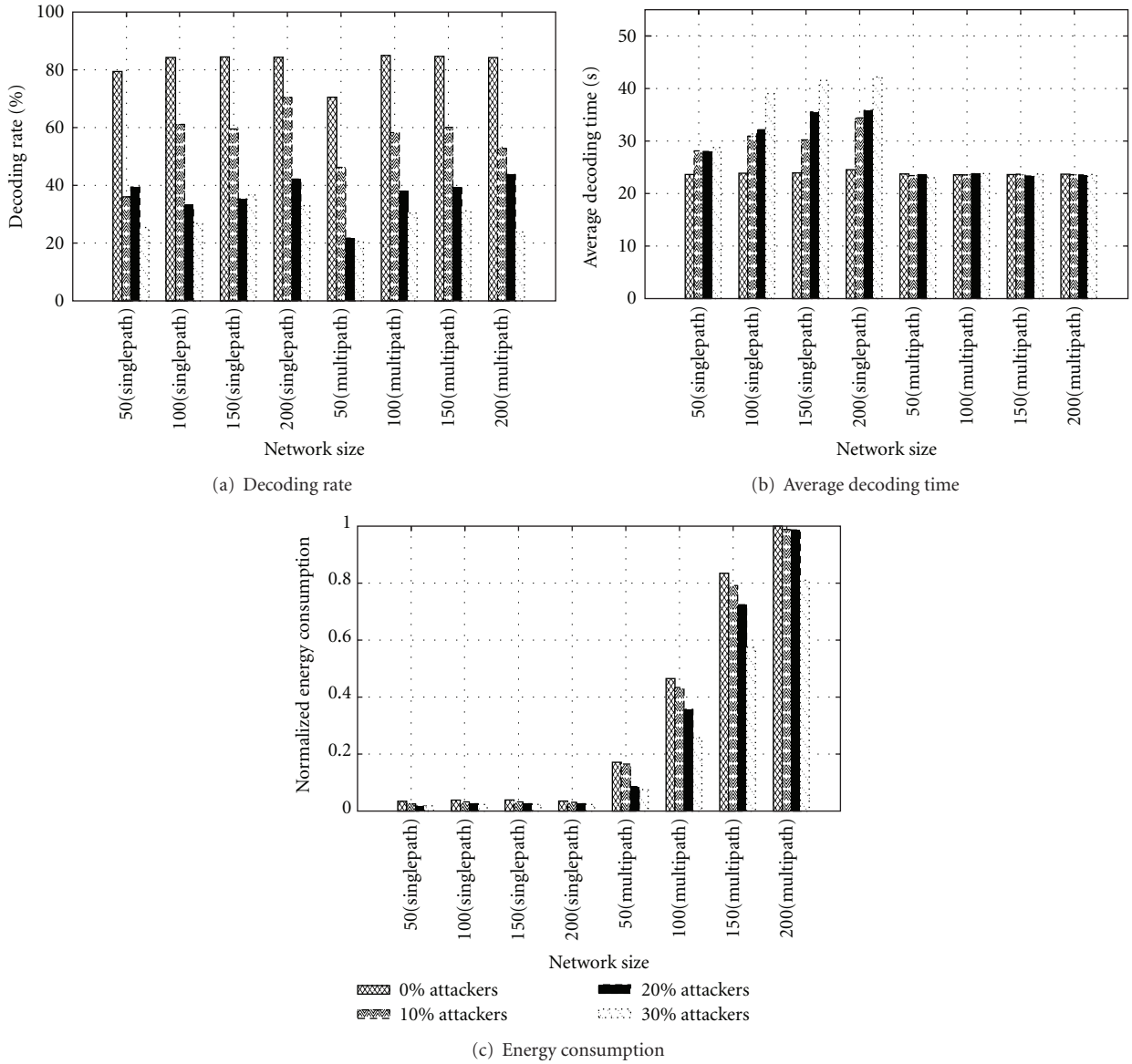
Figure 3: Performance results when multipath network coding is confronting to flooding attackers.

scenario for a 200-node network in Figure 2(c). We observe that, in single path scenarios, the energy consumption is about the same in all cases and is equal to 5% of the normalized value. This is due to the fact that the energy consumption only linearly depends on the length of the path from the source node to the sink. Moreover, in single path scenarios, the energy consumption slightly decreases when the number of attackers increases because the attackers make some packets to disappear as the energy linked with those packets. Multipath scenarios are of course much more energy consuming because several paths are in use. Moreover, bigger the network is, exponentially greater the energy consumption is. This also comes from the previous remark where the possible number of paths exponentially increases according to the size of the network.

In conclusion, we finally state that, as expected, classical multipath network coding strategies are efficient in terms of decoding rate and of average decoding time to defeat selective forwarding attackers on data flows even if the energy cost to pay can be important and even prohibitive when energy preservation is crucial for the considered network (e.g., for highly constrained networks).

