Network Coding and Wireless Physical-layer Secret-key Generation: From Unequal Erasure Protection (UEP) to Unequal Security Protection (USP)

by

Apirath Limmanee

A thesis submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

in Electrical Engineering

Date of Defense: September 29, 2011

School of Engineering and Science
Jacobs University

Acknowledgments

This thesis would not have been completed without the help of my supervisor, Prof. Dr. Werner Henkel, as well as the advice from other two dissertation committee members, Prof. Dr. Jon Wallace and Prof. Dr. Eduard Jorswieck. I have been financially supported by Jacobs University, German National Science Foundation (Deutsche Forschungsgemeinschaft, DFG), and the Bulgarian National Science Fund, and would therefore like to express my gratitude for them.

In addtion, I would like to thank some of my current and former colleagues for invaluable help and encouragement. These include Dr. Khaled Hassan, Dr. Fangning Hu, Dr. Neele von Deetzen, Khodr Ahmad Saaifan, Oana Alexandra Graur, Humberto Neto, and Alexandra Filip.

Abstract

The general abstraction of this thesis is the relationships between two seemingly unrelated topics, which are network coding and wireless physical-layer secret-key generation (WPSG). As build-ups to such relationships, some specific aspects of each topic are discussed in detail at first. To begin with, network coding issues of unequal erasure protection (UEP) and degree distribution distortion of LT-coded symbols are presented. After that, the analysis regarding key length and security enhancement in WPSG is given. Towards the end, relationships between network coding and WPSG are revealed in two aspects, which are security protocols using network coding and scalable security based on the weakly secure network coding concept.

Declaration of originality

I hereby declare that the research recorded in this proposal was composed and originated entirely by myself in cooperation with my advisor in the School of Engineering and Science, Jacobs University, Germany.

Apirath Limmanee

Contents

List of Figures

Part I

Motivation, Overview, and Introduction

Chapter 1

Motivation and Overview

This thesis consists of a series of topics which are divided into three parts. We decide to do so with the aim that each part will be self-contained, i.e., the reader may choose to read or skip one part without any serious damage. For example, those who have sound knowledge in communication systems and networks as well as network coding may skip Part I, those who are interested solely in network coding may read only Part II, and those who only wish to know the relationship between network coding and WPSG may read only the last part.

This first part is the introductory part consisting of three chapters. Chapter 2 discusses the basic knowledge of digital communication systems and networks. Chapter 3 discusses some aspects of graph theory related to network coding, as well as network coding itself, which is the major focus of this thesis.

At the beginning of the millennium, Ahlswede et al. [55] made a breakthrough in information theory by inventing network coding, which significantly increased network throughput by means of coding. The idea related information theory to graph theory, providing a means to reach the graph-theoretic max-flow limit in data networks. It stated that when many sources transmitted different data to many sinks, or when one source transmitted the same data to many sinks (multicast), the maximum amount of data flow (max-flow) could not be achieved by means of routing alone. Network coding was needed to achieve the max-flow.

Many developments have followed. In 2003, Li et al. [69] showed that a simple class of network coding, called linear network coding, sufficed to achieve the maximum flow. In the same year, Koetter and Médard [59] proposed a greedy algorithm to find linear network coding solutions to the multicast problem. However, Koetter and Médard's algorithm was based on the idealized assumption that the given network had no delay, lost no packet, and had centralized knowledge of its topology, which was not the case in practice. Later, some more practical algorithms were proposed.

In 2003, Chou, Wu, and Jain [51] simulated a distributed network coding scheme using the network topologies of several internet service providers, which resulted in almost optimal throughput. Moreover, several pioneering works from 2004 to 2007 [53,54,65,70, 83,84] introduced network coding to wireless communications. Even cryptography became one of network coding applications with the introduction of secure network coding in 2002 [25,46] as well as a subsequent work in [29].

We, however, have maintained a critical stance on network coding. While network coding brings several research opportunities to the field of digital communication, several issues must be dealt with as one looks closely. We focus on the issue of erasures in the network, which will be examined in Part II, consisting of Chapter 4 and 5. We do not merely raise the issue, but also suggest the solutions.

Chapter 4 discusses unequal erasure protection (UEP) of network codes, suggesting that a network code assignment has a strong impact on the quality of the received scalable data for each receiver. Therefore, an assigment scheme based on the concept of equity among sink nodes is proposed, together with a capitalist concept used to assign network codes by means of an auction such that richer nodes get better data quality. After that, Chapter 5 investigates the problem of degree distribution distortion of LT-coded symbols when network coding is used and suggests a solution.

In Part III, We turn our attention to a cryptographic technique called wireless physical-layer secret-key generation (WPSG). In Chapter 6, we investigate the extension of the technique from a direct communication to a relayed one as well as a wireless network in general. In the latter case, the security of the technique relies on the network protocol used for transmitting pilot symbols. It is shown that information mixing in a similar manner to network coding can enhance the protocol security.

In Chapter 7, it is shown that the security of WPSG can be further enhanced by means of key encoding, using the same concept of secret sharing as that in secure network coding. In addition, a concept of scalable security or unequal security protection (USP) is formally introduced and analyzed in the context of both secure network coding and WPSG.

Finally, Chapter 8 gives the conclusion and discusses future research topics.

Chapter 2

Introduction to Digital Communication Systems and Networks

Although the objective of our investigation is communication networks in general, we would like, at first, to introduce the elements of point-to-point digital communication in Section 2.1, as well as the mathematical models of communication channels in 2.2. After that, routing in communication networks is introduced in 2.3.

2.1 Basic Elements of A Secure Digital Communication System

In this section, we discuss those basic elements constituting a digital communication system. When talking about any communication system in general, three basic elements must certainly be there. They are transmitter, channel, and receiver. However, when discussing a digital communication system in particular, we need to be specific about what constitutes the transmitter and the receiver. According to Proakis [30], a basic digital communication system can be shown in a block diagram in Fig. 2.1.

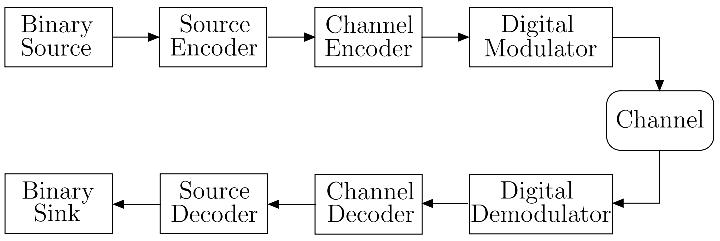

Figure 2.1: A basic digital communication system

To make the system secure, however, we need to insert two more blocks, which are encryptor and decryptor, as shown in Fig. 2.2.

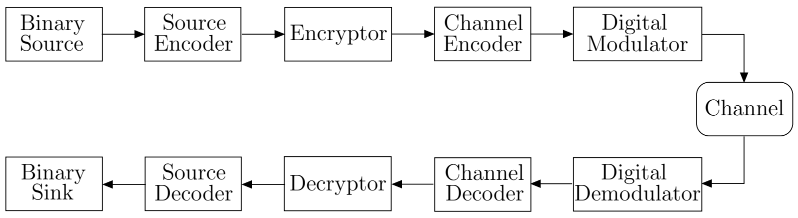

Figure 2.2: A secure digital communication system

A digital transmitter consists of four basic elements, which are a binary source, a source encoder, a channel encoder, and a digital modulator, whereas a digital receiver is composed of a digital demodulator, a channel decoder, a source decoder, and a binary sink.

A binary source in the transmitter generates information to be transmitted, be it audio, speech, or video data, in a binary representation. Usually, this representation is not the most economical one possible, i.e., it is possible to squeeze it into a shorter string of 0s and 1s without reducing the amount of information, or reducing only some insignificant amount of it. The source encoder is made for this task. It transforms the data into another binary representation with less redundancy. These two blocks, however, are not the focus of this thesis.

Our concern in the thesis ranges from the third block, the encryptor, to the ninth block, the decryptor. Although the encryptor in Fig. 2.2 locates right after the source encoder, it is not necessarily the case when we consider some new cryptographic technologies, for example, physical-layer security and physical-layer secret-key generation, which will be discussed in detail in Part III.

An encryptor provides security to the transmitted data. In cryptology, there are two types of security, which are theoretical security and practical security. They are based on different philosophies. Theoretical security, on the one hand, is based on theoretical impossibility of ciphers being broken, necessitating mutual knowledge of a secret key between transmitter and receiver. Practical security, on the other hand, is based on practical difficulty of ciphers being broken, i.e., it takes too much time and labor to break them. The questions regarding how these two types of security are implemented will be dealt with in Part III.

A channel encoder introduces some redundancy to the input data so that the output is immune to deteriorating effects, such as errors and erasures, from the channel. There are numerous channel codes and several ways to classify them. They can be divided into block codes and convolutional codes, fixed-rate and rateless codes, or linear and non-linear codes. In this thesis, we will consider linear rateless codes called LT-codes in Chapter 5.

The purpose of a digital modulator is to map the binary representation after channel encoding into electromagnetic waves that can be transmitted along the channel. The simplest digital modulation is called binary modulation, which maps the binary digit 0 into a waveform $s_o(t)$ and the binary digit 1 into $s_1(t)$. This means each bit is transmitted in a separated waveform. In M-ary modulation ($M > 2$), however, $M = 2^m$ waveforms $s_i(t), i = 0, 1, ..., M - 1$ are used to represent m bits [30].

The communication channel is the physical medium used for transmission, e.g., the free space in wireless communication. In the design and analysis of digital communication systems, communication channels are represented by channel models, which are mathematical abstractions of these physical entities. We will discuss some channel models in the next section.

A digital demodulator, a channel decoder, a decryptor, and a source decoder perform reversed operations of a digital modulator, a channel encoder, an encryptor, and a source encoder, respectively. Their overall purpose is to give the binary sink the data which is as close as possible to that generated by the binary source.

2.2 Channel Models

Since channel models are mathematical abstractions of physical channels, they can be divided into several categories based on two factors, which are the nature of physical channels and the levels of abstraction used. We will discuss four frequently used channel models which suit the purpose of this thesis.

2.2.1 Binary Erasure Channel (BEC)

Being proposed in 1955 by Elias [52], binary erasure channels become practical for data networks, especially after the emergence of the Internet. The model offers no possibility of transmission errors. It assumes that the data is either received correctly or not received at all. This assumption can only hold on network level of abstraction, where errors are assumed to be corrected by the data link layer. In reality, erasures may be caused by buffer overflows at intermediate routers, mismatching of packets' internal check-sum, or packets losing their ways [10].

The model can be illustrated by Fig. 2.3, in which X is the transmitted symbol and Y is the received one. If the erasure probability is p_e, the capacity C of the channel is given by [73]

$$C = 1 - p_e. \tag{2.1}$$

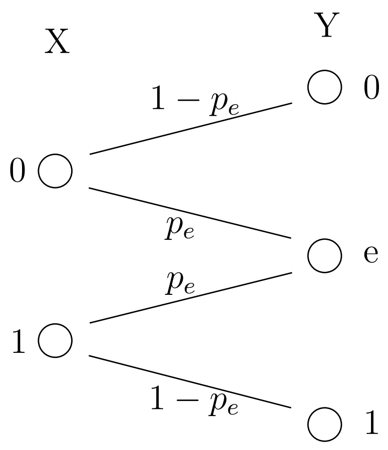

Figure 2.3: Binary Erasure Channel

2.2.2 Binary Symmetric Channel (BSC)

This model only allows received symbols to contain errors, but not to be erased. However, the physical phenomena behind the transmission errors are not included in the model. Therefore, this model is suitable for the data link level of abstraction. It can be illustrated by Fig. 2.4, in which X is the transmitted symbol and Y is the received one. If the error probability is p_s, the capacity C of the channel is given by [73]

$$C = 1 - H(p_s), \tag{2.2}$$

where

$$H(p_s) = p_s \log_2 \frac{1}{p_s} + (1 - p_s) \log_2 \frac{1}{1 - p_s}, \tag{2.3}$$

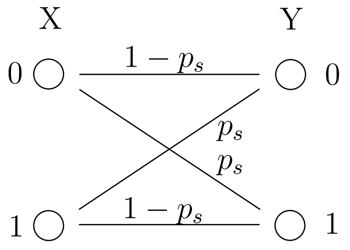

Figure 2.4: Binary Symmetric Channel

2.2.3 Additive White Gaussian Noise (AWGN) Channel

This channel model, belonging to the physical level of abstraction, explains the transmitted signal distortion as an adding effect of some noise $n(t)$, as follows.

$$r(t) = s(t) + n(t), \tag{2.4}$$

where $r(t)$ is the received signal at time t, $s(t)$ is the transmitted signal, and $n(t)$ is the noise which obeys the following Gaussian distribution.

$$p_n(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\{-(x - m_x)^2/2\sigma^2\}, \tag{2.5}$$

where m_x is the mean of the distribution, which is zero in this case, and σ^2 is the noise variance.

The assumption of a Gaussian distribution is supported by the natural phenomenon of thermal agitation of charge carriers inside an electrical conductor. The amplitude is almost Gaussian distributed, and the frequency spectrum is almost white, thus the name additive white Gaussian noise (AWGN) [30].

2.2.4 Rayleigh Fading Channel

The Rayleigh fading channel model is a physical abstraction of wireless channels, where signal rays from the transmitter are scattered by obstacles in the environment, forming a number of transmission paths to the receiver, who receives the summation of signals from these paths. Therefore, the received band-pass signal may be expressed as follows [30].

$$r(t) = \sum_n \alpha_n(t) s[t - \tau_n(t)], \tag{2.6}$$

where $\alpha_n(t)$ is the attenuation factor for the signal received on the n^{th} path and $\tau_n(t)$ is the propagation delay of the n^{th} path. Now, if the transmitted signal $s(t)$ is expressed as

$$s(t) = \Re[s_l(t) e^{j2\pi f_c t}], \tag{2.7}$$

where $s_l(t)$ is the low-pass transmitted signal in complex base-band representation and f_c is the carrier frequency. The received low-pass signal in base-band can be written as

$$r_l(t) = \sum_n \alpha_n(t) e^{-j2\pi f_c \tau_n(t)} s_l[t - \tau_n(t)]. \tag{2.8}$$

According to (2.8), the equivalent low-pass channel in base-band can be described by the following time-variant channel impulse response

$$c(\tau; t) = \sum_n \alpha_n(t) e^{-j2\pi f_c \tau_n(t)} \delta[t - \tau_n(t)]. \tag{2.9}$$

When the number of paths is large enough, the central limit theorem holds that the channel impulse response behaves according to a zero-mean, complex-valued, Gaussian distribution. If we denote the random variables of the real part, the imaginary part, and the envelope of the impulse response by C_R, C_I, and C_E, respectively, it follows that

$$C_E = \sqrt{C_R^2 + C_I^2}. \tag{2.10}$$

If C_R and C_I are independent Gaussian-distributed random variables with zero mean and a variance of σ^2, the probability density function of C_E can be expressed as

$$p_{C_E}(r) = \frac{r}{\sigma^2} \exp -r^2/2\sigma^2. \tag{2.11}$$

This distribution is called Rayleigh, giving rise to the name Rayleigh fading channel. Note that when there are fixed scatterers or lines of sight in the channel, the zero-mean assumption does not hold and the distribution is therefore not Rayleigh-distributed but Ricean distributed. We, however, will not discuss the Ricean distribution in detail.

2.3 Routing in Communication Networks

In more complicated networks such as the Internet, there are more channels and other elements involved in transmission. These elements are situated between the transmitter and the receiver and are often called intermediate nodes. Traditionally, there are three important functions that these nodes may choose to perform to help packets reach the destination. These are store-and-forward, amplify-and-forward, and decode-and-forward. After the emergence of network coding, the last two functions can be modified into amplify-and-forward with network coding and decode-and-forward with network coding, respectively.

