9 Network-coding security

Many of the applications of classical coding techniques can be found at the physical layer of contemporary communication systems. However, coding ideas have recently found their way into networking research, most strikingly in the form of algebraic codes for networks. The existing body of work on network coding ranges from determinations of the fundamental limits of communication networks to the development of efficient, robust, and secure network-coding protocols. This chapter provides an overview of the field of network coding with particular emphasis on how the unique characteristics of network codes can be exploited to achieve high levels of security with manageable complexity. We survey network-coding vulnerabilities and attacks, and compare them with those of state-of-the-art routing algorithms. Some emphasis will be placed on active attacks, which can lead to severe degradation of network-coded information flows. Then, we show how to leverage the intrinsic properties of network coding for information security and secret-key distribution, in particular how to exploit the fact that nodes observe algebraic combinations of packets instead of the data packets themselves. Although the prevalent design methodology for network protocols views security as something of an add-on to be included after the main communication tasks have been addressed, we shall contend that the special characteristics of network coding warrant a more comprehensive approach, namely one that gives equal importance to security concerns. The commonalities with code constructions for physical-layer security will be highlighted and further investigated.

9.1 Fundamentals of network coding

The main concept behind network coding is that data throughput and network robustness can be considerably improved by allowing the intermediate nodes in a network to mix different data flows through algebraic combinations of multiple datagrams. This key idea, which clearly breaks with the ruling store-and-forward paradigm of current message-routing solutions, is illustrated in Figure 9.1. To exchange messages a and b, nodes A and B must route their packets through node S. Clearly, the traditional scheme shown on top would require four transmissions. However, if S is allowed to perform network coding with simple exclusive-or (XOR) operations, as illustrated in the lower diagram, $a \oplus b$ can be sent in a single broadcast transmission (instead of one transmission with b followed by another one with a). By combining the received data with the stored

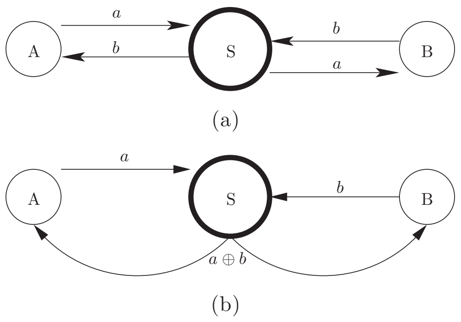

Figure 9.1 A typical wireless network-coding example.

message, A, which possesses a, can recover b, and B can recover a using b. Consequently, in this scenario, network coding saves one transmission (thus saving energy) and one time slot (thus reducing the delay). More sophisticated network-coding protocols view packets as a collection of symbols from a particular finite field and forward linear combinations of these symbols across the network, thus leveraging basic features of linear codes such as erasure-correction capability and well-understood encoding and decoding algorithms.

The resulting techniques seem particularly useful for highly volatile networks, such as mobile ad-hoc networks, sensor networks, and peer-to-peer communications, where stringent constraints due to power restrictions, limited computation capabilities, or unpredictable user dynamics can be countered by broadcasting encoded packets to multiple nodes simultaneously until the destination has enough degrees of freedom to decode and recover the original data, as illustrated by the example in Figure 9.1.

Using information-theoretic reasoning, it is possible to prove that the multicast capacity of a network is equal to the minimum of the maximum flows between the source and any of the individual destinations. Most importantly, routing alone is in general not sufficient to achieve this fundamental limit – intermediate nodes are required to mix the data units they receive from their neighbors using non-trivial coding operations.

The intuition behind this result is well illustrated by the butterfly network shown in Figure 9.2, where each edge is assumed to have unitary capacity. If node 1 wishes to send a multicast flow to sinks 6 and 7 at the max-flow min-cut bound, which in this case is 2, the only way to overcome the bottleneck between nodes 4 and 5 is for node 4 to combine the incoming symbols through an XOR operation. Sinks 6 and 7 can then use the symbols they receive directly from nodes 2 and 3, respectively, in order to reverse this XOR operation and reconstruct the desired multicast flow.

It has also been shown that linear codes are sufficient to achieve the multicast capacity, yielding the algebraic framework for network coding that has since fueled a strong surge in network-coding research.

Although establishing the information-theoretic limits of communication networks with multiple unicast or multicast sessions still seems a distant goal, there is reasonable evidence that network coding allows a trade-off of communication versus computational costs. Furthermore, network coding brings noticeable benefits in terms of throughput,

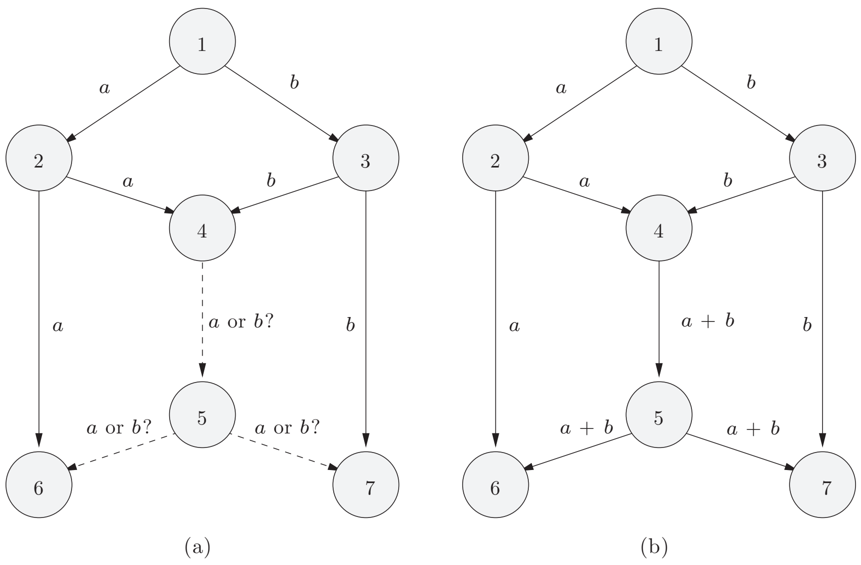

Figure 9.2 Canonical network-coding example.