4.2.2. Analysis for Flooding Attacks. As the flooding attack concerns the suppression of packets in the ACK flow, we only provide the results for the multipath scheme applied on the data flow.

As in the previous case and for the same reason, when there is no attacker in the network, the decoding rate does not reach 100%. However, concerning the decoding



(a) Decoding rate

(b) Average decoding time

(c) Energy consumption

Figure 4: Performance results when single path and multipath network coding are confronting to selective forwarding attackers and to flooding attackers.

rate, the portion of successfully decoded packets, presented in Figure 3(a), we notice a clear degradation of this rate: passing, for a 200 nodes network, from more than 80% when no attackers are present in the network to less than 40% when 30% of attackers are present. This means that many source nodes will continue to send encoded packets until they die. Thus, the success of the attacker is clear in this case.

Comparing those values with the ones of the previous section where no degradation is observed when multipath network coding is confronting to selective forwarder attackers, we deduce that flooding attack affects the network coding process in terms of decoding rate.

When looking at average decoding time shown in Figure 3(b), this value remains about the same for all cases: equal to 24 seconds. This result is exactly the same as the ones given in the previous section. This is due to the fact that the decoding time only concerns messages that have been successfully decoded, that is, messages that have been correctly sent and where the ACK has been correctly received by the source node. In other words, this value only concerns messages that have not encountered any attacker. So, this value remains normally the same.

When ACK is hijacked by flooding attackers, even after the successful decoding process at the sink, the source node continues sending encoded packets, and others receive and forward these packets. *Energy consumption* measured in this section is the sum of these extra consumptions. Scenario with a 50-node network fronting 30% attackers is used as the norm value, and the others are normalized according to this norm, as shown in Figure 3(c). The results concerning the

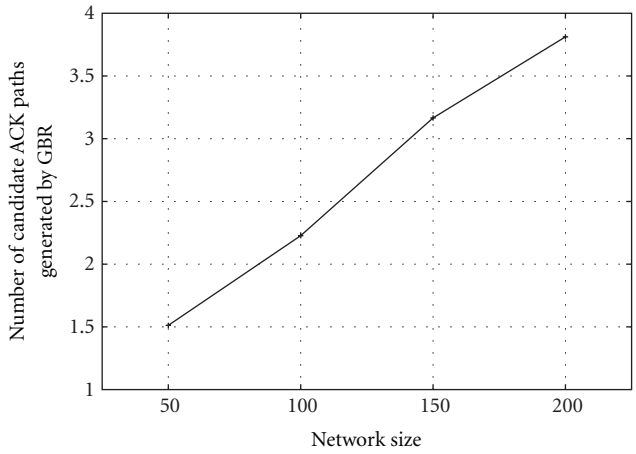


Figure 5: Evolution of the average number of ACK paths generated by GBR as a function of the network size.

case of 0% attackers do not appear on Figure 3(c) because they are all too close to 0. So, the most expensive case is the 50-node network with 30% of attackers. It means that the energy wasted in the network due to the absence of ACK back is huge. The results for 30% of attackers and other network sizes proportionally imply less degradation because the diversity of possible ACK paths is more important leading to waste less energy due to source nodes that continue to send packets. In the same way, with fewer attackers present in the network (10% and 20%), the energy waste is less important because more ACK messages reach their destinations.

In conclusion, and as observed in our simulations, the flooding attack is clearly an efficient attack against the network coding process because network coding does not provide intrinsic mechanisms to prevent attacks against the ACK flow. This is why we propose such a mechanism in our paper.

4.2.3. Analysis for Combining Attacks. A critical question for network coding security is to combine all the solutions dedicated to a given attack and to evaluate the performances in the presence of all kind of adversaries. Our results include both selective forwarding attacks on the data flow and flooding attacks. Those results are presented in Figure 4: the percentage x% of compromised nodes corresponds to x% of flooding nodes on the ACK flow and of x% of selective forwarding nodes in the data flow.