No matter how sophisticated the signal processing techniques are at the intermediate nodes before packets are forwarded, the most important thing is that packets must be forwarded in the right direction. Otherwise, they will not reach the destination or might take a decade to reach it. Therefore, in this section, we discuss the most basic operation, store-and-forward, which is usually called routing. The nodes performing this function are called routers.

2.3.1 Routing Protocols and Algorithms

A routing protocol specifies rules for communication among routers. A routing algorithm is a part of the protocol that deals with route selection. Let us consider the Internet as an example. The Internet can be seen as a collection of sub-networks, each of which having its own specific protocol, connecting together. Since each sub-network is independent of all others, it is often called an autonomous system (AS). A routing algorithm within an AS is called an interior gateway protocol, whereas that used for routing between ASes is called an exterior gateway protocol [7].

The assembly of sub-networks in the Internet is made possible by a mutually agreed protocol, which is called the Internet Protocol (IP). As a network-layer protocol, IP provides best-effort (not guaranteed) services to the transport layer, i.e., transporting datagrams from source to destination, regardless of whether they are in the same network or not. At present, IP version 4 (IPv4) is the dominant protocol, but IP version 6 (IPv6) is also emerging.

Since the Internet is too big a scope to deal with in this thesis, we will now turn our attention to routing inside a network, not between networks. There are two types of routing algorithms, which are static algorithms and dynamic ones. Static algorithms, as the name implies, do not base their routing decisions on measurements or estimates of the current traffic and topology. One example of such algorithms is Dijkstra's shortest-path routing, which routes packets along the shortest path between the transmitter and receiver. Another example is flooding, in which every incoming packet is sent out to every line except the one via which it is received [7].

Despite the simplicity of static algorithms, modern computer networks generally use dynamic algorithms which can adapt the routing decisions to current network topology and load. Two of the most popular dynamic algorithms are distance vector routing and link state routing.

In distance vector routing, each router maintains a routing table containing one entry for each of the other routers. The entry contains two parts, the preferred outgoing line used for sending data to the router of that entry and the time or distance needed. The metric used to select the outgoing line may take number of hops, time delay, or number of packets queued along the path into account. Then, each router periodically transmits the time or distance needed to reach other routers to its neighbors, who update their tables accordingly.

The pitfalls of distance vector routing algorithm are that it does not take line bandwidth into account and takes a long time to converge. Therefore, it was used in the ARPANET only until 1979 before being replaced by link state routing. Unlike distance vector routing, in which each router informs only its neighbors about the cost or delay to all other routers, in link state routing, each router informs all others only about the delay or cost to its neighbors. Then, every router can compute the path that minimizes the cost [7].

Now, we would like to specifically introduce routing algorithms for broadcast and multicast routing in wireline networks.

2.3.2 Broadcast and Multicast Routing in Wireline Networks

Broadcast Routing in Wireline Networks

There are four major algorithms used for broadcasting a packet to all routers.

1. Flooding

Flooding means that every incoming packet is sent out to every outgoing line except the one on which it arrives. This generates vast numbers of duplicate packets. Three measures can be taken to reduce such numbers. One of them is to keep a hop counter in the packet header such that the packet is discarded when the number of hops reaches the tolerable maximum. Another technique is to make sure that the same packet is not transmitted twice by the same router. The last one is called selective flooding, meaning that routers only send the packets only along the lines going approximately to the right direction [7].

2. Multi-destination Routing

In this method, each packet contains either a list of destinations or a bit map indicating the desired destinations. When a packet arrives at a router, the router calculates the best route for each destination and determines the set of all output lines needed by all calculated best routes. The router generates a copy of the packet for each output line and includes, in the packet, a list of those destinations using the line. This process repeats for every router along the best routes until every destination receives the packet [7].

3. Spanning Tree

A spanning tree is a subset of the subnet. It is a graph whose set of nodes comprises all routers in the network. Every pair of nodes are connected by a sequence of edges but there is no loop in the graph. Figure 2.5(b) gives an example of the spanning tree. In this method, the router simply copies an incoming packet to all output lines belonging to the spanning tree except the one on which the packet arrives [7].

4. Reverse Path Forwarding

This algorithm is an attempt to approximate the previous one, even when the routers know nothing about the spanning tree. In this algorithm, when a packet arrives at a router, the router checks to see if it arrives on the line normally used for sending packets to the broadcasting source. If that is the case, there is a good chance that packet has followed the best route from the source and is the first copy arrived at the router. The router therefore copies the packet onto all lines except that on which the packet arrives. However, if the packet arrives on a line other than the preferred one used for reaching the source, the packet is discarded as a duplicate [7].

Multicast Routing in Wireline Networks

Multicasting requires group management so that packets are sent only to those interested in it and authorized to see it. In multicast routing, each router computes a spanning tree covering all other routers in each group. In Fig. 2.5(a), we have two groups, 1 and 2. The spanning tree for the leftmost router to all nodes is shown in Fig. 2.5(b). When a process would like to multicast a packet to a group, the first router examines its spanning tree and prunes it. Figures 2.5(c) and (d) show the pruned spanning tree for groups 1 and 2, respectively. Multicast packets are forwarded only along the pruned multicast tree of the desired destination group [7].

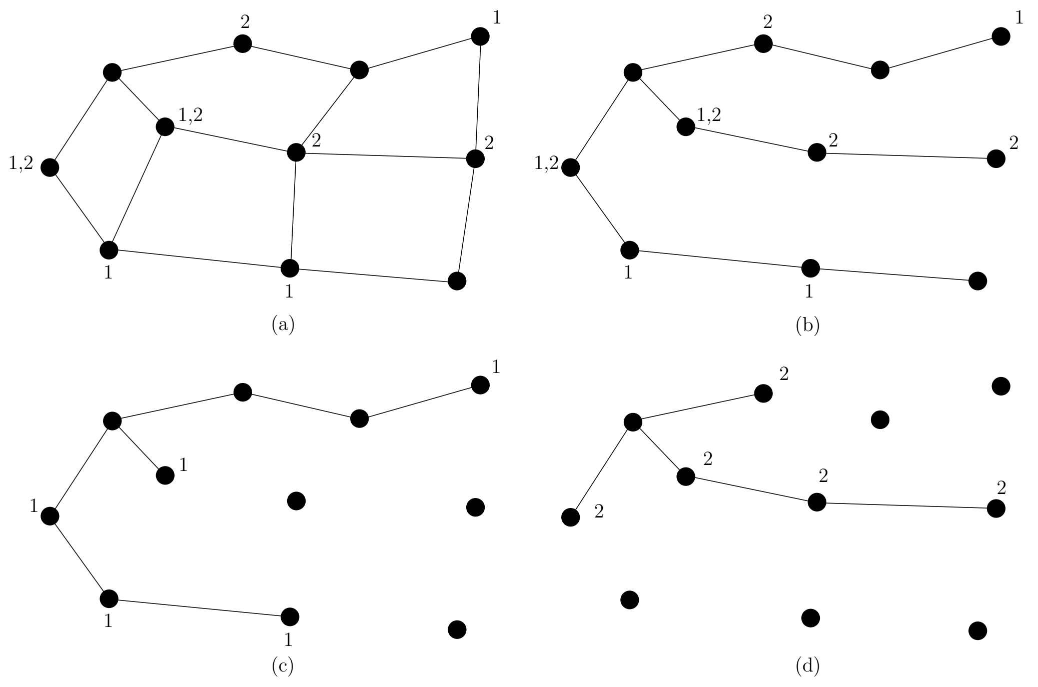

Figure 2.5: (a) A network. (b) A spanning tree for the leftmost router. (c) A multicast tree of group 1. (d) A multicast tree of group 2

Chapter 3

Introduction to Graphs and Network Coding

Since network coding relates graph theory to communication theory, an introduction to the concept of a graph is important for a complete understanding of network coding. In this chapter, after the introduction to graphs in Section 3.1, we explain the max-flow/min-cut theorem in Section 3.2. Network coding is primarily the solution to the max-flow problem in multicast application and will be introduced in Section 3.3. A general formulation of network coding is given in Section 3.4, with a special case of linear network coding in Subsection 3.4.1.

3.1 Graphs and Some Basic Definitions

A graph is a mathematical abstraction and simplification of many physical phenomena, thus enabling us to solve real-world problems under the branch of mathematics called graph theory. Graph theory is employed to tackle problems in such fields as communication engineering, computer science, physics, chemistry, biology, and sociology. Examples of graph-theoretical problems which find many applications are the shortest-path problem, the spanning tree problem, the max-flow problem, and the min-cost flow problem.

In general, a graph consists of only two essential sets of elements, which are the set V

of nodes and the set E of edges. Therefore, a graph G is usually represented by the pair of those two sets such that $G = (V, E)$. Here, we give the following definitions of three types of graphs relevant to our topic, which are simple graphs, multigraphs, and directed graphs [17, 79].

Definition 3.1 *A simple graph G_S consists of a node set $V(G_S)$ and an edge set $E(G_S)$, where each edge is a pair (u, v) of nodes $u, v \in V(G_S)$. Each pair of nodes u and v in G_S is joined by at most one edge.*

Definition 3.2 *A multigraph G_M consists of a node set $V(G_M)$ and an edge set $E(G_M)$, where each pair of nodes u and v in G_M may be joined by more than one edge. Thus, each edge joining a pair of nodes $u, v \in V(G_M)$ is represented by $(u, v)_i$, where i is the index ensuring that the representation of each edge is distinct.*

Definition 3.3 *A directed graph G_D consists of a node set $V(G_D)$ and an edge set $E(G_D)$, where each edge is an ordered pair (u, v), that must be distinguished from (v, u), of nodes $u, v \in V(G_D)$.*

Figures 3.1 and 3.2 give examples of a simple directed graph and a multigraph, respectively. Note that Fig. 3.1 is also a multigraph since, according to our definitions, every simple graph is a multigraph, whereas Fig. 3.2 is neither a simple graph nor a directed graph since three edges connect F and G and no direction is associated with each edge.

When a directed graph is used in practice, it is useful to introduce a variable that measures the quantity flowing in each edge, such as the electric current in a circuit, or, in our case, the data rate in a network. We simply call this quantity "flow." The flow of (u, v) is denoted by a real number x_{uv}. In practice, a positive x_{uv} suggests that the flow moves along the same direction of the arc in a directed graph, whereas a negative flow moves in the opposite direction. The amount of the flow is limited by the capacity in both directions such that $b_{uv} \leq x_{uv} \leq c_{uv}$, where c_{uv} and b_{uv} are the capacities of (u, v) in the forward and backward directions, respectively [17].

Now, we define the term "divergence," which will be used later when considering the max-flow problem [17].

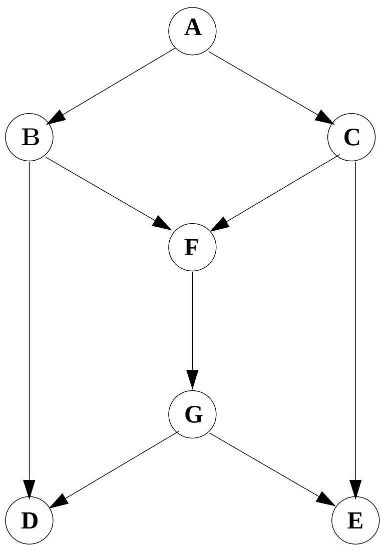

Figure 3.1: An example of a simple directed graph

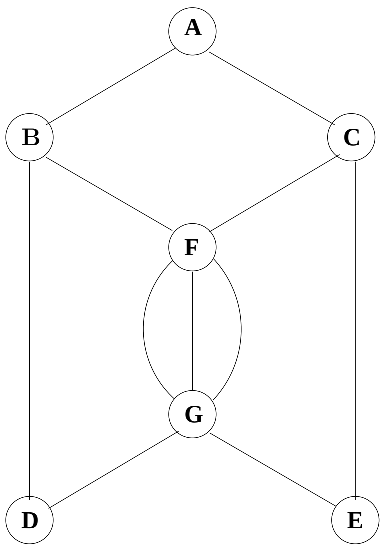

Figure 3.2: An example of a multigraph

Definition 3.4 *In a graph G, the divergence of the node i, denoted by y_i, is the total flow departing from the node i less the total flow arriving at it.*

$$y_i = \sum_{\{j|(i,j)\in E(G)\}} x_{ij} - \sum_{\{j|(j,i)\in E(G)\}} x_{ji} \quad ;\forall i \in V(G) \tag{3.1}$$

When there is an integer-number capacity associated with each edge in a graph representing a digital communication network, such a graph is usually a simple graph, whose edges only denote connectivity among nodes. However, in some cases, the communication capacity between each pair of nodes is indicated by the number of edges connecting them. It follows that multigraphs are required in such cases. For examples, a multigraph in Fig. 3.2 may represent a communication network in which the capacity of the link FG is three times of that belonging to BD.

Before proceeding to the max-flow/min-cut theorem, we would like to introduce some more terms, which are paths, cycles, and acyclic graphs. These terms are important in communication networks since, ideally, data transmission should follow a path from the source to the destination, rather than being caught up in cycles. Therefore, an acyclic graph is normally used as an abstraction of a communication network.

Definition 3.5 *A path P from u to v is a sequence of nodes u_0, u_1, u_2,..., u_k obeying the following properties.*

1. *$u = u_0$ and $v = u_k$.*
2. *An edge joins u_i to u_{i+1}, $i \in 0, 1, 2, ..., k-1$.*
3. *All u_i, $i \in 0, 1, 2, ..., k$, are distinct nodes.*

The length of the path P is the number of edges used for connecting the nodes, which is k [79].

Definition 3.6 *A cycle C is a sequence of nodes u_0, u_1, u_2,..., u_k forming a u_0u_k path, in which there is an edge joining u_k and u_0.*

The length of the cycle C is the number of edges used for connecting the nodes, which is $k+1$ [79].

Definition 3.7 *An acyclic graph is a directed graph without any cycle.*

3.2 The Max-Flow/Min-Cut Theorem

Now, we would like to introduce the max-flow problem in a digital communication network. In this problem, we have a graph with two special nodes, the source S and the sink T. This is an optimization problem whose objective is to maximize the amount of information flowing from S to T. Speaking in terms of flows and divergences, the objective is to find a flow vector that makes the divergences of all nodes other than S and T equal 0 while maximizing the divergence of S. In [17], the max-flow problem is formulated as a special case of the minimum cost flow problem. By adding an artificial arc from T to S, as shown by the dashed line in Fig. 3.3, the max-flow problem becomes a minimum cost flow problem when the cost coefficient at every edge is zero, except the cost of the feedback edge TS, which is -1. In this case, the problem can be formulated in Definition 3.8 [17].