reliability, and fault tolerance in a variety of relevant networking scenarios beyond the single multicast case, including wireless broadcast and peer-to-peer systems. In particular, random linear network coding (RLNC) provides a fully distributed methodology for network coding, whereby each node in the network selects independently and randomly a set of coefficients and uses them to form linear combinations of the data symbols (or packets) it receives. These linear combinations are then sent over the outgoing links together with the coding coefficients until the receivers are able to decode the original data using Gaussian elimination.

9.2 Network-coding basics

Network coding with XOR operations

In the simplest form of network coding all operations are carried out in the binary field. In more practical terms this implies that packets are mixed (and unmixed) by means of straightforward XOR operations. The basic principle and the underlying advantages of this low-complexity approach are illustrated in Figure 9.1. Suppose that, in a wireless network, nodes A and B must route their packets through node S, in order to exchange messages a and b. S sends $a \oplus b$ in a single broadcast transmission (see Figure 9.1(b)). By combining the received data with the stored message, A, which possesses packet a, can recover b, and B can recover a using b, in both cases with one XOR operation.

Naturally, this idea can be easily extended to scenarios with more than three terminals. A node that has several packets in its buffer can choose judiciously how to combine the packets for the next broadcast transmission. This decision may depend on several factors, including the delay or bandwidth requirements, the available knowledge about the communication channels, the network topology, and feedback about the buffer states of neighboring nodes.

Linear network coding

Moving from the binary field to larger field sizes allows the nodes in the network to perform more sophisticated operations on the incoming packets. In the standard network-coding formulation the network is represented as an acyclic directed graph $G = (V, E)$, where the vertices V represent the terminals, the edges E correspond to the available communication links, and each data unit is represented by a symbol from a predefined finite field F with $q = 2^m$ for some integer m. Suppose now that node v has in its buffer a collection of symbols denoted $X_1, \ldots, X_K$ and receives symbols $Y_1, \ldots, Y_L$ from its incoming edges. In linear network coding, the next data symbol Y_e to be transmitted to a different node u on the outgoing edge $e = (v, u)$ is a linear combination of the available symbols and can be computed according to

$$Y_e = \sum_{i=1}^{K} \alpha_i X_i + \sum_{j=1}^{L} \beta_j Y_j,$$

where $\alpha_i, \beta_j \in F$ are the linear coefficients used to encode the data. Since all of the operations are modulo operations over a finite field, mixing packets through linear network coding does not increase the packet size. To recover the transmitted symbols, the destination node waits until it has enough independent linear combinations (or degrees of freedom) and then performs Gaussian elimination to solve the resulting linear system.

An important property of linear network coding is that for a fixed network topology it is possible to find the optimal network code for a multicast session by means of a polynomial-time algorithm that establishes the exact operations that each node has to carry out. This is in sharp contrast with the multicast routing problem (without coding), which is well known to be NP-hard. An important consequence of the algorithm is that it allows us to bound the alphabet size for the network code or, equivalently, the required bandwidth per link.

Random network coding

If the network is large or highly dynamic, computing the network code in a centralized fashion may quickly become unfeasible. Fortunately, it is possible to perform network coding in a fully distributed way, by allowing the nodes to choose their linear coefficients independently and uniformly at random over all elements of the finite field F. This approach achieves the multicast capacity of the network, provided that the field size is sufficiently large. From a more practical point of view, random linear network coding can be implemented effectively in packet-oriented networks by (1) segmenting the payload into data symbols that can be viewed as elements of F, (2) computing the required linear combinations on a symbol-by-symbol basis using random coefficients, (3) placing the resulting symbols on the payload of the outgoing packet, and (4) updating the header with the random coefficients required for decoding.

Table 9.1 Network coding in the layered architecture

Layer	Network-coding (NC) examples
Application	Random NC in overlay networks, distributed storage, content distribution, peer-to-peer applications
Transport	Flow control with special ACKs, reliable NC multicast with feedback, combined NC and back-pressure algorithms
Network	Deterministic NC, random NC with directed diffusion
Link	Opportunistic NC, wireless multicast with feedback
Physical	Analog NC

Upon receiving a sufficient number of packets, each destination node can obtain the coding matrix from the headers of the incoming packets and recover the original data symbols. Notice that, due to the mixing operations, each packet loses its individuality in the sense that the information of a single original packet is spread over multiple instances. Thus, as long as there are enough degrees of freedom to decode the original symbols, the destination is not particularly concerned with obtaining a specific packet.

9.3 System aspects of network coding

Having covered some of the basic principles of network coding, we are now ready to discuss their application in the layered architecture of the dominant communication networks. With this goal in mind, we shall pursue a top-down approach, from the application down to the physical layer, as summarized in Table 9.1.