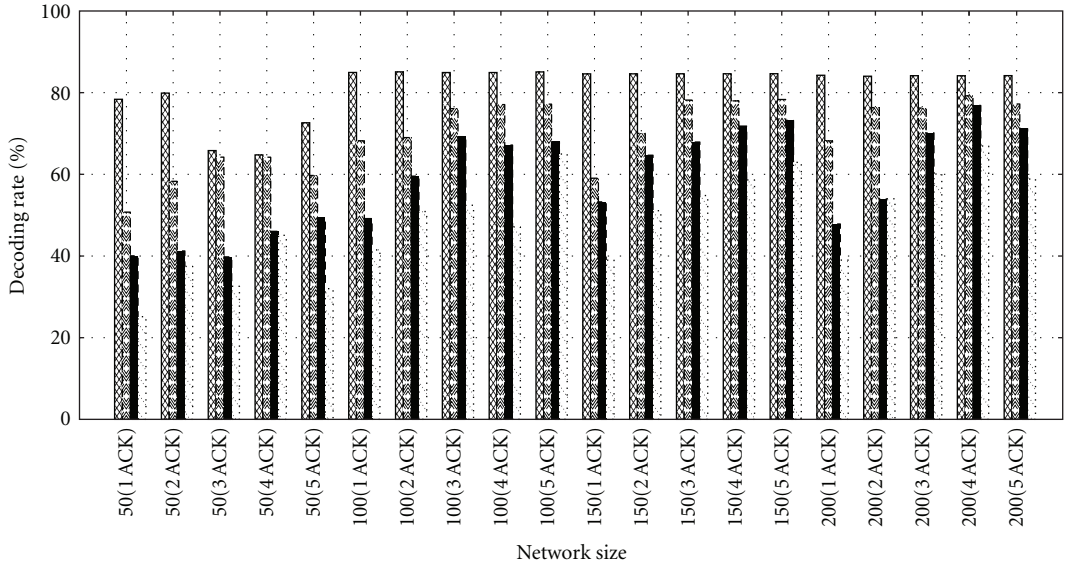
As in Section 4.2.1, we present the results for the dummy example "network coding with single path and single ACK back path" for comparison purpose. In Figure 4(a), we observe that the decoding rate, with respect to the number of attackers, always degrades for all the network sizes and all the strategies. The degradation for the single path strategy comes essentially from the selective forwarding attackers even if the presence of flooding attackers increases the degradation (when compared with Figure 2(a)). Figure 4(a) exactly reflects the severe impact of the flooding attack on the network. The influence is so significant that it overwhelms all the advantages brought by multipath data forwarding. As we can see in Figure 2(a), the multipath data forwarding method is applied against selective forwarding attacks, so the performance results of 10%, 20%, and 30% attackers are close to the ones with 0% attackers. We assume that the multipath method almost compensates all the negative influences from selective forwarding attacks. And we release two attacks in Figure 4(a) scenario: the selective forwarding attack and the flooding attack. The selective forwarding attacks impose great performance degradation onto the data flow from the source to the sink, but the multipath data forwarding method helps the network to overcome the performance loss, according to Figure 2(a). The flooding attacks impose performance degradation on the ACK flow. It is obvious that the performances brought down by flooding attacks are dominant in this scenario. That means that the advantages of multipath data forwarding strategy are totally overwhelmed by the flooding attacks.

Concerning the average decoding time presented in Figure 4(b), surprisingly, the times for the single path strategies are better than the ones in Figure 2(b) for all network sizes. This is due to the fact that less packets arrive at the sink, and less ACKs are returned to the source nodes. So, messages that are correctly decoded are less numerous and require less time to be correctly decoded. As already observed in Figures 2(b) and 4(b), in the case of multipath strategies, there is no significant degradation of decoding time for the same reasons as the ones exposed in Sections 4.2.1 and 4.2.2 This essentially comes from the fact that the decoding time only concerns packets well received at the sink and well acknowledged at the source nodes.

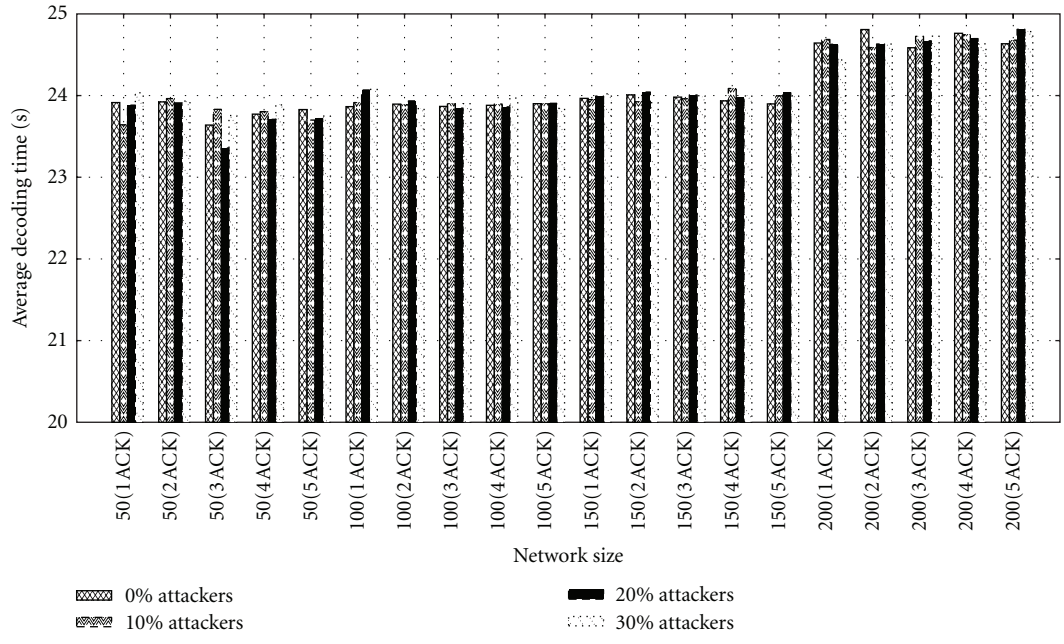
In Figure 4(c), we observe the energy consumption results where the norm value is for 0% attackers, a network with 200 nodes and multipath network coding as in the case of Figure 2(c). Anyway, Figures 4(c) and 2(c) have the same main characteristics. However, the energy consumption for multipath strategies is worst in all cases when both attacks are combined due to the flooding attacks effect. For single path strategies, surprisingly the energy consumption is about the same proportion as in Figure 2(c) (the values are also about to be the same). These surprising results come from the combining effects of flooding attacks that discard the acknowledgements and make the source nodes to continue to send packets and effects of selective forwarding attacks that discard a part of those exceeded packets sent. More generally, the energy consumption of the single path strategies is small when compared with all multipath strategies.