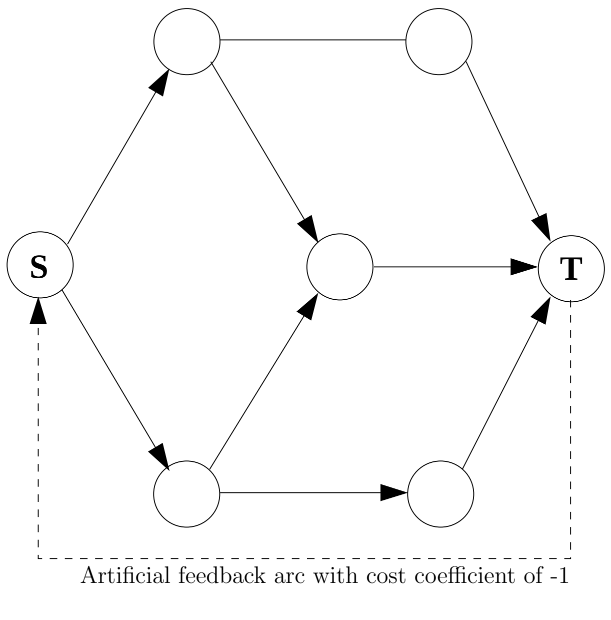

Figure 3.3: Illustration of the max-flow problem as a special case of the minimum cost flow problem [17]

Definition 3.8 *Max-flow Problem:*

maximize x_{TS}

subject to

$$\sum_{\{j|(i,j)\in E(G)\}} x_{ij} - \sum_{\{j|(j,i)\in E(G)\}} x_{ji} = 0,\ \forall i \in V(G) \text{ with } i \neq S \text{ and } i \neq T, \tag{3.2}$$

$$\sum_{\{j|(S,j)\in E(G)\}} x_{Sj} = \sum_{\{i|(i,T)\in E(G)\}} x_{iT} = x_{TS}, \tag{3.3}$$

$$b_{ij} \leq x_{ij} \leq c_{ij},\ \forall (i,j) \in E(G) \text{ with } (i,j) \neq (T,S). \tag{3.4}$$

The Ford-Fulkerson algorithm can be used to solve the max-flow problem. The basic idea is to recursively increase the flow by finding a path from S to T that is unblocked with respect to the flow vector until we can verify that the maximum flow is achieved. Such verification can be done by checking whether there exists a "saturated cut" separating S from T [17]. The following four definitions explain the meaning of the term "saturated cut."

Definition 3.9 *A cut Q in a graph G is a partition of the node set $V(G)$ into two nonempty subsets, a set $\mathcal{C}$ and its complement $V(G) - \mathcal{C}$. We use the notation*

$$Q = [\mathcal{C}, V(G) - \mathcal{C}]. \tag{3.5}$$

Note that the partition is ordered such that the cut $[\mathcal{C}, V(G) - \mathcal{C}]$ is distinct from $[V(G) - \mathcal{C}, \mathcal{C}]$. For a cut $[\mathcal{C}, V(G) - \mathcal{C}]$, using the notation

$$\begin{aligned} Q^+ &= \{(i,j) \in E(G) | i \in \mathcal{C}, j \notin \mathcal{C}\}, && (3.6)\\ Q^- &= \{(i,j) \in E(G) | i \notin \mathcal{C}, j \in \mathcal{C}\}, && (3.7) \end{aligned}$$

we call Q^+ and Q^- the sets of forward and backward arcs of the cut, respectively.

Definition 3.10 *The flux $F(Q)$ across a cut $Q = [\mathcal{C}, V(G) - \mathcal{C}]$ is the total net flow* *coming out of $\mathcal{C}$ such that*

$$F(Q) = \sum_{(i,j)\in Q^+} x_{ij} - \sum_{(i,j)\in Q^-} x_{ij}. \tag{3.8}$$

Definition 3.11 *Given lower and upper flow bounds b_{ij} and c_{ij} for each edge (i, j), the capacity $C(Q)$ of a cut Q is*

$$C(Q) = \sum_{\{j|(i,j)\in Q^+\}} c_{ij} - \sum_{\{j|(i,j)\in Q^-\}} b_{ij}. \tag{3.9}$$

Definition 3.12 *Q is said to be a saturated cut with respect to a set of flow if and only if $F(Q) = C(Q)$.*

Using the Ford-Fulkerson algorithm, the first saturated cut that we find is called the minimum cut or min-cut representing the network bottleneck. This saturated cut is "minimum" in the sense that its flux is minimum among all possible saturated cuts. As suggested by intuition as well as the following max-flow/min-cut theorem, the capacity of the minimum cut equals the maximum flow of the network. The theorem is formally proved in [17].

Theorem 3.1 *Max-Flow/Min-Cut Theorem:*
(a) If x^* is an optimal solution of the max-flow problem, then the divergence out of S corresponding to x^* is equal to the minimum cut capacity over all cuts separating S from T.
(b) If all lower arc flow bounds are zero, the max-flow problem has an optimal solution, and the maximal divergence out of S is equal to the minimum cut capacity over all cuts separating S from T.

Figure 3.4 illustrates the max-flow/min-cut theorem. We can see from (b) that the capacity of the minimum cut is 5, equaling the maximum flow to T in (c).

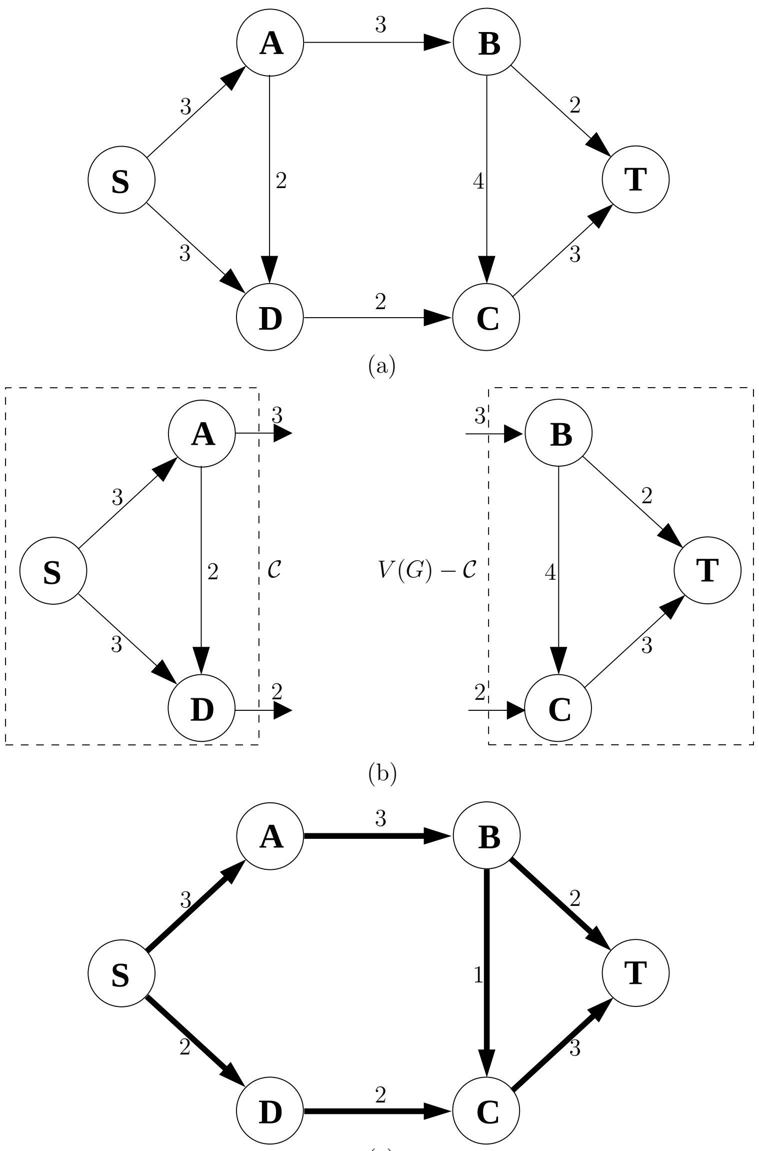

Figure 3.4: (a) A graph G with the associated edge capacities, (b) the minimum cut of G, and (c) the maximum flow of G

3.3 Introduction to Network Coding

In the paper "Network Information Flow" published in July 2000, Ahlswede, Cai, and Li, stated that "Contrary to one's intuition, our work reveals that it is in general not optimal to regard the information to be multicast as a fluid which can simply be routed or replicated. Rather, by employing coding at the nodes, which we refer to as network coding, bandwidth can in general be saved [55]."

While traditional coding is performed at the transmitter, network coding is done by routers, denoted by Ahlswede et al. as nodes. They proved in their paper that information that was multicast from one transmitter, the source node, to a set of receivers, the sink nodes, could achieve its maximum flow (throughput) by network coding, without which this optimum might not be achieved [55].

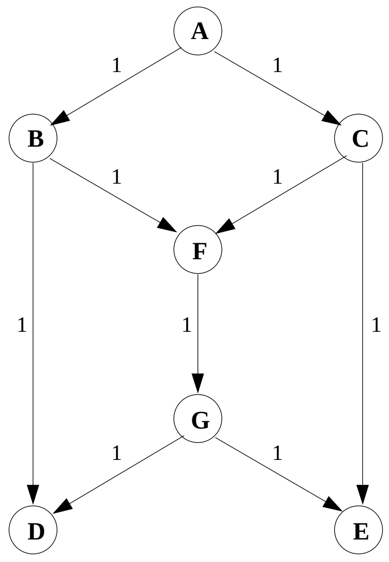

Figure 3.5: The butterfly network with edge capacities

To understand how network coding works, consider the graph in Fig. 3.5, which is usually called the butterfly network. The graph consists of 7 nodes where the source node A wishes to multicast some information to the sink nodes D and E via the 9 edges with their capacities as labeled. The unit of the capacity here is bit per unit time.

Now consider Fig. 3.6, where b_1 and b_2 are multicast to D and E. (This means both D and E will receive both b_1 and b_2.) We can see from Fig. 3.6(a) that D receives b_1 via the path ABD and b_2 via ACFGD, as well as from Fig. 3.6(b) that E receives b_1 via ABFGE and b_2 via ACE. Since the edge FG is shared between sending b_2 to D and sending b_1 to E, its capacity for each of them is halved. Therefore, the maximum flow in this multicast session for D, without using network codes, is 1.5 bits per unit time (1 from the path ABD and 0.5 from ACFGD). The maximum flow for E is also the same.

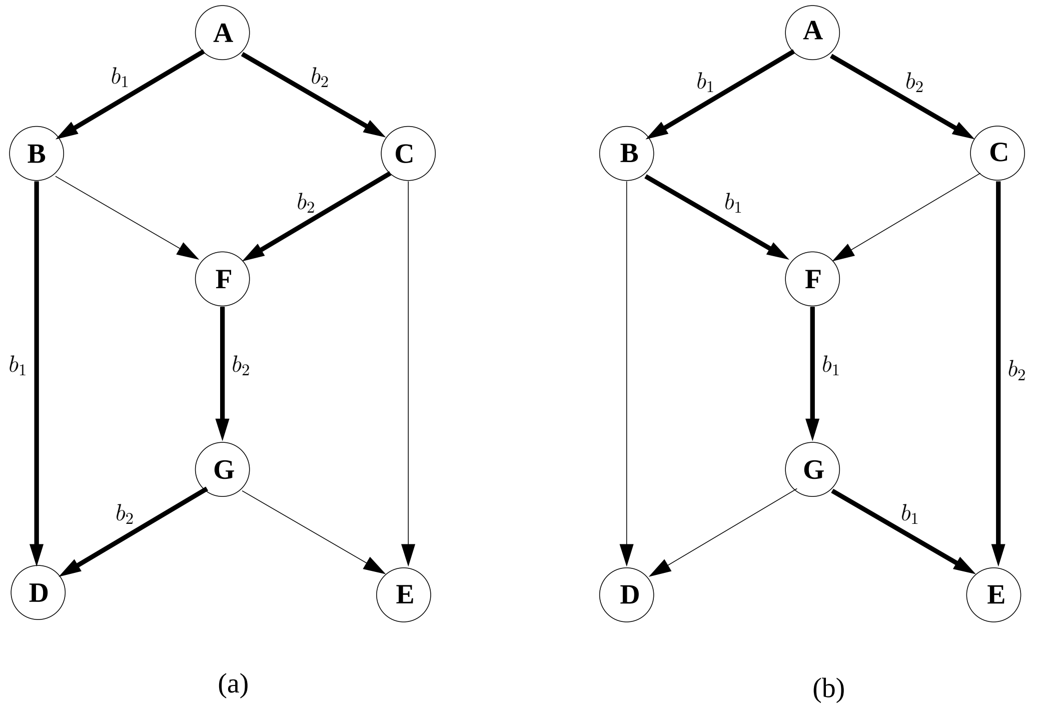

Figure 3.6: Illustration of information flow within the butterfly network

However, network coding can help both D and E achieve the maximum flow of 2 bits per unit time, as shown in Fig. 3.7. The node F does the simplest form of coding by XORing b_1 to b_2. As a result, in one unit time, D receives b_1 and $b_1 \oplus b_2$ whereas E gets b_2 and $b_1 \oplus b_2$, all of which can be easily decoded into b_1 and b_2. Note that the maximum flow of 2 bits per unit time is the same as the maximum flow for each individual flow in the absence of another, which constitutes the limit that no multicast flow can exceed, according to Ahlswede et al. [55].

Since real communication networks are more complicated than the butterfly network, we need a mathematical model to treat the general problems of network coding. This is explained in the following section.

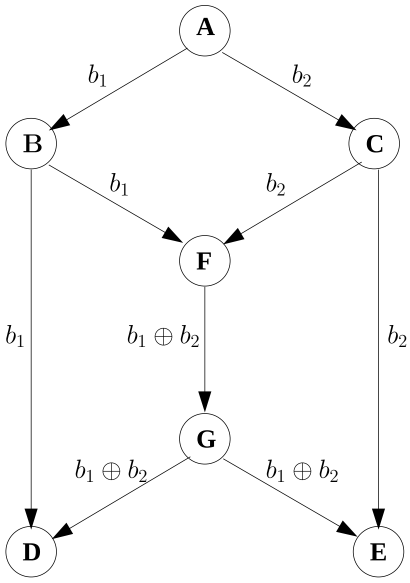

Figure 3.7: Network coding in a butterfly network

3.4 Theory of Network Coding

For the sake of this section's discussion, the source node will be depicted as a rectangle, as shown in Fig. 3.8. In addition, each edge will represent the capacity of 1 data unit per time unit. A data unit refers to one element of a base field $\mathbb{F}$, for instance, $\mathbb{F} = GF(2)$, which is the binary field. A message consists of ω data units and is therefore represented by an ω-dimensional vector $\mathbf{x}$, where $\mathbf{x} \in \mathbb{F}^{\omega}$ [61]. Note that although the source may send several messages, the data in each message can only be network-coded with that in the same message.

For each non-source node U, $\mathcal{I}(U)$ denotes the set of its incoming edges and $\mathcal{O}(U)$ denotes that of outgoing ones. $|\mathcal{I}(U)|$ and $|\mathcal{O}(U)|$ denote the number of incoming edges and outgoing edges, respectively, at the node U. For a source node S, $\mathcal{I}(S)$ is a set of imaginary edges, which terminate at S without any originating node, as shown by dashed arrows in Fig. 3.8. We always have ω imaginary edges in a graph.

Although network coding in real networks can be performed at all nodes except the sinks, in order to correctly decode, the sinks need not know how all intermediate nodes do the coding. All they need to know is the accumulated effect that all network coding has on the source messages. Chou, Wu, and Jain, suggest in the paper "Practical Network

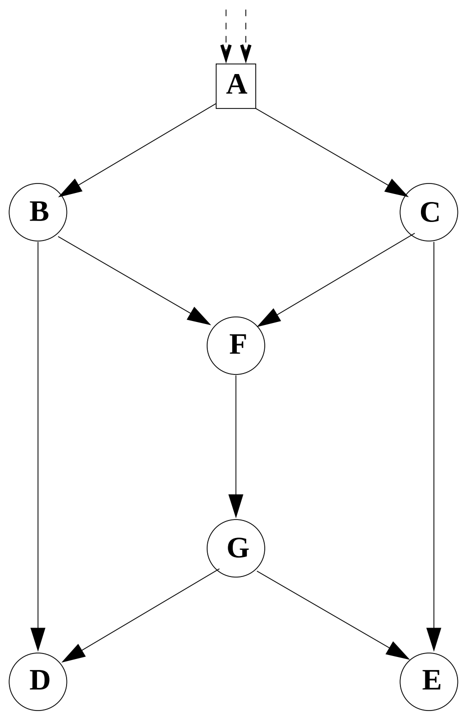

Figure 3.8: The butterfly network with imaginary edges

Coding" [51] that each intermediate node combines the network coding from the previous nodes along the path with the coding of its own before writing the information about the combined code on data packets' headers and sending the packets downstream. Here, the function of the combined code is called "global encoding mapping" whereas that of the local code is called "local encoding mapping." They are defined as follows.