The application layer is particularly well suited as a first arena for the development of network-coding protocols. Implementing the required software at the outer edge of the network means that neither the routing and medium-access protocols nor the routers need to be modified or replaced. The routers can be kept exactly as they are. Naturally, the simplified protocol design at the application layer comes at the price of discarding any throughput and robustness benefits that network coding may bring when applied at the lower layers of the protocol stack.

On the other hand, the application layer allows us to define overlay network topologies that are particularly useful in conjunction with random network-coding protocols. One of the real-life applications that exploits these synergies is the Microsoft Secure Content Distribution software, also known as Avalanche. Here, each node wishing to download a large file contacts a number of nodes through a peer-to-peer overlay network and collects linear combinations of file fragments until it is able to recover the entire file. The results indicate that significant reductions in download time and increased robustness against sudden changes in the network are achieved.

Since the random-coding coefficients have to be placed in the header of the packets, random network-coding protocols introduce some extra overhead, which can vary a lot depending on the packet size, the field size, and the number of packets that are combined (also called a *generation*). This is illustrated in Table 9.2. For internet applications,

Table 9.2 Network coding overhead per packet

IP packet size (Bytes)	Generation size	Overhead (%) $q = 2^8$	Overhead (%) $q = 2^{16}$
1500	20	1.3	2.7
	50	3.3	6.7
	100	6.7	13.3
	200	13.3	26.7
5000	20	0.4	0.8
	50	1.0	2.0
	100	2.0	4.0
	200	4.0	8.0
8192	20	0.2	0.5
	50	0.6	1.2
	100	1.2	2.4
	200	2.4	4.8

where the IP packet size can be fairly large, the overhead is often negligible. However, in wireless communication networks the size of the generation and resulting overhead must be further restricted because packets are generally smaller. It is also worth mentioning that in error-prone channels the header requires extra protection by means of error-correction codes, which further increase the overhead incurred.

In distributed-storage applications with unreliable storage elements, random network coding increases data persistence by spreading linear combinations in different locations of the network. As long as there exist enough degrees of freedom stored in the network, the data can be decoded even in highly volatile environments.

The facts that the receiver has to wait until it receives enough packets to decode the data and that the decoding algorithm itself has non-negligible complexity ($O(n^3)$ for generation size n) have aroused some skepticism as to whether network coding can be used effectively in real-time applications, e.g. video streaming in peer-to-peer networks. Nevertheless, recent work on combining network coding with a randomized push algorithm shows promising results.

At the transport layer, the Transmission Control Protocol (TCP) typically fulfills two different tasks: (a) it guarantees reliable and in-order delivery of the transmitted data from one end of the network to its final destination by means of retransmissions, and (b) it implements flow and congestion control by monitoring the network state, as well as the buffer levels at the receiver. To fulfill its purpose, TCP relies on the acknowledgments sent by the receiver to decide which packets must be retransmitted and to estimate vital metrics pertaining to the round-trip delay and the congestion level of the network.

In the case of network-coding protocols, the transmitted packets are linear combinations of different packets within one generation, and therefore the acknowledgments sent by the receiving end may provide a very different kind of feedback. More specifically, instead of acknowledging specific packets, each destination node can send back requests for degrees of freedom that increase the dimension of its vector space and allow faster decoding. Upon receiving acknowledgments from the different receivers, the source node minimizes the delay by sending the most innovative linear combination, i.e. the one that is useful to most destination nodes. Without network coding, end-to-end reliability for multicast sessions with delay constraints is generally perceived as a very difficult problem.

Random network coding seems to perform well in combination with back-pressure techniques, which analyze the differences in queue size between the two ends of a link in order to implement distributed flow control and thus maintain the stability of the network.

The single path or route that lies at the core of common routing algorithms is arguably less clear when network coding is involved, because network coding seems most effective when there are multiple paths carrying information from the source to the destination. One way to proceed is to combine network coding with directed diffusion techniques, whereby nodes spread messages of interest in order to route the right data to the right nodes. Even if one opts to use a standard routing algorithm to forward the data, the topology-discovery phase, during which nodes broadcast link-state advertisements to all other nodes, is likely to benefit from the throughput gains of network-coding-based flooding protocols.

At the link layer, opportunistic network coding emerges as a promising technology, which has been implemented successfully in a real wireless mesh network. The main idea is that nodes in a wireless network can learn a lot about the flow of information to neighboring nodes simply by keeping their radios in promiscuous mode during idle times. On the basis of this acquired knowledge and the buffer state information sent by neighboring nodes, it is possible to solve a local multicast problem in the one-hop neighborhood of any given node. The XOR combination of queued packets that is useful to most neighbors can then be broadcast, thus minimizing the number of transmissions and saving valuable bandwidth and power. To reap the largest possible benefits from this approach, the medium-access protocols and scheduling algorithms at the link layer must be redesigned to allow opportunistic transmission.

Network-coding ideas have recently found their way to the physical layer by means of a simple communication scheme whereby two nodes send their signals simultaneously to a common relay over a wireless channel. The transmitted signals interfere in an additive way and the relay amplifies and broadcasts back the signal it received via its own antenna. This form of *analog* network coding, which had already been implemented using software-defined radios, further reduces the number of transmissions required for the nodes in Figure 9.1 to communicate effectively with each other. There is a strong relationship between the resulting schemes and the relay methods that have been proposed in the area of cooperative transmission among multiple receivers (see also Chapter 8).