When combining both attacks, clearly the simulation results also combine the worst performances of each attack so the decoding rate for single path strategies has about the same behavior (in worst) as in the case of selective forwarding attackers whereas the decoding rate and the decoding time for multipath strategies have about the same behavior (in worst) as in the case of flooding attackers.

4.3. Attacks under Study with Multipath ACK Network Coding Strategy. In this part, we sum up our simulation results and the corresponding analysis when our multipath ACK network coding strategy is used in the network. All simulations are performed using the same experimental



(a) Decoding rates



(b) Decoding times

Figure 6: Decoding rates and decoding times when network coding is confronting with flooding attackers.

conditions and the same metrics than the ones described in Section 3. We first study the evolution of the number of paths available in the network to send back the ACK, as this parameter is critical in our problem.

4.3.1. Average Number of Paths from Random GBR. As explained in Section 3.3, the successful transmission of the ACK depends essentially on the capabilities and the opportunity to send back the ACK packets to the source node. Intuitively, it should be accomplished by using as many paths as possible. In fact, the ideal number of ACK paths is not "the bigger the better," as this will be bounded by the routing protocol parameters. Our simulation results show, in Figure 5, that, for GBR and for the network sizes considered here, the average number of established ACK paths is always less than 4.

This average value becomes constant as the network grows as shown by other simulations not drawn in Figure 5 where a clear logarithmic effect appears. So in this case, it however remains better to use 4 or 5 paths to send back ACK packets rather than 2 or 3. Those results can also be seen in Figure 6(a).