Definition 3.13 *Let $\mathbb{F}$ be a finite field and ω a positive integer. An ω-dimensional $\mathbb{F}$-valued network code on an acyclic communication network consists of a local encoding mapping*

$$\tilde{k}_e : \mathbb{F}^{|\mathcal{I}(U)|} \to \mathbb{F} \tag{3.10}$$

for each node U in the network and each edge $e \in \mathcal{O}(U)$ [61].

Definition 3.13 tells us that the local encoding mapping $\tilde{k}_e$ for a specific outgoing edge e of the node U maps the $|\mathcal{I}(U)|$-dimensional vector $\mathbb{F}^{|\mathcal{I}(U)|}$, each element of which represents the data from a distinct incoming edge in the set $\mathcal{I}(U)$, into an outgoing data.

Definition 3.14 *Let $\mathbb{F}$ be a finite field and ω a positive integer. An ω-dimensional $\mathbb{F}$-*

valued network code on an acyclic communication network consists of a local encoding mapping $\tilde{k}_e : \mathbb{F}^{|\mathcal{I}(U)|} \to \mathbb{F}$ *and a global encoding mapping* $\tilde{f}_e : \mathbb{F}^{\omega} \to \mathbb{F}$ *for each edge* e *in the network such that:*

- *For every node* U *and every channel* $e \in \mathcal{O}(U)$, $\tilde{f}_e(\mathbf{x})$ *is uniquely determined by the global encoding mappings from incoming edges* $(\tilde{f}_d(\mathbf{x}), d \in \mathcal{I}(U))$, *and* $\tilde{k}_e$ *is the mapping via*

$$(\tilde{f}_d(\mathbf{x}), d \in \mathcal{I}(U)) \to \tilde{f}_e(\mathbf{x}) \tag{3.11}$$

- *For the* ω *imaginary edges* e, *the mappings* $\tilde{f}_e$ *are the projections from the space* $\mathbb{F}^{\omega}$ *to the* ω *different coordinates [61].*

Definition 3.14 shows formally the difference between the local encoding mapping $\tilde{k}_e$ and the global one $\tilde{f}_e$. The former, on the one hand, relates the data from incoming edges in the set $\mathcal{I}(U)$ to the outgoing data. The latter, on the other hand, maps the source message $\mathbf{x} \in \mathbb{F}^{\omega}$ to the data output at e. Moreover, (3.11) tells us that each node can derive $\tilde{f}_e$ from its local encoding mapping $\tilde{k}_e$ and the global encoding mapping $\tilde{f}_d$ of the incoming edges $d \in \mathcal{I}(U)$. This means any global encoding mapping computed by a node and written on data packets' headers will be used by successive nodes, together with their local encoding mapping, to successively compute their global encoding mapping to be further transmitted as headers. The process repeats until data packets reach the sinks, which decodes them according to headers.

This work focuses on a class of network codes called linear network codes, which is described in the next subsection.

3.4.1 Linear Network Codes

Linear network codes are those of which all local encoding mapping and global encoding mapping are nothing more than dot-product of input vectors and mapping vectors. Linear local and global encoding mapping are defined formally in definitions 3.15 and 3.16 [61].

Definition 3.15 *Let the ω-dimensional row vector $\mathbf{x}$ represent the message generated by the source node S, a global encoding mapping $\tilde{f}_e(\mathbf{x})$ is called linear if there exists an ω-dimensional column vector $\mathbf{f_e}$ such that*

$$\tilde{f}_e(\mathbf{x}) = \mathbf{x} \cdot \mathbf{f_e} \quad . \tag{3.12}$$

Definition 3.16 *Let the $|\mathcal{I}(U)|$-dimensional row vector $\mathbf{y}$ represent the data units received at the node U, a local encoding mapping $\tilde{k}_e(\mathbf{y})$ is called linear if there exists an $|\mathcal{I}(U)|$-dimensional column vector $\hat{\mathbf{k}}_\mathbf{e} = [k_{1,e}\ k_{2,e}\ \ldots\ k_{|\mathcal{I}(U)|,e}]^T$ such that*

$$\tilde{k}_e(\mathbf{y}) = \mathbf{y} \cdot \hat{\mathbf{k}}_\mathbf{e} \quad . \tag{3.13}$$

In Definition 3.17, the phrase "adjacent pair" is defined to facilitate definitions 3.18 and 3.19. Definition 3.18 defines the local encoding kernel matrix $\mathbf{K_U}$ at the node U as a concatenation of all the linear local encoding mappings $\hat{\mathbf{k}}_\mathbf{e}$ at U. Definition 3.19 shows that the vector $\mathbf{f_e}$ in Definition 3.15, which is used for the global encoding mapping at the node U to the outgoing edge e, can be derived from the global encoding mappings $\mathbf{f_d}$ of all the incoming edges d and the local encoding kernel $\mathbf{K_U}$. $\mathbf{f_e}$ is now called global encoding kernel.

Definition 3.17 *A pair of edges (d, e) is called an adjacent pair if and only if there exists a node U such that $d \in \mathcal{I}(U)$ and $e \in \mathcal{O}(U)$ [61].*

Definition 3.18 *Let $\mathbb{F}$ be a finite field and ω a positive integer. An ω-dimensional $\mathbb{F}$-valued linear network code on an acyclic communication network consists of a scalar $k_{d,e}$, called the local encoding kernel, for every adjacent pair (d, e). The local encoding kernel at the node U means the $|\mathcal{I}(U)| \times |\mathcal{O}(U)|$ matrix $\mathbf{K_U}$, of which element at the d^{th} row and e^{th} column is $k_{d,e}$.*

$$\mathbf{K_U} = [k_{d,e}]_{d \in \mathcal{I}(U), e \in \mathcal{O}(U)} \tag{3.14}$$

Note that $\mathbf{K_U}$ is a concatenation of all vectors $\hat{\mathbf{k}}_\mathbf{e}$, $e \in \mathcal{O}(U)$ in Definition 3.16 [61].

Definition 3.19 *Let $\mathbb{F}$ be a finite field and ω a positive integer. An ω-dimensional $\mathbb{F}$-valued linear network code on an acyclic communication network consists of a scalar $k_{d,e}$ for every adjacent pair (d, e) in the network as well as an ω-dimensional column vector $\mathbf{f_e}$ for every channel e such that*

$$\mathbf{f_e} = \sum_{d \in \mathcal{I}(T)} k_{d,e} \cdot \mathbf{f_d} \tag{3.15}$$

where $e \in \mathcal{O}(T)$. In addition, the vectors $\mathbf{f_e}$ for the ω imaginary edges $e \in \mathcal{I}(S)$ form the natural basis of the vector space $\mathbb{F}^\omega$. The vector $\mathbf{f_e}$ is called the global encoding kernel for the channel e [61].

From (3.12) and (3.15), we can now write the linear global encoding mapping $\tilde{f}_e(\mathbf{x})$ in the form of local encoding kernel's elements and global encoding kernels from incoming edges.

$$\tilde{f}_e(\mathbf{x}) = \mathbf{x} \cdot \mathbf{f_e} = \mathbf{x} \cdot \sum_{d \in \mathcal{I}(T)} k_{d,e} \cdot \mathbf{f_d} = \sum_{d \in \mathcal{I}(T)} k_{d,e}(\mathbf{x} \cdot \mathbf{f_d}) \tag{3.16}$$

Example 3.1 *Figure 3.9 shows all the global encoding kernels $\mathbf{f_e}$ according to network coding in Fig. 3.7. In this case, the message $\mathbf{x}$ is a 2-dimensional vector of the binary field, $\mathbf{x} = [b_1\ b_2]$.*

We can see that for the edge AB,

$$\tilde{f}_{AB}(\mathbf{x}) = \tilde{f}_{AB}(b_1, b_2) = [b_1\ b_2] \cdot [1\ 0]^T = b_1 \tag{3.17}$$

Similarly, $\tilde{f}_{BF}(b_1, b_2) = b_1$, $\tilde{f}_{AC}(b_1, b_2) = \tilde{f}_{CF}(b_1, b_2) = b_2$, $\tilde{f}_{FG}(b_1, b_2) = b_1 + b_2$, and so on. Now, considering the node F, we can see that, from (3.13),

$$\tilde{k}_{FG}(\mathbf{y}) = k_{FG}(b_1, b_2) = [b_1\ b_2] \cdot \hat{\mathbf{k}}_{\mathbf{FG}} = b_1 + b_2. \tag{3.18}$$

Therefore, $\hat{\mathbf{k}}_{\mathbf{FG}} = [1\ 1]^T$ and $\mathbf{K_F} = [1\ 1]^T$ since there is only one outgoing edge FG. Similarly, $\hat{\mathbf{k}}_{\mathbf{GD}} = \hat{\mathbf{k}}_{\mathbf{GE}} = [1]$ and $\mathbf{K_G} = [1\ 1]$, which is the concatenation of $\hat{\mathbf{k}}_{\mathbf{GD}}$ and $\hat{\mathbf{k}}_{\mathbf{GE}}$.

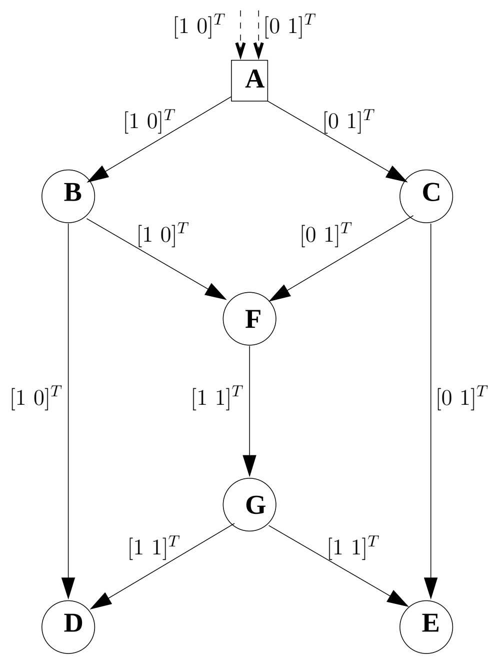

Figure 3.9: The butterfly network with global encoding kernels labelling all edges

Part II

Network Coding Issues: Unequal Erasure Protection and Degree Distribution Distortion in LT-Coded Symbols

The emergence of scalable video coding standard means that some parts of transmitted data should be better protected than others. This concept is called unequal error/erasure protection (UEP). We will show in Chapter 4 that network coding inherently supports this concept. However, this will inevitably leads to conflicts among receiving nodes in a multicast session. We provide an economic analysis of the conflicts as well as an auction algorithm to resolve them.

Rateless codes, such as LT-codes and Rapter codes, are originally designed to guarantee that erasures in the network do not affect data recovery. This seems, at first thought, to imply that they do not support UEP, since the recovery of every data priority seems to be guaranteed. However, a subsequent work by Rahnavard, vellambi, and Fekri, shows that UEP can still be implemented using a similar concept of unequal recovery time (URT) [47]. Their technique ensures that, for an unlimited time, data of all priorities is recovered, but for a specific time range, higher-priority data will be recovered before the lower-priority one. This allows the receiving nodes to decide how long they want to spend time receiving data, according to the number of priorities that they need.

However, when rateless codes are used in a network with network coding, another problem arises. We call it the degree distribution distortion problem. This will make the recovery time of every data priority longer than the normal case without network coding. In Chapter 5, we investigate this problem in LT-codes with the simplest case of a butterfly network and gives a solution.

Chapter 4

Unequal Erasure Protection (UEP) in Network Coding

4.1 Introduction to Erasures and Unequal Erasure Protection (UEP) in Network Coding

Recent works on network coding consider errors and erasures in networks [5, 58, 60, 82], whereas earlier ones model networks as graphs in which each edge represents an erasure-free channel with a unit capacity [14, 61, 63]. In this chapter, we consider each edge in the network to represent the data transmission rate of one symbol per unit time in a binary erasure channel (BEC), i.e., the edge capacity is reduced from 1 to $1-p$, if p denotes the erasure probability. We discuss this in the context of generalized networks in Section 4.2.

When data is of different importance, it is natural that one prefers to better protect the high-priority data than the low-priority one against errors and erasures. This concept is called unequal error/erasure protection (UEP).

Scalable video and image data, such as Scalable Video Coding (SVC) standardized by JVT as an extension of H.264/AVC, consists of several layers of data [64]. Upper layers represent fine details added to lower ones. As shown in Fig. 4.1, when the third layer data is missing, the receiver can only recover the first two layers since the recovery of the last two layers depends on successful recovery of the third. Scalable data therefore asks for unequal erasure protection (UEP), sometimes mentioned as unequal loss protection (ULP), such that the parts of data with higher priority are better protected against erasures.

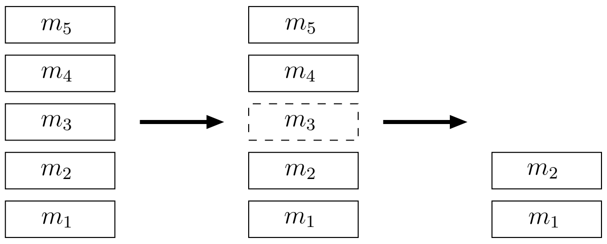

Figure 4.1: Recovery of layered data with the third layer missing

UEP has been implemented in several components of digital communication systems. UEP bit-loading for multi-carrier modulation was studied in [77], UEP coded modulation in [50], UEP bit-loading for MIMO-OFDM (Multiple-Input Multiple-Output Orthogonal Frequency Division Multiplexing) in [37], UEP Turbo codes based on puncturing and pruning in [78], and UEP LDPC Codes based on irregular variable and/or check node degrees in [41, 68].

Unequal-erasure-protected network coding was proposed for the first time by us in [5]. We analyzed the effect of erasures on the recovery of scalable data when linear network coding was applied. We found that global encoding kernels (GEKs) describing linear network codes had different levels of built-in unequal-erasure-protecting (UEP) capability, allowing better protection of high-priority data when GEKs were wisely assigned. UEP in network coding is introduced in Section 4.3.

We define some related economic terms and concepts of scalable data in Section 4.4, which allows us to give a mathematical expression of the expected utility of recovered scalable data in the presence of erasures in Section 4.5. In Section 4.6, we quantitatively show that linear network codes have built-in UEP mechanisms affecting the utility of the recovered scalable data. Section 4.7 discusses a UEP network code assignment problem, which will be simplified by a procedure described in Section 4.8 before being solved by two strategies suggested in Section 4.9 and 4.10. The last section discusses the results and concludes the topic.

4.2 The Edge-Disjoint Path Model with Binary Erasure Channels for Network Coding

Figure 4.2 shows network coding in a network that is usually called the butterfly network, where A aims to multicast two binary symbols b_1 and b_2 to D and E. In Fig. 4.2, we can see that the node F encodes b_1 and b_2 together to achieve the multicast rate of 2 bits per unit time, if each edge represents the capacity of one bit per unit time, D receives b_1 from the path ABD and can recover b_2 from the symbol $b_1 \oplus b_2$ from $ACFGD$, whereas E receives b_2 from the path ACE and recovers b_1 from the symbol $b_1 \oplus b_2$ from $ABFGD$. Had network coding not been there, only either b_1 or b_2 would have been able to pass the bottleneck FG in one unit time, i.e., one receiver would have been unable to use one of its possible transmission paths [55].