9.4 Practical network-coding protocols

Although network coding is a fairly recent technology, there exist already protocol proposals regarding the use of network coding for higher throughput or robustness in

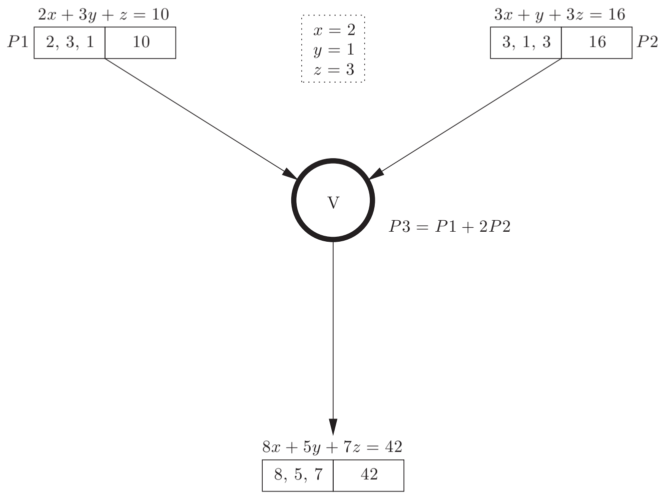

Figure 9.3 An instance of random linear network coding.

various applications and communication networks. From the point of view of network security, it is useful to divide network-coding protocols into two main classes:

(1) *stateless network-coding protocols*, which do not rely on any form of state information to decide when and how to mix different packets in the sender queue;
(2) *state-aware network-coding protocols*, which rely on partial or full network state information (for instance buffer states of neighboring nodes, network topology, or link costs) to compute a network code or determine opportunities to perform network coding in a dynamic fashion.

As will be demonstrated in Section 9.5, the security vulnerabilities of the protocols in the first and second classes are quite different from each other, most notably because the former require state information and node identification to be disseminated in the network and are thus vulnerable to a wide range of impersonation and control-traffic attacks.

Stateless network-coding protocols

As mentioned earlier, random linear network coding (RLNC) is a completely distributed methodology for combining different information flows and therefore leads to stateless protocol design. The basic principle is that each node in the network selects a set of coefficients independently and randomly and then sends linear combinations of the data symbols (or packets) it receives. Figure 9.3 illustrates the linear operations carried out at intermediate node v (using integers for simplicity). The symbols x, y, and z denote the native packets, which convey the information to be obtained at the receivers via Gaussian elimination. $P1$ and $P2$ arrive at intermediate node v in the network through its incoming links. $P3$, which is sent through the only outgoing link of node v, is the result of a random linear combination of $P1$ and $P2$ at node v, with chosen coefficients 1 and 2, respectively.

Recall that the *global encoding vector*, i.e. the matrix of coefficients used to encode the packets, is sent along in the packet header to ensure that the end receivers are capable of decoding the original data. Specifically, it was shown that, if the coefficients are chosen at random from a large enough field, then Gaussian elimination succeeds with overwhelming probability. RLNC can also be used in asynchronous packet networks, and it has been shown that it is capacity-achieving even on lossy packet networks.

A framework for packetized network coding (Practical Network Coding, PNC) leverages RLNC's resilience against disruptions such as packet loss, congestion, and changes of topology, in order to guarantee robust communication over highly dynamic networks with minimal (or no) control information. The framework defines a packet format and a buffering model. The packet format includes in its header the *global encoding vector*, which is the set of linear transformations that the original packet goes through on its path from the source to the destination. The payload of the packets is divided into vectors according to the field size (2^8 or 2^{16}, i.e. each symbol has 8 or 16 bits, respectively). Each of these symbols is then used as a building block for the linear operations performed by the nodes.

The buffering model divides the stream of packets into *generations* of size h, such that packets in the same generation are tagged with a common generation number. Each node sorts the incoming packets in a single buffer according to their generation number. When there is a transmission opportunity at an outgoing edge, the sending node generates a new packet, which contains a random linear combination of all packets in the buffer that belong to the *current* generation. If a packet is *non-innovative*, i.e. if it does not increase the rank of the decoding matrix available at the receiving node, then it is immediately discarded. As soon as the matrix of received packets has full rank, Gaussian elimination is performed at the receivers to recover the original packets.

RLNC seems particularly beneficial in dynamic and unstable networks – that is, networks in which the structure or topology of the network varies within a short time, such as mobile ad-hoc networks and peer-to-peer content-distribution networks. The benefits of RLNC in wireless environments with rare and limited connectivity, either due to mobility or battery scarcity, suggest an algorithm aimed at reducing the overhead of probabilistic routing algorithms with applications in delay-tolerant networks.

The potential impact of RLNC in content-distribution networks can be analyzed as follows. Since each node forwards a random linear combination independently of the information present at other nodes, its operation is completely desynchronized and locally based. Moreover, when collecting a random combination of packets from a randomly chosen node, there is a high probability of obtaining a linearly independent packet in each time slot. Thus, the problem of redundant transmissions, which is typical of traditional flooding approaches, is considerably reduced. In content-distribution applications, there is no need to download one particular fragment. Instead, any linearly independent segment brings innovative information. A possible practical scheme for large files allows nodes to make forwarding decisions solely on the basis of local information. It has been shown that network coding not only improves the expected file-download time, but also improves the overall robustness of the system with respect to frequent changes in the network.

The performance of network coding has been compared with that of traditional coding measures in a distributed-storage setting in which each storage location has only very limited storage space for each file and the objective is that a file-downloader connects to as few storage locations as possible to retrieve a file. It was shown that RLNC performs well without any need for a large amount of additional storage space at a centralized server. There exists also a general graph-theoretic framework for computing lower bounds on the bandwidth required to maintain distributed-storage architectures. It is now known that RLNC achieves these lower bounds.