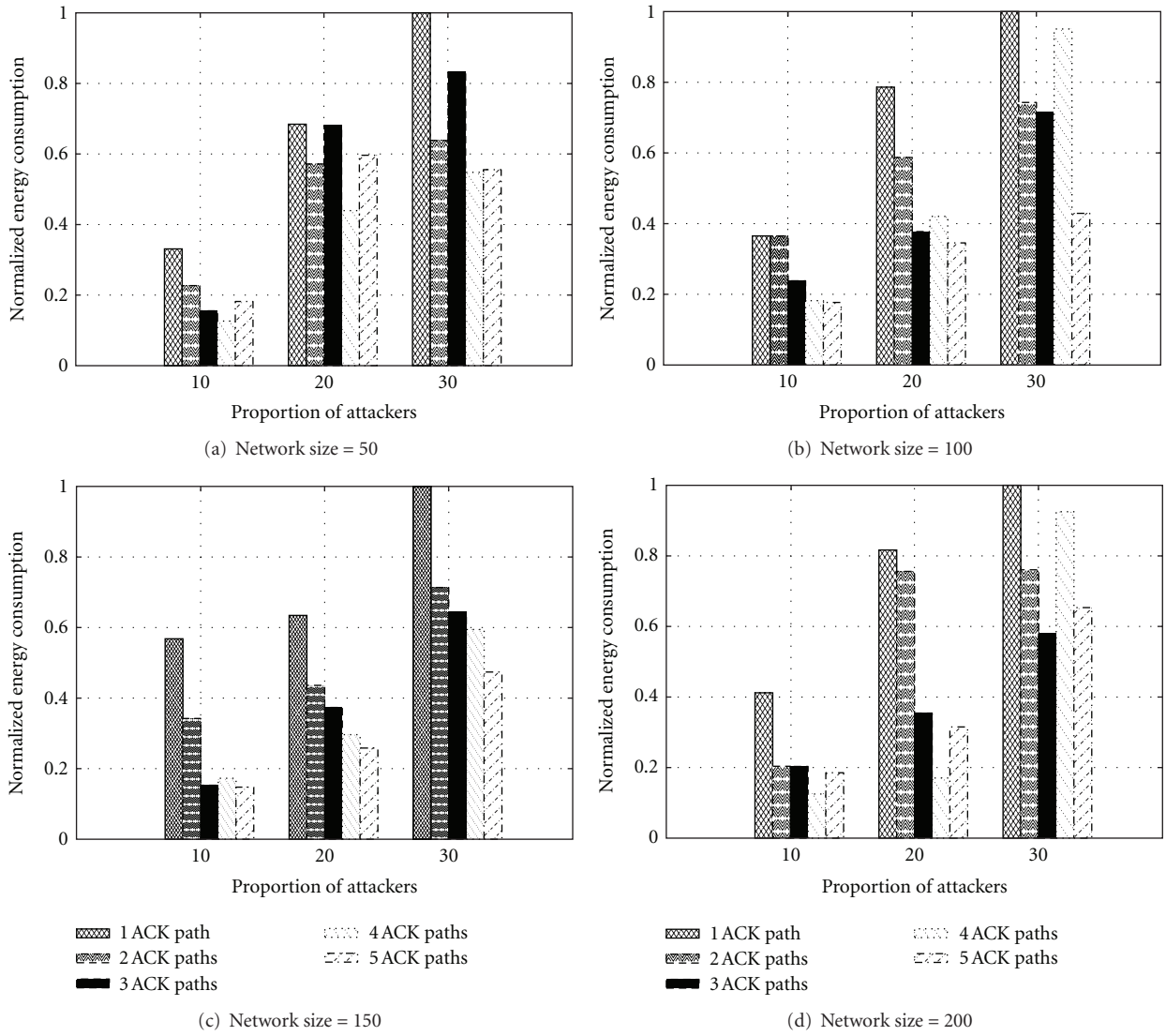
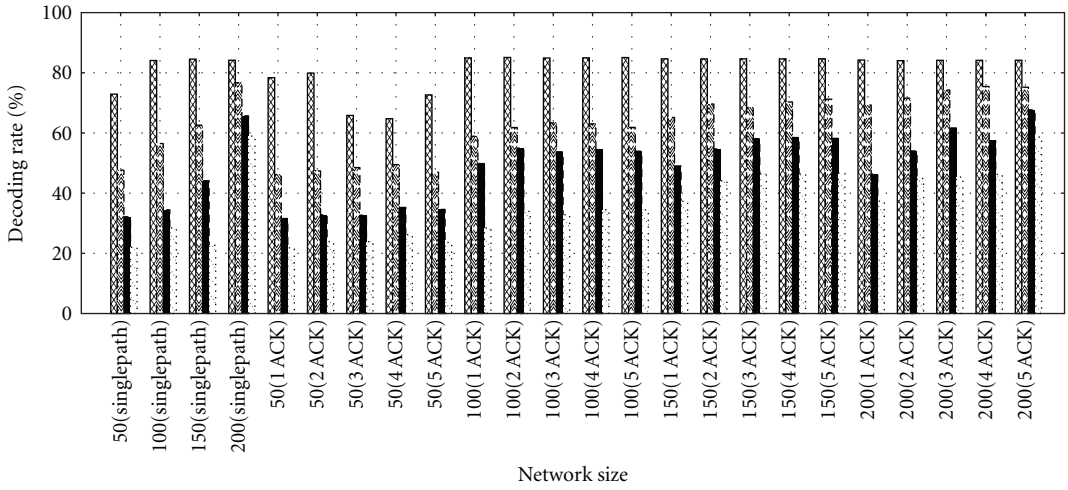


Figure 7: Extra energy consumption *when network coding is confronting with flooding attackers *when ACK is intercepted, the source node still sends encoded packets. The sending and receiving of these packets cause extra energy consumption. *Refers to all figures presesnted in this figure.