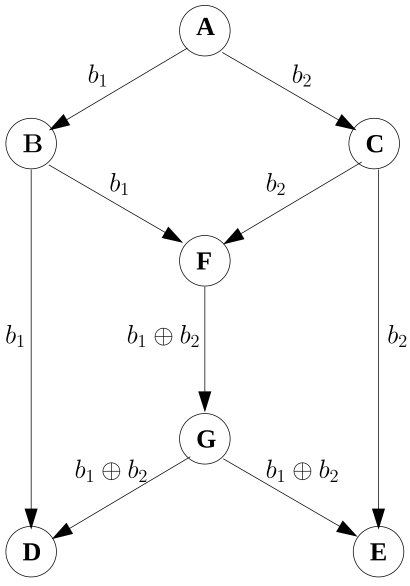

Figure 4.2: Network coding in a butterfly network

By means of network coding, all receivers can use all of their possible paths at the same time. In general, the multicast rate of ω suggests the existence of ω non-intersecting paths from the source to each sink, which are called edge-disjoint paths by Jaggi et al. [63], although the paths destined to different receivers may share some edges.

If each edge is modeled as a binary erasure channel (BEC), the erasure probability of an edge-disjoint path can be computed from erasure probabilities of all the edges belonging to that path from the source to the sink. The loss of a symbol in each edge-disjoint path does not affect the recovery in another. For example, the loss of b_1 at the edge BF does not affect the recovery of b_2 at D via $ACFGD$.

Let $p_{i,j,k}$ represents the erasure probability of the k^{th} edge belonging to the j^{th} path of the i^{th} sink. The overall erasure probability of data symbols from the j^{th} path belonging to the i^{th} sink, denoted by $P_{e,ij}$, becomes [5]

$$P_{e,ij} = 1 - \prod_{k=1}^{|E(i,j)|} \left(1 - p_{i,j,k}\right), \tag{4.1}$$

where $|E(i,j)|$ denotes the number of edges in the j^{th} path of the i^{th} sink.

4.3 UEP Issues in Network Coding

Let us consider Fig. 4.2 again and see what will happen if one symbol is received at each sink whereas another one is erased. At D, if $b_1 \oplus b_2$ is erased, D still obtains b_1, but if b_1 is erased, D obtains neither b_1 nor b_2. This means, for D, b_1 is better protected than b_2. On the other hand, for E, b_2 is better protected. In case b_1 and b_2 are equally important, the network coding is fair. However, if b_1 is more important than b_2 and the erasure probability of each symbol is the same, D is in favor because its more important symbol is better protected. Thus, to achieve fairness and optimality, the network encoding function of each edge-disjoint path should be carefully chosen [5].

The preferred choice of network codes for each receiver may vary depending on the situation. Although, normally D has no reason to prefer the network coding pattern shown in Fig. 4.2 to that in Fig. 4.3, it will change its mind if suddenly the link BD is broken. Thus, clever choice of network codes can also provide insurance.

Since the problem of insurance and fairness in the distribution of wealth is an economic

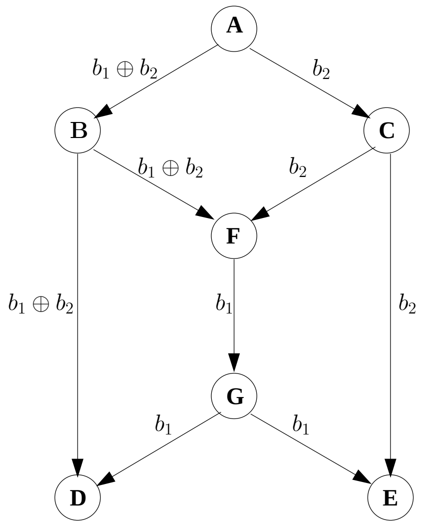

Figure 4.3: Another network coding pattern in a butterfly network

problem, so is our problem. Therefore, in the next section, we will formulate it using such economic terms as utility, marginal value, and anticipation.

4.4 The Utility of Scalable Data

The “utility” is a numerical value representing the amount of satisfaction that a user receives from an object. In our case, a “user” simply refers to a receiver, whereas the term “object” deserves a careful consideration. If we define an object by a layer of scalable data, it might appear at first that we now have several types of objects, since each layer of data has different properties. However, for the sake of simplicity and orderliness, we will treat all the layers as the same type of objects which are arranged in order of importance. Moreover, each layer corresponds to an inseparable object, i.e., half a layer is considered meaningless. Thus, the word “layer” to be considered here needs not be the same as the one defined in any particular technical standard. It simply denotes an inseparable unit of data symbols. In this way, our formulation will correspond to the following law of diminishing marginal utility in economics [6].

Law 4.1 : Law of Diminishing Marginal Utility

1. The utility that each receiver gains from receiving scalable data depends on the number *of received data layers.*

2. The extra utility that a receiver gains from an increase in the number of received layers, which is called a marginal utility, is less if the previously received number of layers is greater.

According to Law 4.1, if any two receivers have exactly the same capability to extract utility from the same amount of received data, it can be concluded that an unequal amount of received layers between them will make the marginal utility of the one with privilege less than the other. Conversely, if a given receiver have more capability to extract utility, an equal distribution results in the larger marginal utility for that receiver.

The following definitions formally express the utility concept in scalable data. We formulate, in Definition 4.1, the mapping of scalable data into a scalable message in which each successive layer adds more details to the data. Therefore, each element in the scalable message corresponds to the "object" of our interest. After explaining the term "dependency level" in Definition 4.2, we define the "cumulative utility vector" and "marginal utility vector" corresponding to the law of diminishing marginal utility in Definition 4.3 and 4.4, respectively. Then, we define an "ordered set of scalable data" that can be mapped into an "ordered scalable message" in Definition 4.5, which possesses an interesting practical property obeying Law 4.1: The incremental quality obtained by each recovered layer in the message is in a decreasing order. The ordered scalable message thus always prefers more erasure protection in the preceding symbols than the subsequent ones.

Definition 4.1 *For any $\mathcal{S} = \{s_1, s_2, ..., s_\omega\}$, representing a set of scalable data with progressively increasing quality from s_1 to s_ω, the i^{th} element s_i can be mapped into a prefix vector $\mathbf{P_i}$= $[m_1, m_2, ..., m_i]$ of a scalable message $\mathbf{M}$= $[m_1, m_2, ..., m_\omega]$, which is an ω-dimensional row vector of a finite field $\mathbb{F}$. [3, 5]*

$$s_i \longmapsto [m_1, m_2, ..., m_i] \tag{4.2}$$

Definition 4.2 *A symbol m_k belonging to the scalable message $\mathbf{M} = [m_1, m_2, ..., m_\omega]$ has a dependency level of $\lambda(m_k) = j$ if its significance depends on the successful recovery of the symbols having the dependency level of $j-1$ but not on those with larger dependency level. A symbol of which significance does not depend on any symbol has the dependency level of 1. [3, 5]*

Definition 4.3 *A functional vector $\mathbf{U_k}(\mathcal{S}) = [u_k(s_1), u_k(s_2), ..., u_k(s_\omega)]$, $k = 1, 2, ..., N$, where N is the number of sink nodes in the network-coded multicast, is called the cumulative utility vector assigned by the sink node k to the set of scalable data $\mathcal{S}$ described in Definition 4.1 if each element $u_k(s_i)$, $1 \leq i \leq \omega$, is a non-negative real number representing the private utility that the sink node k assigns to the scalable data s_i. [3]*

Definition 4.4 *A functional vector $\Delta\mathbf{U_k}(\mathcal{S})$ is called the marginal utility vector assigned by the sink node k to the set of scalable data $\mathcal{S}$ described in Definition 4.1 if*

$$\begin{aligned} \Delta\mathbf{U_k}(\mathcal{S}) &= [\Delta_1, \Delta_2, ..., \Delta_\omega] && (4.3) \\ &= [u_k(s_1) - u_k(s_0), u_k(s_2) - u_k(s_1), ... \\ &\quad ..., u_k(s_\omega) - u_k(s_{\omega-1})], && (4.4) \end{aligned}$$

where $u_k(s_0) = 0$ and $u_k(s_i)$, $1 \leq i \leq \omega$, represents the element in the cumulative value vector $\mathbf{U_k}(\mathcal{S})$ defined in Definition 4.3. [3]

Definition 4.5 *The set of scalable data $\mathcal{S}$ is said to be ordered if and only if the two following conditions are fulfilled.*

$$\begin{aligned} \lambda(m_u) &\geq \lambda(m_v), \quad 1 \leq v < u \leq \omega && (4.5) \\ \Delta_{i-1} &\geq \Delta_i, \quad 1 < i \leq \omega && (4.6) \end{aligned}$$

where $\lambda(m_j)$ and Δ_i denote the dependency level of the symbol m_j and the i^{th} element in the marginal utility vector of $\mathcal{S}$, respectively. A message $\mathbf{M}$ corresponding to an ordered set $\mathcal{S}$ of scalable data is called an ordered scalable message. [3]

4.5 Utility of Scalable Data as Probabilistic Objects

Since data transmission is a random process subject to probabilistic errors and erasures, we need to add probabilistic aspects into our concept of an “object” according to the following laws.

Law 4.2 : Laws of a Probabilistic Object

1. The concept of an “object” is extended to include the combination of objects with stated probabilities. For example, if A and B are objects, a 30-70 chance of A or B is also an object.

2. If the object A is preferred to the object B, and B to the object C, there will be some probability combination of A and C such that the individual is indifferent between the combination and B.

These probabilistic aspects are in accordance with Friedman and Savage’s “The Utility Analysis of Choices Involving Risk [43],” since erasures and errors in our data transmission can be considered as risks.

From Definition 4.4, if the prefix $\mathbf{P}_{j-1}$ in the ordered scalable message M has already been successfully recovered, the recovery of the symbol m_j will increase the utility by Δ_j. Now, since the transmission channels are assumed to be binary erasure channels, the object obtained at each receiver is a probabilistic object in Law 4.2. This means, according to Paragraph 1, that there are chances of receiving A, B, C, etc., which are data with different qualities and probabilities. However, all of these can be considered as one object with its own utility. Generalizing the law in Paragraph 2 with a linearity assumption, we can derive the utility of such an object by the following expected value [3, 5].

$$E[U_i] = \sum_{j=1}^{\omega} \left[\prod_{l=1}^{j} \varrho_{i,l} \right] \cdot \Delta_j \tag{4.7}$$

$$= \sum_{j=1}^{\omega} \rho_{i,j} \cdot \Delta_j \quad , \tag{4.8}$$

where ϱ_{il} and $\rho_{i,j}$ represent the probabilities that the symbol m_l and the prefix $\mathbf{P_j}$ are recovered at the sink i, respectively. From (4.8), the quality improves if the term $\rho_{i,j}$ becomes larger, especially for a small j implying a large Δ_j. This reaffirms the essence of UEP, which is to better protect the high-priority preamble. $\rho_{i,j}$ depends on the transmission channels and our network codes, which will be investigated in the next section.

The equation (4.8) satisfies the paragraph 2 of Law 4.2 generalized by linearity assumption, such that any two probabilistic objects having the same expected value, despite being linearly-combined by different probabilistic proportion of data layers, are considered economically equivalent.

4.6 Utility of Global Encoding Kernels (GEKs) for Linear Network Codes

In the previous section, we consider scalable data as probabilistic objects and derive its utility. In this section, after having a quick review of the meaning of global encoding kernels (GEKs), we will see that the assignment of GEKs to the edges in the network is analogous to assigning received symbols to the sinks in an erasure-free network. In case there are erasures, we can identify the utility of GEKs based on erasure probabilities of transmission channels just as we can identify the utility of scalable data.

Let us now revisit the global encoding kernel (GEK) defined in Chapter 3. For a network that employs linear network coding, each of its edges in the graphical model, such as Fig. 4.2, is used to transmit the linear combination of the source symbols. This linear combination can either be represented locally as a linear function of symbols from adjacent edges, which is called “a local encoding mapping,” or globally as a linear function of source symbols, which is called “a global encoding mapping” [61]. The global encoding mapping is described by a vector called ”a global encoding kernel (GEK),” which is introduced in Definition 4.6.

Definition 4.6 *Let $\mathbb{F}$ be a finite field, ω a positive integer, and the ω-dimensional, $\mathbb{F}$-valued vector $\mathbf{M}$ the message generated by the source node $\mathcal{S}$. A function $f_e(\mathbf{M})$ of the edge e is said to be a linear global encoding mapping if there exists an ω-dimensional $\mathbb{F}$-valued column vector $\mathbf{f}_e$ such that*

$$f_e(\mathbf{M}) = \mathbf{M} \cdot \mathbf{f}_e. \tag{4.9}$$

$\mathbf{f}_e$ *is called the global encoding kernel [61].*

The assignment of the global encoding kernel $\mathbf{f}_e$ to the edge $e \in \mathrm{In}(T)$, where $\mathrm{In}(T)$ is the set of all incoming edges of the sink T, determines which linear combinations of symbols in the message $\mathbf{M}$ are received at T, when there is no erasure.

Since the symbols in the message have unequal utility, so do the combinations of them. It then follows logically that GEKs also have unequal utility. Before determining the utility of each GEK, we will classify the GEKs into levels based on the linear combination it yields. The levels of GEKs will be used to calculate their utility later.

We only consider the problem of assigning, given some local constraints, a suitable global encoding kernel for each edge according to the expected quality in (4.8), since the local encoding mapping can be easily derived thereafter.

To relate $\rho_{i,j}$ in (4.8) to network codes, we firstly define the UEP level of a global encoding mapping as follows [5].

Definition 4.7 *For a scalable message $\mathbf{M}$, which is an ω-dimensional row vector of a finite field $\mathbb{F}$, a global encoding mapping $\gamma_i(\mathbf{M})$ is of the i^{th} UEP level, $0 < i \leq \omega$, if there exists an ω-dimensional, $\mathbb{F}$-valued column vector $\mathbf{C}_i = [c_1, c_2, ..., c_i, 0, 0, ..., 0]^T$, $c_i \neq 0$, such that*

$$\gamma_i(\mathbf{M}) = \mathbf{M} \cdot \mathbf{C}_i = \sum_{j=1}^{i} c_j \cdot m_j. \tag{4.10}$$

$\mathbf{C}_i$ *is then called an i^{th}-UEP-level global encoding kernel.*

Now, let us consider an ordered scalable message $\mathbf{M} = [m_1, m_2, m_3]$, where m_1, m_2, and m_3 are binary symbols of the first, second, and third dependency level, respectively.

According to Definition 4.7, there exist seven possible GEKs, one of the first UEP level, two of the second level, and four of the third level, as shown in Table 4.1 [5].

Table 4.1: GEKs, their UEP levels, and the resulting network-coded symbols

UEP Levels	GEKs	Resulting Network-Coded Symbols
1	$[1\ 0\ 0]^T$	m_1
2	$[0\ 1\ 0]^T$	m_2
	$[1\ 1\ 0]^T$	$m_1 + m_2$
3	$[0\ 0\ 1]^T$	m_3
	$[1\ 0\ 1]^T$	$m_1 + m_3$
	$[0\ 1\ 1]^T$	$m_2 + m_3$
	$[1\ 1\ 1]^T$	$m_1 + m_2 + m_3$

From Table 4.1, the prefix $\mathbf{P}_2 = [m_1, m_2]$ can be recovered either from any two symbols from the levels 1 and 2 or from any three symbols from the level 3. For an arbitrary scalable message $\mathbf{M}= [m_1, m_2, ..., m_\omega]$, we can state as a general rule that, in order to recover the prefix $\mathbf{P}_i$, we need either i network coded-symbols belonging to i linearly independent GEKs of UEP levels not exceeding i, or more than i symbols in case some UEP levels of GEKs exceed i [5].