State-aware network-coding protocols

State-aware network-coding protocols rely on the available state information to optimize the coding operations carried out by each node. The optimization process may target the throughput or the delay, among other performance metrics. Its scope can be *local* or *global*, depending on whether the optimization affects only the operations within the close neighborhood of a node or addresses the end-to-end communication across the entire network. If control traffic is exchanged between neighbors, each node can perform a local optimization step to decide on-the-fly how to mix and transmit the received data packets. One way to implement protocols with global scope is to use the polynomial-time algorithm on a given network graph, which, given a network graph, can be used to determine the optimal network code before starting the communication. By exploiting the broadcast nature of the wireless medium and spreading encoded information in a controlled manner, state-aware protocols promise considerable advantages in terms of throughput, as well as resilience with respect to node failures and packet losses. Efficiency gains come mainly from the fact that nodes make use of every data packet they overhear. In some instances network coding reduces the required amount of control information.

As an example, the COPE protocol inserts a coding layer between the IP and MAC layers for detecting coding opportunities. More specifically, nodes overhear and store packets that are exchanged within their radio range, and then send reception reports to inform their neighbors about which packets they have stored in their buffers. On the basis of these updates, each node computes the optimal XOR mixture of multiple packets in order to reduce the number of transmissions. It has been shown that this approach can lead to strong improvements in terms of throughput and robustness to network dynamics.

9.5 Security vulnerabilities

The aforementioned network-coding protocols expect a well-behaved node to do the following:

- *encode the received packets correctly*, thus contributing to the expected benefits of network coding;
- *forward the encoded packets correctly*, thus enabling the destination nodes to retrieve the intended information;

- *ignore data for which it is not the intended destination*, thus fulfilling basic confidentiality requirements.

In the case of state-aware network-coding protocols, we must add one more rule:

- the node must *participate in the timely dissemination of correct state information*, thus contributing to a sound knowledge base for network-coding decisions.

It follows that an attack on a network-coding-based protocol must result from one or more network nodes breaking one or more of these basic rules. Naturally, the means to achieve a successful attack are, of course, highly dependent on the specific rules of the protocol, and therefore it is reasonable to distinguish between the aforementioned stateless and state-aware classes of network-coding schemes.

If properly applied, stateless network protocols based on RNLC are potentially less subject to some of the typical security issues of traditional routing protocols. First, stateless protocols do not depend on exchange of topology or buffer-state information, which can be faked (e.g. through *link-spoofing* attacks). Second, the impact of *traffic-relay refusal* is reduced, due to the inherent robustness that results from spreading the information by means of network coding. Third, the information retrieval depends solely on the data received and not on the identity of nodes, which ensures some protection against *impersonation* attacks.

In contrast, state-aware network-coding protocols rely on vulnerable control information disseminated among nodes in order to optimize the encodings. On the one hand, this property renders them particularly prone to attacks based on the generation of false control information. On the other hand, control-traffic information can also be used effectively against active attacks, such as the injection of erroneous packets. For protocols with local scope, the negative impact of active attacks is limited to a confined neighborhood, whereas with end-to-end network coding the consequences can be much more devastating. Opportunistic network-coding protocols, which rely on the information overheard by neighboring nodes over the wireless medium, are obviously more amenable to eavesdropping attacks than are their wireline counterparts.

More generally, in comparison with traditional routing, the damage caused by a malicious (Byzantine) node injecting corrupted packets into the information flow is likely to be higher with network coding, irrespective of whether a protocol is stateless or state-aware. Since network coding relies on mixing the content of multiple data packets, a single corrupted packet may very easily corrupt the entire information flow from the sender to the destination at any given time.

9.6 Securing network coding against passive attacks

Having discussed the specific vulnerabilities of network coding, we now turn our attention to the next natural question, namely how to find appropriate mechanisms for securing network-coding protocols. Our main goal here is to show how the specific characteristics of network coding can be leveraged to counter some of the threats posed by

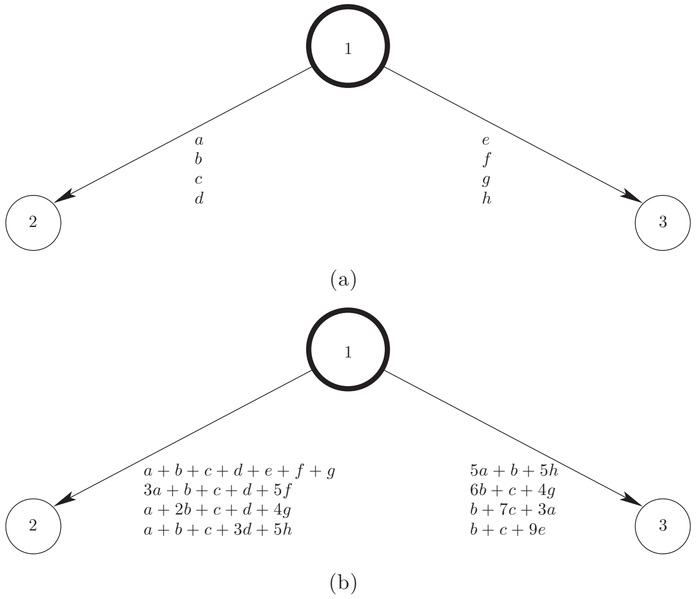

Figure 9.4 Example of algebraic security. The top scheme discloses data to intermediate nodes, whereas the bottom scheme can be deemed algebraically secure.

eavesdroppers and Byzantine attackers. We shall start by presenting countermeasures against passive attacks, with special emphasis on three different scenarios. First, we shall consider nice but curious nodes, which do not break any of the established rules except for not ignoring the data for which they are not the intended receivers. In the second instance, the eavesdropper is able to wiretap a subset of network links. Finally, the third type of attacker is a worst-case eavesdropper who is given full access to all the traffic in the network.