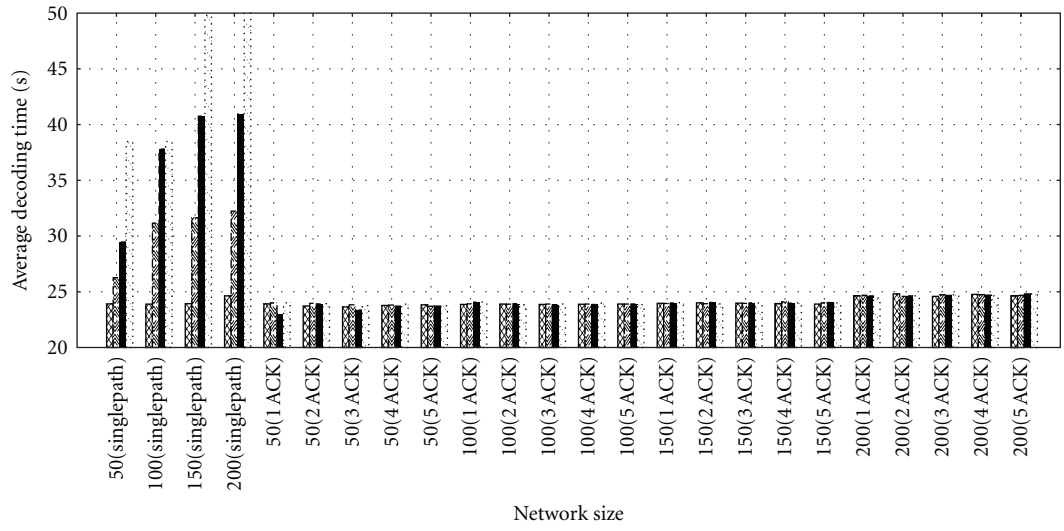
4.3.2. Results Concerning Flooding Attackers. The results in Figure 6(a) also show that even if our multipath ACK strategy is not so efficient in small networks, it becomes interesting (increasing the rate of successfully decoded packets) as soon as the network is sufficiently large, that is, dense. For example, for 5 ACK and 200 nodes, the decoding rate is equal to 79% when 10% of attackers are present into the network and decreases to 62% when 30% of nodes are malicious which gives better rates and better digressions than with only one ACK path.

The results are more significant in larger networks because smaller networks have fewer paths (as shown in Figure 5) available for the sink to send back ACK packets. Therefore, multipath ACK strategy is much more suitable for networks with larger size, that is, dense networks. On the other side, we should notice that using more ACK paths does not always help improving the performances, as we already explained in Section 4.3.1 and as shown in Figure 6(a). We can see in every figure that the performance gap among scenarios with one ACK path, two ACK paths, and three ACK paths is larger than others; that is, the number of packets successfully decoded in scenarios with two ACK paths and three ACK paths is 28% and 47% more than for the scenario with one ACK path approximately, while scenarios with four and five ACK paths have improvements of 45% and 53%, respectively. Employing many ACK paths is interesting only when numerous paths are available which is not always the case even for dense networks as shown in Section 4.3.1.

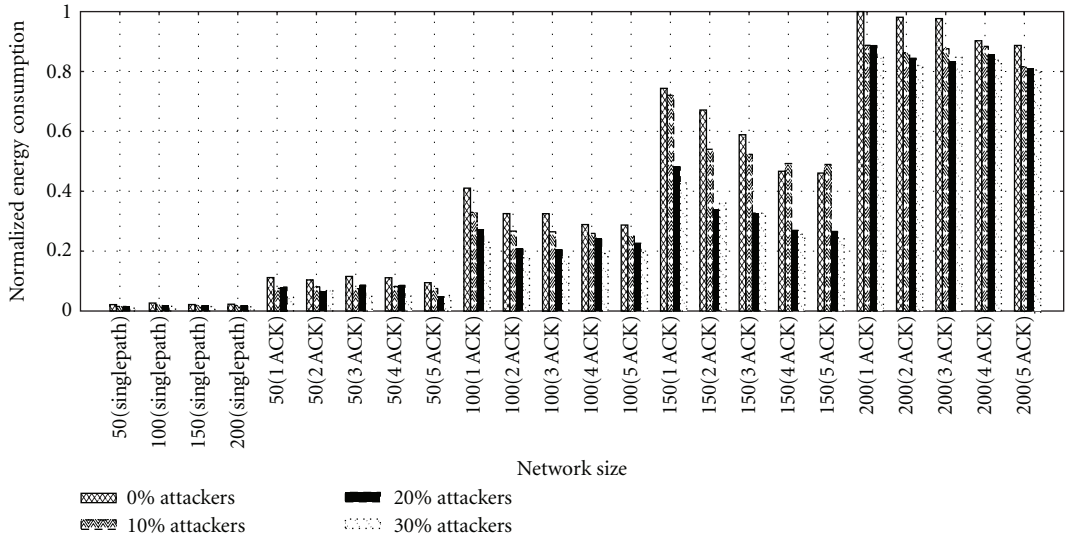
The worst case possible scenario to occur is when attackers are inserted on all different paths between the sink and the source node. This can happen when we deal with very clever attackers (this is not the case here where the attackers are randomly picked among all the nodes). Those particular attackers have an excellent analysis of the network traffic. However, our proposal stays efficient because the routes are at each time taken as random (due to the design of



(a) Decoding rates



(b) Decoding times



(c) Energy consumptions

Figure 8: Combined flooding and selective forwarding attackers: comparison of the number of decoding rates, decoding times, and energy consumptions in case of 1 ACK path, 2 ACK paths, 3 ACK paths, 4 ACK paths, and 5 ACK paths.

the random GBR protocol described in Section 3.2.2) where an attacker could not know all the random routes used by the encoded packets from a source to the destination as explained in [25].