The implication following from the analysis is that the GEKs at lower levels have more utility than those in higher levels since they contribute more to the recovery of high-priority bits. In order to determine the expected utility in the presence of erasures, we now have to consider the erasure probability in each edge-disjoint path discussed in Section 4.2.

Reconsidering Eq. (4.8), assuming that the k^{th} edge-disjoint path of the sink i is used to transmit the k^{th} network-coded symbol to the sink i, the parameter $\rho_{i,j}$, which is the probability that the prefix $\mathbf{P}_j$ is recovered at the sink i, can be written as

$$\rho_{i,j} = \prod_{k=1}^{\omega} \mu_{j,k} \cdot [1 - P_{e,ik}] , \tag{4.11}$$

where $P_{e,ik}$ denotes the erasure probability of the path k used to transmit the k^{th} network-coded symbol to the sink i. $\mu_{j,k} = 1$ if the k^{th} network-coded symbol is needed to recover the prefix $\mathbf{P}_j$. Otherwise, $\mu_{j,k} = 0$ [5].

Equations (4.8) and (4.11) show that, by changing the GEKs allocated to the edges, the expected utility of received data varies. Therefore, if the receiver i is allowed to assign a GEK to an edge or a set of edges, it can do so in such a way that its utility increases the most, sometimes at the expense of other receivers. The more GEKs a node i is allowed to choose, the better its satisfaction. This means there is a marginal utility associated with each increase in the number of such permissions, which is defined as follows [3].

Definition 4.8 *A functional vector $\mathbf{\Theta}_i$ is called the marginal utility vector assigned by the sink node i to the GEK allocation permissions if*

$$\begin{aligned} \mathbf{\Theta}_i &= [\theta_1, \theta_2, ..., \theta_\zeta] \qquad (4.12)\\ &= [\mathcal{I}_{bi1} - \mathcal{I}_{wi1}, \mathcal{I}_{bi2} - \mathcal{I}_{wi2}, ... \\ &\quad ..., \mathcal{I}_{bi\zeta} - \mathcal{I}_{wi\zeta}], \qquad (4.13)\end{aligned}$$

where ζ is the number of possible permissions. $\mathcal{I}_{bij}$ is the expected utility $E[U_i]$ in (4.8) when the sink i is allowed to allocate j GEKs, whereas $\mathcal{I}_{wij}$ is that when it is only allowed to allocate $j-1$ GEKs before one worst-case GEK is assigned to it in the j^{th} allocation. Thus, the difference between $\mathcal{I}_{bij}$ and $\mathcal{I}_{wij}$ reflects how important it is for i to receive the j^{th} allocation permission.

4.7 The Problem of GEK Assignment

According to (4.8) and (4.11), the best option for the sink i is to make $\rho_{i,j}$ large for small j to satisfy the UEP requirements and optimize the scalable data quality. To do so, it first sorts the erasure probability $P_{e,ik}$ such that $P_{e,ik} \leq P_{e,ir}$ for any $1 \leq k < r \leq \omega$. Then, we find that the ideal strategy is to allocate a first-UEP-level GEK to the path with the index $k = 1$, such that we only need the path with the lowest erasure probability to recover the most important prefix $\mathbf{P}_1$. In this case, $\rho_{i,1} = 1 - P_{e,i1}$, which is the highest possible $\rho_{i,1}$. Next, we allocate a second-UEP level GEK to the path with next-to-the-lowest erasure probability, i.e., $k = 2$, such that $\rho_{i,2} = (1 - P_{e,i1})(1 - P_{e,i2})$, which is the highest possible $\rho_{i,2}$. We then keep allocating a q^{th} level GEK to the path with the index $k = q$ until reaching the last path [5].

However, as earlier discussed in Section 4.3, this simple allocation scheme may not be possible due to conflicts among sink nodes as well as linear independence and dependence constraints of GEKs. Linear independence constraints ensure that the maximum information flow is achieved at each node, whereas linear dependence represents topological constrains, i.e., the GEK of any outgoing edge of the node i must be linearly dependent on the GEKs of incoming edges [5]. These constraints will be discussed in detail in the next section.

The conflicts among receiving nodes pose an economic problem of optimal distribution of goods, which is discussed countlessly in economic literature. The proper solution depends on the nature of the problem as well as the camp to which the decision maker belongs, i.e., whether he is a socialist, capitalist, or something in between. Although we will not give arguments about economic philosophy in general, we propose two solutions to our GEK assignment problem according to different standards of evaluation. The first one in Section 4.9 is more socialist and the second in Section 4.10 more capitalist.

Before presenting the two solutions in Section 4.9 and 4.10, we explain how to reduce the complexity of the problem in the next section.

4.8 Subtree Decomposition Technique for Complexity Reduction of the GEK Assignment Problem

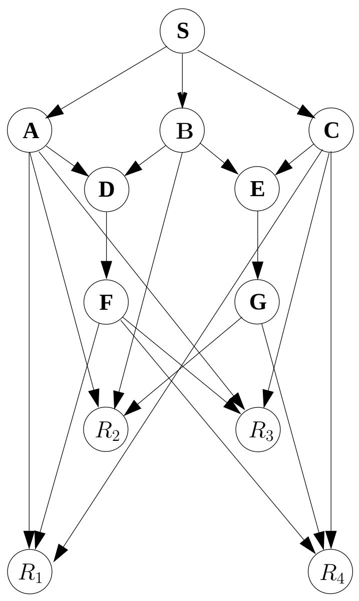

Figure 4.4: Network example with scalable data multicast

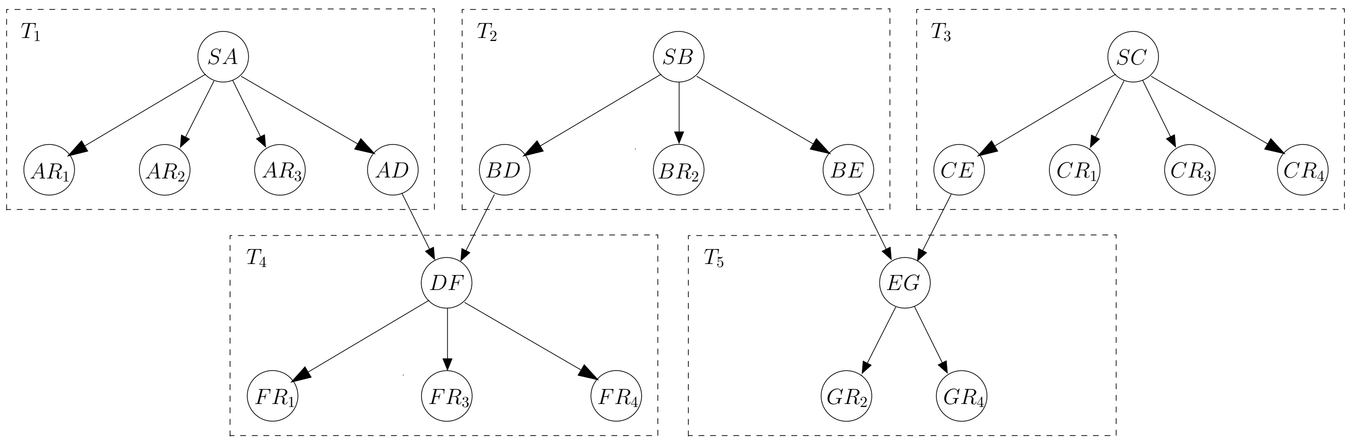

Figure 4.5: Line graph derived from Fig. 4.4

Consider the network in Fig. 4.4. The source S would like to multicast an ordered scalable message $\mathbf{M} = [m_1, m_2, m_3]$, of which elements m_1, m_2, and m_3 have dependency levels of 1, 2, and 3, respectively, to four sink nodes R_1, R_2, R_3, and R_4. Every edge in the graph is capable of transmitting one symbol per time unit.

Before assigning a GEK to each edge, we can simplify the graph in Fig. 4.4, using

Fragouli, Soljanin, and Shokrollahi's approach [14]. Figure 4.4 is first transformed into a line graph shown in Fig. 4.5, where each node represents an edge from Fig. 4.4. Any two nodes in Fig. 4.5 are connected if the corresponding edges in Fig. 4.4 are adjacent.

The nodes in Fig. 4.5 are grouped into five subtrees, each of which is bounded by dashed lines, such that the members in each subtree are forced by the topology to have the same GEK. For example, in the subtree T_1, SA and AD must have the same GEK, since, according to Fig. 4.4, the node A has only one incoming edge SA and thus can do nothing but copy the received symbols and forward the copies to all outgoing edges AR_1, AR_2, AR_3, and AD, hence the same GEK among them.

Accordingly, our problem of assigning GEKs to twenty-one edges is reduced to that of assigning GEKs to five subtrees. The minimum subtree graph is shown in Fig. 4.6.

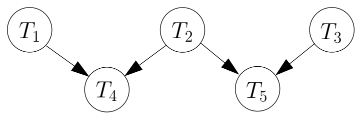

Figure 4.6: The minimum subtree graph derived from Fig. 4.5

In this case, the subtrees $T_1 = \{SA, AD, AR_1, AR_2, AR_3\}$, $T_2 = \{SB, BD, BE, BR_2\}$, $T_3 = \{SC, CE, CR_1, CR_3, CR_4\}$, $T_4 = \{DF, FR_1, FR_3, FR_4\}$, $T_5 = \{EG, GR_2, GR_4\}$.

The edges connecting T_1 and T_2 to T_4, as well as T_2 and T_3 to T_5 in Fig. 4.6 imply that the GEK of T_4 must be derived from those of T_1 and T_2 whereas that of T_5 must be derived from those of T_2 and T_3, i.e., the sets of vectors $\{\mathbf{f}_{T_1}, \mathbf{f}_{T_2}, \mathbf{f}_{T_4}\}$ and $\{\mathbf{f}_{T_2}, \mathbf{f}_{T_3}, \mathbf{f}_{T_5}\}$, where $\mathbf{f}_{T_x}$ denotes the GEK of the subtree T_x, must be linearly dependent.

In a similar manner, linear independence constraints can be represented by the sets $\{\mathbf{f}_{T_1}, \mathbf{f}_{T_2}, \mathbf{f}_{T_3}\}$, $\{\mathbf{f}_{T_1}, \mathbf{f}_{T_3}, \mathbf{f}_{T_4}\}$, $\{\mathbf{f}_{T_1}, \mathbf{f}_{T_2}, \mathbf{f}_{T_5}\}$, and $\{\mathbf{f}_{T_3}, \mathbf{f}_{T_4}, \mathbf{f}_{T_5}\}$. The three GEKs in each set must be linearly independent in order to ensure a full-rank system of linear equations at each of the source and the sinks, when there is no erasure. For example, the source S is connected to the subtrees T_1, T_2, and T_3, hence the linearly independent constraint $\{\mathbf{f}_{T_1}, \mathbf{f}_{T_2}, \mathbf{f}_{T_3}\}$.

The next two sections show how to solve the problem under these constraints with different optimization objectives. In Section 4.9, the objective is equity of received data quality among receiving nodes, whereas that of Section 4.10 is fair competition among the nodes, i.e., the node who pays more receives better quality.

4.9 Minimax Assignment of Global Encoding Kernels (GEKs)

In this section, we propose a strategy for assigning a GEK to each subtree. The ultimate goal is equal data quality for every receiving node. To derive a solution that is as close as possible to that goal, we use a minimax criterion, i.e., minimizing the maximum expected degradation, which is equivalent to maximizing the expected data quality of the poorest sink [5].

Since our example concerns allocating GEKs to five subtrees from a set of seven available GEKs, as shown in Table I, the attempt to find the optimal solution at once is impractical because it results in the search within the space of the size 7^5. Therefore, we may adopt an iterative approach, in which at the iteration t, $n(t)$ GEKs are allocated to $n(t)$ subtrees, resulting in the search within the space of the size $7^{n(t)}$ [5].

The assignment of $n(t)$ GEKs to $n(t)$ subtrees at the t^{th} iteration does not have the same effect on every sink node. An individual sink may either 1) not be connected to any of those subtrees and thus gain nothing more than the previous iteration, or 2) be connected to certain subtrees but still cannot recover any other data than that received in the previous iteration, or 3) be connected to certain subtrees and can recover some more data. In the first and the second cases, the expected data quality of the sink does not improve at the t^{th} iteration, whereas, in the third case, it does.

In each iteration, we aim to ensure that the temporary minimax criterion is satisfied. To do so, we first have to evaluate the temporary expected utility $E_{t-1}[U_i]$ from the previous $t-1^{th}$ iteration for the sink i. After that, we find the minimum $E_t[U_i]$ among all i for every possible GEK assignment at the current t^{th} iteration, and then select the assignment that gives the maximum value of min $E_t[U_i]$ [5].

The temporary expected utility $E_t[U_i]$ of the i^{th} sink at the t^{th} iteration is given by

$$E_t[U_i] = \sum_{j=1}^{\omega} \phi_{t,i,j} \cdot \Delta_j, \tag{4.14}$$

where $\phi_{t,i,j}$ represents the probability that the prefix $\mathbf{P}_j$ can be recovered at the sink i after the GEK assignment at the iteration t. $\phi_{t,i,j}$ equals $\rho_{i,j}$ in Eq. (4.8) if $\mathbf{P}_j$ can be recovered by the GEKs assigned so far up to the iteration t. Otherwise, it equals zero. Δ_j denotes the incremental utility if $\mathbf{P}_j$ is recovered [5].

We suggest that the number $n(t_1)$ of GEKs assigned at the iteration t_1 should be greater than or equal to the number $n(t_2)$ at the iteration t_2 if $t_1 < t_2$, i.e., it is better to take larger search spaces into consideration during some first iterations, after which things do not improve much.

In our network example, we assume that every edge-disjoint path has an erasure probability of 0.1 and the utility vector $\Delta\mathbf{U}_k = [\Delta_1, \Delta_2, \Delta_3] = [1, 0.5, 0.25]$ for every sink node R_k. If we let $n(1) = 3$ and $n(2) = n(3) = 1$, then the minimum temporary expected data quality in the first iteration is maximized by allocating the GEKs $[1\ 0\ 0]^T$ to T_3, $[0\ 1\ 0]^T$ to T_1, and $[1\ 1\ 0]^T$ to T_5. The sink R_1 now has the temporary expected utility of $E_1[U_1] = (0.9)(1) + (0.81)(0.5) = 1.305$, since, by receiving $[1\ 0\ 0]^T$ from T_3 with the probability of 0.9, R_1 can recover the first prefix, and by receiving both $[1\ 0\ 0]^T$ from T_3 and $[0\ 1\ 0]^T$ from T_1 with the probability of 0.81, it can recover the second prefix.

In the same manner, the temporary expected data qualities $E_1[U_2]$, $E_1[U_3]$, and $E_1[U_4]$ of R_2, R_3, and R_4 become 1.305, 1.305, and 1.215, respectively. This gives the minimum temporary expected data quality, min $E_1[U_i]$, of 1.215 at the sink R_4, which is the maximum that one can find at this iteration.

Table 4.2 shows the resulting expected data qualities at all sink nodes and the minimum one, after the third iteration finishes and all subtrees have obtained their GEKs.

Table 4.3 shows that, in our network example, the suggested strategy achieves the global optimum. In addition, on average, a random GEK assignment results in the minimum expected quality that is closer to that of the worst case than the best one. Thus,

Table 4.2: The resulting expected utilities of the recovered scalable data in the network example when the suggested strategy is applied

$E_3[U_1]$	1.4873
$E_3[U_2]$	1.4873
$E_3[U_3]$	1.4873
$E_3[U_4]$	1.3973
min $E_3[U_i]$	1.3973

a significant amount of quality can be saved if we take the UEP mechanism of network codes into consideration and apply the suggested strategy.