Nice but curious nodes

Consider first a threat model in which the network consists entirely of *nice but curious* nodes, which comply with the communication protocols (in that sense, they are well behaved), but may try to acquire as much information as possible from the data flows that pass through them (in which case, they are potentially ill-intentioned). Under this scenario, stateless protocols that exploit the RLNC scheme described in Section 9.4 possess an intrinsic security feature: depending on the size of the code alphabet and the topology of the network, it is in many instances unlikely that an intermediate node will have enough degrees of freedom to perform Gaussian elimination and gain access to the transmitted data set.

On the basis of this observation, it is possible to characterize the threat level posed by an intermediate node according to an *algebraic security criterion* that takes into account the number of components of the global encoding vector it receives. In the example of Figure 9.4, which uses integers for simplicity, the upper (uncoded) transmission scheme leaves partial data unprotected, whereas in the lower (network-coding) scheme the intermediate nodes 2 and 3 are not able to recover the data symbols.

Wiretappers with access to some links

A different threat model, which is commonly found in the recent literature on secure network coding, assumes that one or more external eavesdroppers (or wiretappers) have access to a subset of the available communication links. The crux of the problem is then the need to find code constructions capable of splitting the data among different links in such a way that reconstruction by the attackers is either very difficult or impossible. Under this assumption, there exist secure linear network codes that achieve perfect information-theoretic secrecy for single-source multicast transmission. The corresponding secure network-coding problem can in fact be cast as a variant of the so-called wiretap channel type-II problem. There, the eavesdropper is able to select and access μ symbols of the n coded symbols transmitted by the legitimate sender. It can be shown that in that case a maximum of $k = n - \mu$ information symbols can be transmitted in perfect secrecy. These results can be generalized to multi-source linear network codes by using the algebraic structure of such codes to derive necessary and sufficient conditions for their security. More specifically, it has been shown that the code constructed maximizes the amount of secure multicast information while minimizing the necessary amount of randomness. Such a coding scheme can achieve a maximum possible rate of $n - \mu$ information-theoretically secure packets, where n is now the number of packets sent from the source to each receiver and μ denotes the number of links that the wiretapper can observe. This can be applied on top of any communication network without requiring any knowledge of the underlying network code and without imposing any coding constraints. The basic idea is to use a "nonlinear" outer code, which is linear over an extension field $\mathbb{F}_{q^m}$, and to exploit the benefits of this extension field. Some contributions propose a different criterion whereby a system is deemed secure if an eavesdropper is unable to get any uncoded or immediately decodable (also called *meaningful*) source data. Other contributions exploit the network topology to ensure that an attacker is unable to get any meaningful information and add a cost function to the secure network-coding problem. The problem then becomes finding a coding scheme that minimizes both the network cost and the probability that the attacker is able to retrieve all the messages of interest.

Since state-aware network-coding protocols with local code optimization (e.g. COPE) expect neighboring nodes to be able to decode all the packets they receive, confidentiality must be ensured by means of end-to-end encryption.

Worst-case eavesdroppers

In this case, the threat model is one in which the attacker has access to all the packets traversing the network but not to the secret keys shared among legitimate communicating parties. Secure Practical Network Coding (abbreviated as SPOC, not SPNC) is a lightweight security scheme for confidentiality in RLNC, which provides a simple yet powerful way to exploit the inherent security of RLNC in order to reduce the number of cryptographic operations required for confidential communication. This is achieved by protecting (or "locking") only the source coefficients required in order to decode the linearly encoded data, while allowing intermediate nodes to run their network-coding operations on substitute "unlocked" coefficients that provably do not compromise the hidden data.

To evaluate the level of security provided by SPOC, one analyzes the mutual information between the encoded data and the two components that can lead to information leakage, namely the matrices of random coefficients and the original data. This analysis, which is independent of any particular cipher used for locking the coefficients, assumes that the encoding matrices are based on variants of RLNC and can be accessed only by the source and sinks. The results, some of which hold even with finite block lengths, prove that information-theoretic security is achievable for any field size without loss in terms of decoding probability. In other words, since correlation attacks based on the encoded data become impossible, protecting the encoding matrix is generally sufficient to ensure the confidentiality of network-coded data.

9.7 Countering Byzantine attacks

Although Byzantine attacks can have a severe impact on the integrity of network-coded information, the specific properties of linear network codes can be used effectively to counteract the impairments caused by traffic-relay refusal or injection of erroneous packets. In particular, RLNC has been shown to be very robust with respect to packet losses induced by node misbehavior. More sophisticated countermeasures, which modify the format of coded packets, can be subdivided into two main categories: (1) end-to-end error correction and (2) misbehavior detection, which can be carried out either packet by packet or in generation-based fashion.