In Figure 6(b), we present the results concerning the average decoding time. This time stays about the same in all cases even if the cases with 4 and 5 ACK paths seem to give the best decoding time. In all cases, the values observed stay around 24 seconds and do not seem to generate a big degradation of performances. However, the decoding time for a 200 nodes network is a little bit greater due to the size of the network. We have implemented the same scenario with bigger network sizes and have noted that the decoding time growth steepens from network size 200 and above.

In Figure 7, we present the results concerning energy consumption gain. When ACK flow is hijacked by flooding attackers, even after the successful decoding process at the sink, the source node continues sending encoded packets, and others receive and forward these packets. Figure 7 highlights those extra consumptions. The norm value of our figures equal to 1 (which is the most energy consuming one) is for each network size, the energy consumed when 30% of attackers are present in the network and when only one ACK path is used. This corresponds with the case where the most of energy is dissipated in the network due to the source nodes that continue to send encoded packets as already mentioned.

It is interesting to notice here that even if multiplying the ACK paths consumes energy, this consumption is marginal when compared to the flooding provoked by the disappearance of the ACK packets. So, in terms of energy consumption, our multipath ACK solution is really efficient when compared with the single ACK path (e.g., when 30% attackers are present in the network, 5 ACK paths solution only consumes half of the energy of the 1 ACK path solution). Indeed, with only one ACK path, the probability that the ACK packets are thwarted by the attackers is high; thus the source and intermediate nodes continue sending and forwarding packets, which is exactly the cause of unnecessary energy waste.

4.3.3. Results When Combining Selective Forwarding Attackers and Flooding Attackers. As already mentioned in Section 4, it is really important when a security solution is proposed to combine possible attacks and to evaluate the performances in the presence of all kinds of adversaries. The results presented in this section include both selective forwarding attacks on the data flows and flooding attacks. Those results are presented in Figure 8: as previously, the percentage x% of compromised nodes corresponds to x% of flooding nodes on the ACK flow and of x% of selective forwarding nodes in the data flow.

When we bring two attacks into the network, as shown in Figure 8, the performances of single path scenarios do not vary from results of Figure 4. When we switch on the multipath option, *average decoding time* keeps up with the good results of Figures 2(b) and 6(b), but *decoding rate* has been drawn back by flooding attackers. Because the attacks take effects on different flows, analysis on separated attacks is much more effective and clearer to unveil the advantages brought about by multipath method.

All the results presented in Figure 8 are always worse than those presented in Figures 6 and 7. This comes from the fact that selective forwarding attackers on the data flows introduce a delay for a correct decoding of the packets, and as the simulations made here hold the same time in all cases, the portion of correctly decoded packets is worse. Those effects are less significant for larger networks because the delay induced by selective forwarding is less important. Note also that for dense networks our multipath ACK strategy against flooding attackers stays efficient.

This combined attacks scenario also highlights the fact that our strategy is more efficient in cases of dense networks as shown in Figure 8(a). Moreover and as expected, the impact of selective forwarding is not efficient due to the intrinsic nature of the network coding.

5. Conclusion

We have considered selective forwarding attacks against both data flows and ACK flows in network coding applications. The impact of those attacks has been studied when the adversary randomly compromised the nodes.

Due to its intrinsic multipath nature, network coding is resilient against selective forwarding attackers even if this kind of attacks introduces a little delay in the network. This is the first step we want to demonstrate in this paper. We do not develop here a dedicated mechanism to identify and avoid attackers in the network because we only want simple mechanisms that could be added to the routing layer complementary with network coding to bypass the attackers at a reasonable cost.

Against flooding attacks, our countermeasure is based on multipath ACK, and it is a randomized variant of GBR that allows to build several backward paths we use for the ACK sent. Our simulation results have shown that our solution is efficient as soon as we have a sufficient number of distinct backward paths. Such condition is easily obtained in dense networks.

The choice of the routing protocol is critical, and the key feature is the capacity to generate randomly many paths: greater are the paths of the ACK, higher is the probability to thwart flooding attacks.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (no. 61103040).

References

[1] L. Lima, J. Barros, and M. Médard, "Random linear network coding: a free cypher?" in *Proceedings of the IEEE International Symposium on Information Theory (ISIT '07)*, pp. 176–180, Nice, France, July 2007.

[2] N. Cai and R. W. Yeung, "Secure network coding," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT '02)*, p. 323, July 2002.