Table 4.3: Comparison of minimum expected utilities of the received scalable data in the network example for different strategies

The Best Assignment	The Suggested Strategy	The Average over All Possible Assignments	The Worst Assignment
1.3973	1.3973	1.3055	1.2757

4.10 An Ascending-bid Auction Algorithm for GEK Allocation

Unlike the previous section, the objective of the GEK assignment algorithm in this section is a fair competition among sink nodes. We assume that the source node is an auctioneer and the sinks are participants who bid for the rights to choose GEK allocation. Since we will not give a monopoly to any single sink node, the allocation problem is formulated as a multiple-item auction. For this kind of auction, the dynamic counterpart of Vickrey's effective static design [80] has been proposed since 2004 by Ausubel [39], whose idea will be used in our auction as follows. The source node calls a price, bidders respond with quantities, and the process iterates with increasing price until demand is not greater than supply.

Although the supply is the total number of GEK allotments, it does not equal the total number of edges, because, as discussed earlier, some edges are forced by the topology to have the same GEK. Before the auction begins, the source node has to derive the minimum subtree graph of the network, as explained in Section 4.8 [3].

Let $\mathcal{T}$ represent the set of all subtrees. The source wishes to allocate $|\mathcal{T}|$ GEKs to $|\mathcal{T}|$ subtrees, not on its own but by means of a $|\mathcal{T}|$-item auction. The sink i who wins x_i items is allowed to choose the allocation of x_i GEKs to x_i subtrees in order to maximize its received utility. The source then distributes the data accordingly and informs the intermediate coding nodes about the local encoding functions they have to use [3].

According to Ausubel's idea, a bidder's payment is not the product of its final quantity and the final price. Rather, at the price ψ, the auctioneer sees if the aggregate demand $x_{\setminus i}$ of bidder i's competitiors is less than the supply $|\mathcal{T}|$. If so, $|\mathcal{T}| - x_{\setminus i}$ items are clinched and awarded to the bidder i with the price ψ. Since the winner's payment depends on its competitors bids and not its own bids, every participant has an incentive to reveal truthfully his or her value for the item. [39]

Our auction, however, requires some modification since each clinched item and the resulting GEK allocation of a winning sink affects the demand of others. This is best explained by an example.

Table 4.4: Preference order of GEK allocation
.

Sink	Preference Order		
R_1	T_4	T_3	T_1
R_2	T_2	T_1	T_5
R_3	T_1	T_4	T_3
R_4	T_5	T_4	T_3

Suppose that the source node $\mathcal{S}$ in Fig. 4.4 would like to multicast an ordered scalable message $\mathbf{M} = [m_1, m_2, m_3]$, of which each element m_i has a dependency level of i, to four sink nodes R_1, R_2, R_3, and R_4. We assume that Table 4.4 reflects the preference order of GEK allocation, e.g., if R_1 is allowed to allocate only one GEK, it will choose the best one and assign it to T_4. If two allotments are allowed, R_1 will assign two GEKs to T_4 and T_3. The preference order relates to the erasure probability of each edge-disjoint path discussed in earlier sections. Each sink's main preference is to allocate the GEKs with lower UEP levels to the paths with lower erasure probability. This means, according to Table 4.4, the path from the source that reaches R_1 via T_4 has lower erasure probability than those reaching R_1 via T_3 and T_1.

Let each receiver's estimate of its marginal values θ_1, θ_2, and θ_3, as defined in Definition 4.8 be shown below.

$\boldsymbol{\Theta}$	R_1	R_2	R_3	R_4
θ_1	123	75	85	45
θ_2	113	5	65	25
θ_3	40	3	7	5

Now, let the auction start with the initial price of 10. At this price, R_1 is happy to buy three allotments, while R_2 will buy only one, since the price exceeds the second marginal value. Accordingly, the response from each receiver is shown in the first row of Table 4.5.

Table 4.5: The sinks' responses to the increasing price

.

Sinks	R_1	R_2	R_3	R_4
Responses to 10	3	1	2	2
Responses to 25	3	1	2	1
Responses to 25+ϵ	3	1	1	1
Responses to 40	2	1	1	1

Since nobody clinches anything at this point, the source node raises the price. When the price reaches 25, R_4 has no profit to be gained from the second allotment, and therefore changes its response, as shown in the second row of Table 4.5.

From R_1's perspective, the demand of all other bidders is four, while five allotments are available. If other sinks bid monotonically, R_1 is now guaranteed to win at least one allotment. Thus, according to our rule, R_1 clinches one allotment at the price of 25. It then chooses the first-level GEK to allocate to T_4. This allocation affects the marginal values of every sink except R_1. They are updated as follows.

$\mathbf{\Theta}$	R_1	R_2	R_3	R_4
θ_1	123	75	85	45
θ_2	113	4	6	3
θ_3	40	2	-	-

Due to the allocation of T_4, R_3 and R_4's last entries are removed since they now have only (T_1, T_3) and (T_5, T_3), respectively, to bid for. At the next announced price $25+\epsilon$, R_3's response changes, as shown in the third row of Table 4.5.

Since R_1 is now guaranteed to win two items, it is allowed to allocate one GEK to T_3. After that, the marginal values are updated again, as follows.

$\mathbf{\Theta}$	R_1	R_2	R_3	R_4
θ_1	123	71	62	42
θ_2	113	3	-	-
θ_3	40	1	-	-

At the price of 40, demand equals supply, as shown in the fourth row of Table 4.5, and the market clears. Each of R_2, R_3, and R_4 clinches one object at this price and assigns a GEK to T_2, T_1, and T_5, respectively.

The algorithm implementing the auction at the source node is shown as follows [3].

Algorithm 4.1 *Problem: Allocate a GEK $\mathbf{f}_t \in \mathbb{F}^\omega$ to each $t \in \mathcal{T}$ such that the sets of linear independence and dependence constraints are satisfied.*

1. Initialize the number of available items $N_T = |\mathcal{T}|$, the cumulative clinches $C = 0$, the cumulative quantity $X = 0$, the individual current clinches $\gamma_i = 0$ and individual cumulative clinches $\Gamma_i = 0$ for the node i, $i = 1, 2, ..., n$. Set the current price ψ as the initial price ψ_0. Broadcast linear dependence and independence constraints to all sink nodes.

2. At an appropriate time $t+\Delta t_i$, where Δt_i is the time offset calculated from the distance between the source and the sink i, send the current price ψ to the sink i and wait for response.

3. Upon receiving the quantity x_i from every sink i, update X, Γ_i, and γ_i as follows.

$$X = \sum_{i=1}^{n} x_i \tag{4.15}$$

$$x_{\setminus i} = X - x_i \tag{4.16}$$

$$Y_i = \Gamma_i \tag{4.17}$$

$$Z = C \tag{4.18}$$

$$\Gamma_i = \begin{cases} |\mathcal{T}| - x_{\setminus i} & \text{if } |\mathcal{T}| > x_{\setminus i} \\ 0 & \text{otherwise} \end{cases} \tag{4.19}$$

$$\gamma_i = \Gamma_i - Y_i \tag{4.20}$$

$$C = \sum_{i=1}^{n} \Gamma_i \tag{4.21}$$

4. From 3, if $C - Z > 0$, go to 5. Otherwise, go to 10.

5. Find the index $j \in \mathcal{K}$, $\mathcal{K} = \{\forall k | \gamma_k \neq 0\}$ such that $x_j \geq x_i$, for every $i \neq j$, $i \in \mathcal{K}$. If more than one x_j are found, randomly select one of them.

6. Inform the sink node j about the individual current clinches γ_j. Wait for response.

7. If the response is negative, let $x_j = x_j - 1$ and go back to 3. In case the affirmative response, together with the GEK $\mathbf{f}_r$ that j chooses to allocate to the subtree $r \in \mathcal{T}$, is received, check if the allocation violates the linear dependence or independence constraints. If so, randomly allocate a GEK satisfying the constraints to r. If not, allocate $\mathbf{f}_r$ to r.

8. Broadcast the index of the allocated subtree in 7 and its GEK to all nodes.

9. Let $Z = Z + 1$, $\mathcal{K} = \mathcal{K} - \{j\}$, and $N_T = N_T - 1$. If $N_T > 0$, go back to 4. Otherwise, the algorithm ends.

10. Update the price such that $\psi = \psi + \Delta\psi$. Go back to 2.

In Step 3, the cumulative clinches Γ_i for the sink i is computed as the difference between the supply $|\mathcal{T}|$ and the aggregate demand $x_{\setminus i}$ of its opponents. Y_i is simply a variable used to store the previous Γ_i such that, after Γ_i is updated in (4.19), the individual clinches γ_i at the current price can be computed from (4.20). In a similar manner, the variable Z in (4.18) is used to count the previous cumulative clinches C, which will be updated in (4.21).

In Step 4, if the updated cumulative number of clinches C is greater than the previous one Z, we continue with Steps 5 to 8, which allow one GEK to be allocated. According to Steps 5 and 6, when more than one sink node clinches some objects at a specific price, the nodes which bid for higher quantity are allowed to choose the allocation prior to those bidding for lower quantity.

Step 9 increases Z by 1 and checks whether there are still some items available. If there are, we return to Step 4 to compare the increased Z with C. If C is still greater than Z, we repeat Steps 5 to 9. Otherwise, no more item is clinched at this price and the price is raised in Step 10 [3].

4.11 Conclusion

The proposed auctioning algorithm running at the source has a complexity that increases linearly with the number of sinks n. Since, at every time an item is clinched, each sink node needs to update the marginal values of GEK allocation permissions described in Definition 4.8, the computational complexity at each sink grows linearly with the number of subtrees $|\mathcal{T}|$. The auctioning algorithm has lower complexity than the minimax algorithm in Section 4.9, whose complexity increases exponentially with the number of subtrees.

The distributed nature of the auctioning algorithm offers one clear advantage over a centralized optimization algorithm: If a sink ends up buying an item at an inappropriate price, it has done something wrong in the estimation of marginal values, thus having only itself to blame. However, if the same thing occurs with an algorithm centralized at the source, the source would easily be accused of being unfair.

One disadvantage of the distributed auctioning algorithm is that it requires communications between the source and the sinks which might be difficult if there is a congestion problem. In that case, a sealed bid auction is an alternative to the proposed open auction.

Chapter 5

Network Coding with LT Codes as Erasure Codes

This chapter shows that Luby-transform (LT) encoding at the source node and network coding at intermediate nodes cannot be applied together sequentially in binary erasure channels (BECs) without significant receiver performance degradation due to the distortion of degree distribution in received LT-coded symbols. Two countermeasures are discussed. The first one is wise assignment of network codes, which totally eradicates the distortion in some specific, but not all cases. The second one, being more universal, is a cooperation between the source and the relay nodes. The source introduces buffers for temporarily storing LT-coded output prior to transmission. Each buffer is a first-in-first-out (FIFO) queue associated with one output line from the source node. It is shown that under some conditions related to the degree of a particular LT-coded symbol, it is better to put that symbol into one buffer instead of another in order to keep the degree distribution distortion low. In addition, we paradoxically discover that the relay node, instead of always performing network coding, can improve the receiver performance by discarding some LT-coded symbols with a certain probability, since this will reduce the degree distribution distortion.

5.1 Introduction and the System Model of Relay Network Multicast with LT and Network Codes

LT codes or other rateless codes as a forward error correction (FEC) scheme are arguably most useful in multicast applications in which its one-to-many nature makes the acknowledgement scheme very unpleasant. LT codes can potentially generate an infinite stream of encoded output symbols such that, even when some symbols are erased, the receiver can recover k original symbols using any K received encoded symbols when K is only slightly larger than k.

Like LT codes, network codes find their first and simplest application in multicast. However, both types of codes have opposite aims. While LT codes increase redundancy in the networks to compensate for erasures, network codes decrease it by means of coding at the bottlenecks.

According to Fig. 5.1, where A wants to multicast two binary symbols b_1 and b_2 to D and E, we can see that the node F encodes b_1 and b_2 together to achieve the multicast rate of 2 bits per unit time, if each edge represents the capacity of one bit per unit time. D receives b_1 from the path ABD and can recover b_2 from the symbol $b_1 \oplus b_2$ from $ACFGD$, whereas E receives b_2 from the path ACE and recovers b_1 from the symbol $b_1 \oplus b_2$ from $ABFGD$. Had network coding not been there, only either b_1 or b_2 would have been able to pass the bottleneck FG in one unit time, i.e., one receiver would have been unable to use one of its possible transmission paths.

In this chapter, we assume that the source node generates LT-coded symbols which are subsequently network coded by intermediate nodes along their ways to the receivers. To make things more concrete, Fig. 5.2 displays a block diagram showing all encoding and decoding processes as well as the buffer structure from the source to the destinations in accordance with the network in Fig. 5.1.

One can observe a switch placed after the LT-encoder block and prior to two buffers. While traditional network coding only requires that the switch turns to each buffer half of

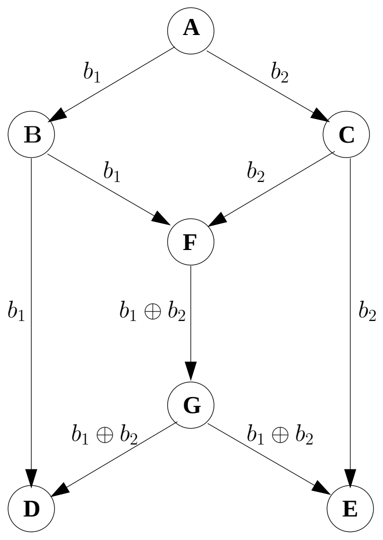

Figure 5.1: Network coding in a butterfly network

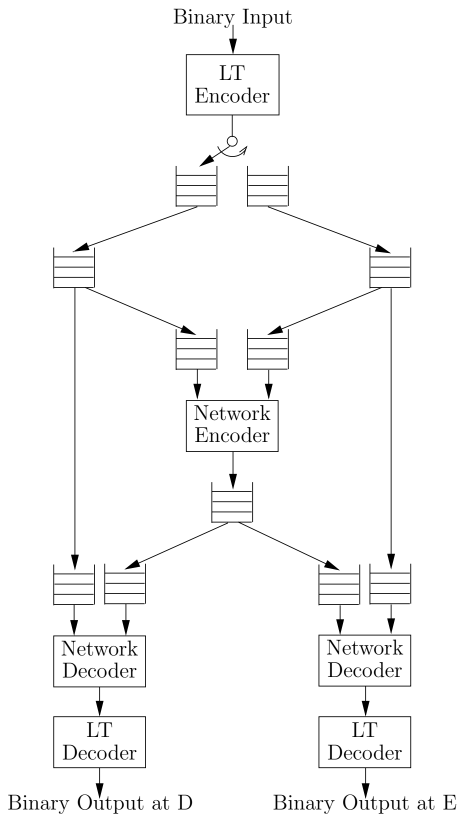

Figure 5.2: Detailed system model including LT-encoding and decoding blocks as well as the buffer structure

the time, it is not the case in this chapter, in which the switching decision depends on the degree of the current LT-encoding output. This is one major discovery in this chapter.

Despite having our own system shown in Fig. 5.2, we are aware of previous works dealing with similar problems. Therefore, we present a short review of them in the next section.

After that, Section 5.3 elaborates on important parts in our system, such as LT encoder and decoder. The most crucial function influencing the LT-decoder performance is the degree distribution which must be carefully chosen by the encoder. Section 5.4, however, shows that the well-designed degree distribution can easily be distorted by network coding in binary erasure channels (BECs), leading to a significant degradation in receiver performance.