End-to-end error correction

The main advantage of *end-to-end error-correcting codes* is that the burden of applying error-control techniques is left entirely to the source and the destinations, such that intermediate nodes are not required to change their mode of operation. The typical transmission model for end-to-end network coding is well described by a matrix channel $\mathbf{Y} = \mathbf{AX} + \mathbf{Z}$, where $\mathbf{X}$ corresponds to the matrix whose rows are the transmitted packets, $\mathbf{Y}$ is the matrix whose rows are the received packets, $\mathbf{Z}$ denotes the matrix corresponding to the injected error packets after propagation over the network, and $\mathbf{A}$ describes the transfer matrix, which corresponds to the global linear transformation performed on packets as they traverse the network. In terms of performance, error-correction schemes can correct up to the min-cut between the source and the destinations. Rank-metric error-correcting codes in RLNC under this setting appear to work well, including in scenarios in which the channel may supply partial information about erasures and deviations from the sent information flow. Still under the same setting, a probabilistic error model for random network coding provides bounds on capacity and presents a simple coding scheme with polynomial complexity that achieves capacity with an exponentially low probability of failure with respect to both the packet length and the field size. Bounds on the maximum achievable rate in an adversarial setting can be obtained from generalizations of the Hamming and Gilbert–Varshamov bounds.

A somewhat different approach to network error correction consists of robust network codes that have polynomial-time complexity and attain optimal rates in the presence of active attacks. The basic idea is to regard the packets injected by an adversarial node as a second source of information and add enough redundancy to allow the destination to distinguish between relevant and erroneous packets. The capacity achieved depends on the rate at which the attacker can obtain information, as well as on the existence of a shared secret between the source and the sinks.

Misbehavior detection

Generation-based detection schemes generally offer similar advantages to those obtained with network error-correcting codes in that the often computationally expensive task of detecting the modifications introduced by Byzantine attackers is carried out by the destination nodes. The main disadvantage of generation-based detection schemes is that only nodes with enough packets from a generation are able to detect malicious modifications, and thus usage of such detection schemes can result in large end-to-end delays. The underlying assumption is that the attacker cannot see the full rank of the packets in the network. It has been shown that a hash scheme with polynomial complexity can be used without the need for secret-key distribution. However, the use of a block code forces an a-priori decision on the coding rate.

The key idea of *packet-based detection schemes* is that some of the intermediate nodes in the network can detect polluted data on-the-fly and drop the corresponding packets, thus retransmitting only valid data. However, packet-based detection schemes require active participation of intermediate nodes and are dependent on hash functions, which are generally computationally expensive. Alternatively, this type of attack can be mitigated by signature schemes based on homomorphic hash functions. The use of homomorphic hash functions is specifically tailored for network-coding schemes, since the hash of a coded packet can easily be derived from the hashes of previously encoded packets, thus enabling intermediate nodes to verify the validity of encoded packets prior to mixing them algebraically. Unfortunately, homomorphic hash functions are also computationally expensive.

There exists a homomorphic signature scheme for network coding that is based on Weil pairing in elliptic-curve cryptography. Homomorphic hash functions have also been considered in the context of peer-to-peer content distribution with rateless erasure codes for multicast transfers. With the goal of preventing both the waste of large amounts of bandwidth and the pollution of download caches of network clients, each file is compressed to a smaller hash value, with which receivers can check the integrity of downloaded blocks. Beyond its independence from the coding rate, the main advantage of this process is that it is less computationally expensive for large files than are traditional forward error-correction codes (such as Reed–Solomon codes).

A cooperative security scheme can be used for on-the-fly detection of malicious blocks injected in network coding-based peer-to-peer networks. In order to reduce the cost of verifying information on-the-fly while efficiently preventing the propagation of malicious blocks, the authors propose a distributed mechanism whereby every node performs block checks with a certain probability and alerts its neighbors when a suspicious block is found. Techniques to prevent denial-of-service attacks due to the dissemination of alarms are available.

The basic idea is to take advantage of the fact that in linear network coding any valid packet transmitted belongs to the subspace spanned by the original set of vectors. A signature scheme is thus used to check that a given packet belongs to the original subspace. Generating a signature that is not in the subspace yet passes the check has been shown to be hard.

A comparison of the bandwidth overhead required by Byzantine error-correction and -detection schemes can be carried out as follows. The intermediate nodes are divided into regular nodes and trusted nodes, and only the latter are given access to the public key of the Byzantine detection scheme in use. Under these assumptions, it is shown that packet-based detection is most competitive when the probability of attack is high, whereas a generation-based approach is more bandwidth-efficient when the probability of attack is low.

Key-distribution schemes

The ability to distribute secret keys in a secure manner is an obvious fundamental requirement towards assuring cryptographic security. In the case of highly constrained mobile ad-hoc and sensor networks, key pre-distribution schemes emerge as a strong candidate, mainly because they require considerably less computation and communication resources than do trusted-party schemes or public-key infrastructures. The main caveat is that secure connectivity can be achieved only in probabilistic terms, i.e. if each node is loaded with a sufficiently large number of keys drawn at random from a fixed pool, then with high probability it will share at least one key with each neighboring node.

It has been shown that network coding can be an effective tool for establishing secure connections between low-complexity sensor nodes. In contrast with pure key-pre-distribution schemes, it is assumed that a mobile node, e.g. a hand-held device or a laptop computer, is available for activating the network and for helping to establish secure connections between nodes. By exploiting the benefits of network coding, it is possible to design a secret-key-distribution scheme that requires only a small number of pre-stored keys, yet ensures that shared-key connectivity is established with a probability of unity and that the mobile node is provably oblivious to the distributed keys.