[3] S. Y. El Rouayheb and E. Soljanin, "On Wiretap networks II," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT '07)*, pp. 551–555, Nice, France, June 2007.

[4] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan, "An efficient scheme for securing XOR network coding against pollution attacks," in *Proceedings of the 28th IEEE Communications Society Conference on Computer Communications (IEEE INFOCOM '09)*, pp. 406–414, Rio de Janeiro, Brazil, April 2009.

[5] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan, "An efficient signature-based scheme for securing network coding against pollution attacks," in *Proceedings of the 27th IEEE Communications Society Conference on Computer Communications (IEEE INFOCOM '08)*, pp. 2083–2091, Phoenix, Ariz, USA, April 2008.

[6] J. Dong, R. Curtmola, and C. Nita-Rotaru, "Practical defenses against pollution attacks in intra-flow network coding for wireless mesh networks," in *Proceedings of the 2nd ACM Conference on Wireless Network Security (WiSec '09)*, pp. 111–122, ACM, March 2009.

[7] A. Apavatjrut, W. Znaidi, A. Fraboulet, C. Goursaud, C. Lauradoux, and M. Minier, "Energy friendly integrity for network coding in wireless sensor networks," in *Proceedings of the 4th International Conference on Network and System Security (NSS '10)*, pp. 223–230, IEEE, September 2010.

[8] D. Charles, K. Jain, and K. Lauter, "Signatures for network coding," *International Journal in Information and Coding Theory*, vol. 1, no. 1, pp. 3–14, 2009.

[9] D. Boneh, D. Freeman, J. Katz, and B. Waters, "Signing a linear subspace: signature schemes for network coding," in *Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography (PKC '09)*, vol. 5443 of *Lecture Notes in Computer Science*, pp. 68–87, Springer, Irvine, Calif, USA, 2009.

[10] S. Agrawal and D. Boneh, "Homomorphic MACs: MAC-based integrity for network coding," in *Proceedings of the 7th International Conference on Applied Cryptography and Network Security (ACNS '09)*, vol. 5536 of *Lecture Notes in Computer Science*, pp. 292–305, Paris, France, 2009.

[11] J. Dong, R. Curtmola, and C. Nita-Rotaru, "Secure network coding for wireless mesh networks: threats, challenges, and directions," *Computer Communications*, vol. 32, no. 17, pp. 1790–1801, 2009.

[12] C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram, and D. Zamboni, "Analysis of a denial of service attack on TCP," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 208–223, IEEE Computer Society, Oakland, Calif, USA, May 1997.

[13] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[14] T. Ho and D. Lun, *Network Coding: an Introduction*, Cambridge University Press, 2008.

[15] R. W. Yeung, S.-Y. R. Li, N. Cai, and Z. Zhang, *Network Coding Theory*, NOW Publishers, 2005.

[16] J. Cannons, R. Dougherty, C. Freiling, and K. Zeger, "Network routing capacity," *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 777–788, 2006.

[17] T. Ho, M. Médard, R. Koetter et al., "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[18] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures," *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 293–315, 2003.

[19] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM '00)*, pp. 255–265, ACM, Boston, Mass, USA, August 2000.

[20] H. Krawczyk, "LFSR-based hashing and authentication," in *Proceedings of the Annual International Cryptology Conference (CRYPTO '94)*, vol. 839 of *Lecture Notes in Computer Science*, pp. 129–139, Springer, Santa Barbara, Calif, USA, 1994.

[21] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," 1997, rFC 2104.

[22] J. Black and P. Rogaway, "CBC MACs for arbitrary-length messages: the three-key constructions," *Journal of Cryptology*, vol. 18, no. 2, pp. 111–131, 2005.

[23] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS '02)*, pp. 41–47, ACM, Washington, DC, USA, November 2002.

[24] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the air: practical wireless network coding," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 497–510, 2008.

[25] O. Erdene-Ochir, M. Minier, F. Valois, and A. Kountouris, "Toward resilient routing in wireless sensor networks: gradient-based routing in focus," in *Proceedings of the 4th International Conference on Sensor Technologies and Applications (SENSORCOMM '10)*, pp. 478–483, Venice, Italy, July 2010.

[26] J. Faruque and A. Helmy, "Gradient-based routing in sensor networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 7, no. 4, pp. 50–52, 2003.

[27] A. Fraboulet, G. Chelius, and E. Fleury, "Worldsens: development and prototyping tools for application specific wireless sensors networks," in *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN '07)*, pp. 176–185, ACM, April 2007.

[28] M. Médard and R. Koetter, "Beyond routing: an algebraic approach to network coding," in *Proceedings of the IEEE Communications Society Conference on Computer Communications (IEEE INFOCOM '02)*, pp. 122–130, IEEE, New York, NY, USA, June 2002.