The basic idea of the protocol, which is illustrated in Figure 9.5, can be summarized in the following tasks:

(a) prior to sensor-node deployment:
   (1) a large pool P of N keys and their N identifiers are generated off-line;
   (2) a different subset of L keys drawn randomly from P and the corresponding L identifiers are loaded into the memory of each sensor node;
   (3) a table is constructed with the N key identifiers and N sequences that result from performing an XOR of each key with a common protection sequence X;
   (4) the table is stored in the memory of the mobile node;

(b) after sensor-node deployment:
   (1) the mobile node broadcasts HELLO messages that are received by any sensor node within wireless transmission range;

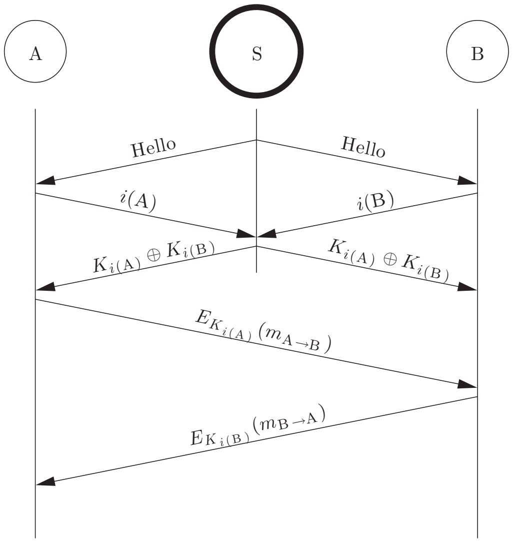
A
S
B
Hello
Hello
$i(A)$
$i(B)$
$K_{i(A)} \oplus K_{i(B)}$
$K_{i(A)} \oplus K_{i(B)}$
$E_{K_{i(A)}}(m_{A\to B})$
$E_{K_{i(B)}}(m_{B\to A})$

Figure 9.5 Secret-key-distribution scheme. Sensor nodes A and B want to exchange two keys via a mobile node S. The process is initiated by a HELLO message broadcast by S. Upon receiving this message, each sensor node sends back a key identifier $i(\cdot)$ corresponding to one of its keys $K_{i(\cdot)}$. Node S then broadcasts the result of the XOR of the two keys, $K_{i(\mathrm{A})} \oplus K_{i(\mathrm{B})}$. Once this process is concluded, sensor nodes A and B can communicate using the two keys $K_{i(\mathrm{A})}$ and $K_{i(\mathrm{B})}$ (one in each direction). Here, $E_{K_{i(\mathrm{A})}}(m_{\mathrm{A}\to\mathrm{B}})$ denotes a message sent by A to B, encrypted with $K_{i(\mathrm{A})}$, and $E_{K_{i(\mathrm{B})}}(m_{\mathrm{B}\to\mathrm{A}})$ corresponds to a message sent by B to A, encrypted with $K_{i(\mathrm{B})}$.

(2) each sensor node replies with a key identifier;
(3) on the basis of the received key identifiers the mobile node locates the corresponding sequences protected by X and combines them through an XOR network-coding operation, thus canceling out X and obtaining the XOR of the corresponding keys;
(4) the mobile node broadcasts the resulting XOR sequence;
(5) by combining the received XOR sequence with its own key, each node can easily recover the key of its neighbor, thus sharing a pair of keys that is kept secret from the mobile node.

Although the use of network coding hereby presented is limited to XOR operations, more powerful secret-key-distribution schemes are likely to result from using linear combinations of the stored keys.

9.8 Bibliographical notes

The field of network coding emerged from the seminal work of Ahlswede, Cai, Li, and Yeung [184], who proved that the multicast capacity of a general network can be achieved only if intermediate nodes are allowed to encode incoming symbols. Li, Yeung, and Cai proved that linear codes [185] are sufficient to achieve the aforementioned multicast capacity. The algebraic framework for network coding developed by Koetter and Médard in [186] and the development of random linear network coding by Ho *et al.* in [187] led to practical applications in which the nodes in the network generate linear combinations of information symbols using random coefficients. A practical approach to random linear network coding was proposed and tested by Chou, Wu, and Jain in [188]. Microsoft presented the first application of network coding for content distribution in a paper by Gkantsidis and Rodriguez [189]. A system implementation of XOR-based network coding for wireless networks (the COPE protocol) was presented by Katti *et al.* in [190]. Secure network coding with wiretapping limited to a subset of the network links was first addressed by Cai and Yeung in [191]. The connection between secure network coding and the wiretap channel of type II of Ozarow and Wyner [192] was investigated by Rouayheb and Soljanin in [193]. A weak criterion for the algebraic security provided by network coding was presented by Bhattad and Narayanan in [194]. The security potential of the algebraic structure of network coding in large-scale networks was analyzed by Lima, Médard, and Barros in [195]. Vilela, Lima and Barros provided in [196] a lightweight security solution for network coding, whereby only the coding coefficients must be encrypted. The proposed scheme found its first application in wireless video [197]. Solutions for Byzantine attacks were presented by Jaggi *et al.* in [198] and by Gkantsidis and Rodriguez in [199]. An information-theoretic treatment of network error correction was provided by Cai and Yeung in [200]. Koetter and Kschischang later presented code constructions for network error correction in [201]. Kim *et al.* provided a unified treatment of robust network coding for peer-to-peer networks under Byzantine attacks in [202]. Oliveira, Costa, and Barros investigated the properties of wireless network coding for secret-key distribution in [203]. Kim, Barros, Médard, and Koetter showed how to detect misbehaving nodes in wireless networks with network coding using an algebraic watchdog in [204]. A detailed account of the many facets of network coding can be found in the books by Ho and Lun [205] and by Fragouli and Soljanin [206, 207].