Section 5.5 provides a solution to the problem for some special cases by means of wise assignment of network codes. In other cases, however, we need a cooperative scheme proposed in Section 5.6 to improve the receiver performance, as shown in Section 5.7.

5.2 Related Literature

Although many recent works regarding network coding focus on two-way wireless relay networks rather than relay network multicasts [15, 53, 83], it is suggested in [83] that the problem of two-way relay networks, and of information exchange in general, can be transformed into a multicast problem via some graph transformations. Thus, the study of information multicast in this work might lead to further discoveries in more generalized cases.

In addition, main components used in those systems are identical or similar to ours. The decode-and-forward (DF) scheme used in [15] employs a channel encoder at each transmitter and a network encoder at the relay, which is structurally identical to our system. However, we use LT codes instead of Turbo codes and therefore need no decoder at the relay. This is similar to distributed LT codes proposed in [67], but the receiver in that system receives information only from the relay, whereas our sinks receive it via direct paths as well. Since LT codes are erasure codes, binary erasure channels (BECs) are used in our model instead of Gaussian channels.

Like in our system, the buffering scheme is considered important and given careful attention in [66, 83].

5.3 LT and Network Encoding and Decoding

Three steps are needed to generate an LT-coded symbol. Firstly, a degree d is selected from a degree distribution, which is a discrete probability density function mapping a degree to the probability that the degree is selected. Secondly, d input symbols are chosen uniformly at random. Finally, an output symbol is derived by XORing all chosen symbols from the previous step [44].

In this thesis, we use a robust soliton distribution, which is constructed such that the failure probability of the message-passing decoder is δ for a given number $K = k + O(\sqrt{k} \cdot \ln^2(k/\delta))$ of received symbols [44].

Definition 5.1 *The robust soliton distribution (RSD) $\mu(i)$ is derived from the normalization of two functions $\rho(i)$ and $\tau(i)$ as*

$$\mu(i) = \frac{\rho(i) + \tau(i)}{\beta}, \; 1 \leq i \leq k, \tag{5.1}$$

where

$$\rho(i) = \begin{cases} 1/k, & i = 1 \\ 1/(i(i-1)), & 2 \leq i \leq k, \end{cases} \tag{5.2}$$

$$\tau(i) = \begin{cases} R/(i \cdot k), & 1 \leq i \leq (round(k/R)) - 1, \\ (R \ln(R/\delta))/k, & i = k/R, \\ 0, & otherwise, \end{cases} \tag{5.3}$$

$$R = c \cdot \sqrt{k} \cdot \ln(k/\delta), \tag{5.4}$$

$$\beta = \sum_{i=1}^{k} (\rho(i) + \tau(i)). \tag{5.5}$$

The paramater c in (7.4) is a suitable non-negative constant used in the design, whereas δ is the failure probability mentioned earlier [44].

The robust soliton distribution with parameters $k = 1000$, $c = 0.1$, and $\delta = 0.5$ is shown in Fig. 3. We choose these parameters according to [67]. The most important observation regarding the plot is that it has two peaks at 2 and 42.

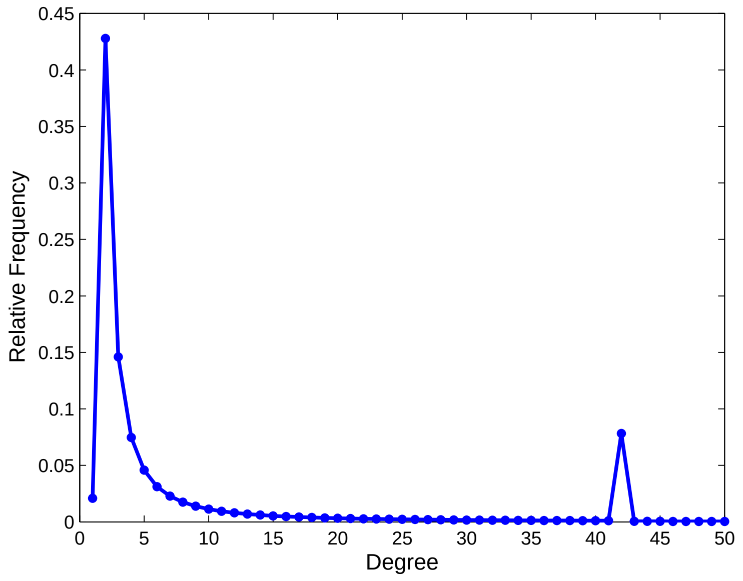

Figure 5.3: Robust soliton distribution with $k = 1000$, $c = 0.1$, and $\delta = 0.5$

Now, let us consider the network in Fig. 5.1. Assume that node A would like to multicast LT-coded outputs b_1 and b_2 with the robust soliton distribution shown in Fig. 5.3

to D and E. After the node D receives b_1 and $b_1 + b_2$, it first deducts b_1 from $b_1 + b_2$ to obtain b_2. We call this deduction "network decoding." After that, both b_1 and b_2 are LT-decoded by the message-passing algorithm, as described in [44], to obtain the original message.

Unfortunately, when network coding is used in severe erasure channels, the well-designed degree distribution can be distorted before LT-coded symbols reach the destination. The next section addresses this issue as well as the resulting receiver performance deterioration.

5.4 Degree Distribution Distortion and Receiver Performance Deterioration in Binary Erasure Channels

Suppose that the edge BD is a binary erasure channel having an erasure probability of 0.1 whereas other edges are erasure-free. Thus, on average, once every ten times the receiver D does not have b_1 to deduct from $b_1 + b_2$ to complete the network decoding process. Instead, $b_1 + b_2$ enters directly into the LT decoding process. Therefore, on average, for every ten symbols of b_1 and ten of b_2 transmitted, D receives nine b_1, nine b_2 (after network decoding), and one $b_1 + b_2$ (which cannot be network-decoded). If b_1 and b_2 follow the robust soliton distribution $\mu(i)$, the overall degree distribution becomes

$$\psi(i) = \frac{18}{19}\mu(i) + \frac{1}{19}\varphi(i), \tag{5.6}$$

where $\varphi(i)$ is the degree distribution of $b_1 + b_2$.

It is precisely the $\varphi(i)$ that alters the overall degree distribution of D's received symbols from the robust soliton case. Since, in practice, the number k of LT input symbols is large, we can assume that the LT-encoded symbols b_1 and b_2 are not made up of some common original symbols. This allows us to approximate the overall degree distribution at D after network decoding as

$$\psi(i) \approx \begin{cases} \frac{18}{19}\mu(i), \ i = 1 \\ \frac{18}{19}\mu(i) + \frac{1}{19}\sum_{j=1}^{i-1}\mu(j)\mu(i-j), \ \text{otherwise.} \end{cases} \tag{5.7}$$

Figures 5.4 and 5.5 compare the plot of the robust soliton distribution $\mu(i)$ in the erasure-free case with the distorted distribution due to erasures at BD obtained by the analytical approximation $\psi(i)$ in (5.7) and that obtained from simulation by counting the degree of 10^7 symbols received at D around the two peaks at 2 and 42, respectively. We can see that the steepness of the peaks is reduced in the distorted case. Moreover, an unwanted peak is formed at 44.

Although the distortion of the degree distribution is small, its effect on the performance is clearly visible. Figure 5.6 shows the histogram of the number of LT-coded symbols needed to be transmitted until the original symbols can be recovered by D. Figure 5.7 makes a comparison between the number of symbols needed to be transmitted when the erasure probability of BD is 0.1 in the network in Fig. 5.1 and that when the erasure probability is 0.05 in normal point-to-point communication. Although the average erasure probabilities in both cases are the same, the network coding case requires more symbols due to the distortion of the degree distribution.

5.5 The First Solution: Wise Assignment of Network Codes

In case erasures occur only along the edge BD with the erasure probability of 0.1, the solution to the degree distribution distortion problem is very simple. We change our network codes from those in Fig. 5.1 into those in Fig. 5.8. Since we now transmit $b_1 + b_2$ instead of b_1 along BD, when $b_1 + b_2$ is erased, D only receives b_1, yielding no distortion, when $b_1 + b_2$ is not erased, D can network-decode and receive both b_1 and b_2, yielding no distortion either.

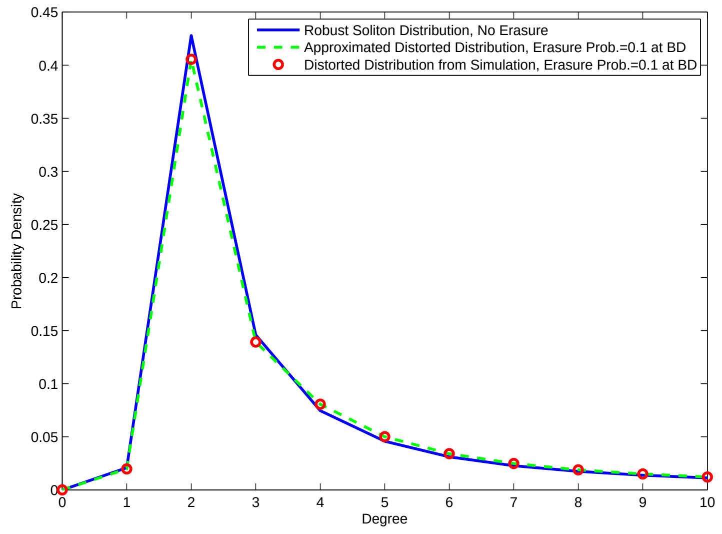

Figure 5.4: Robust soliton distribution around the first peak in comparison with distorted distribution caused by erasures at the edge BD

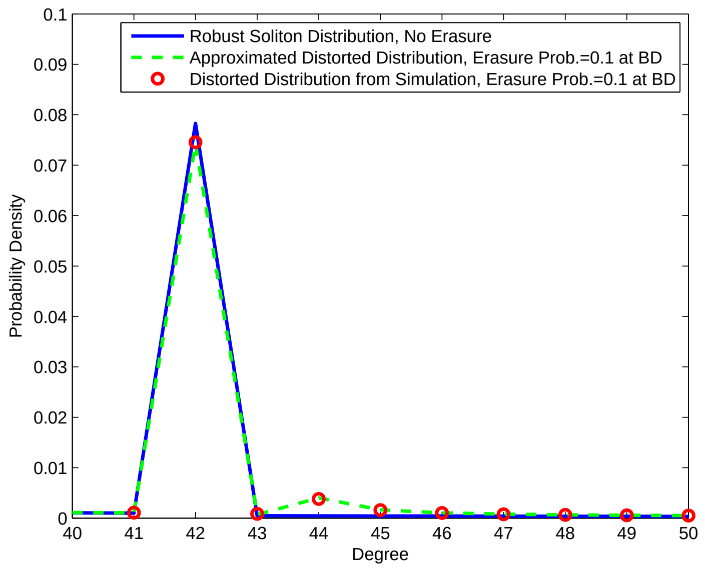

Figure 5.5: Robust soliton distribution around the second peak in comparison with distorted distribution caused by erasures at the edge BD

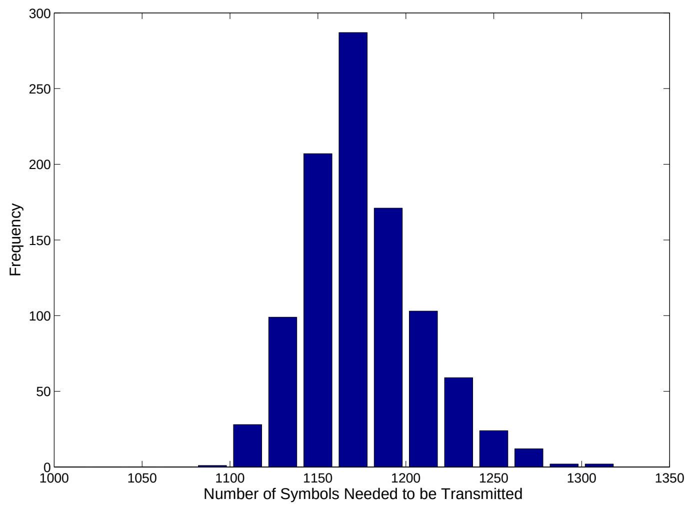

Figure 5.6: Histogram of the number of LT-encoded symbols needed to be transmitted in an erasure-free case such that all original symbols are recovered

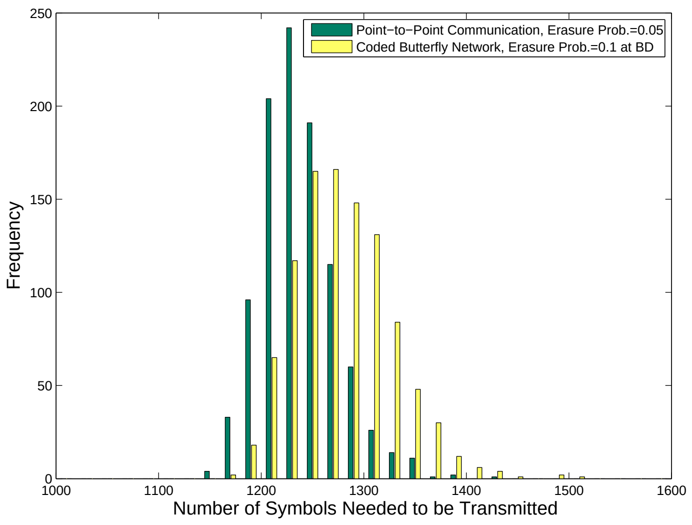

Figure 5.7: Comparison of histograms of the number of LT-encoded symbols needed to be transmitted such that all original symbols are recovered in two cases, a point-to-point communication with an erasure probability of 0.05 and a coded butterfly network with an erasure probability of 0.1 at BD

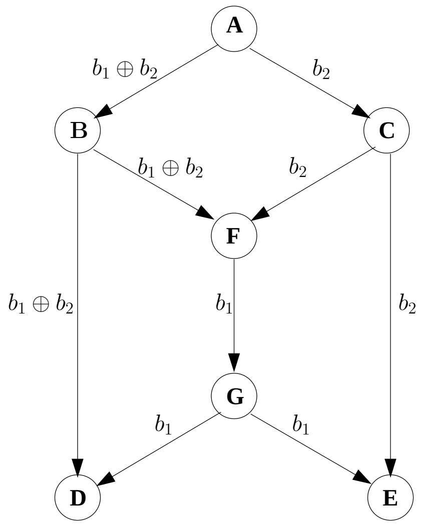

Figure 5.8: Network coding in a butterfly network when only BD is erasure-prone

This solution can be performed by the source alone. When the source A is informed by B that there are severe erasures at BD, it simply transmits $b_1 + b_2$ instead of b_1 to B while other nodes just work as usual. In general, this solution is applicable when erasures occur along a single edge, with the transmission pattern depending on which edge is erasure-prone, as shown in Table 5.1.

Table 5.1: Recommended Transmission When An Edge is Erasure-prone

Erasure-prone edge	Choices of recommended transmission to (AB, AC)
AB or BF or GE	(b_1, b_2), (b_2, b_1), $(b_1 + b_2, b_1)$, $(b_1 + b_2, b_2)$
AC or CF or GD	(b_1, b_2), (b_2, b_1), $(b_1, b_1 + b_2)$, $(b_2, b_1 + b_2)$
BD	$(b_1 + b_2, b_1)$, $(b_1 + b_2, b_2)$
CE	$(b_1, b_1 + b_2)$, $(b_2, b_1 + b_2)$
FG	(b_1, b_2), (b_2, b_1)

The next section deals with a more general case in which more than one edge is erasure-prone.

5.6 The Cooperative Solution Performed by the Source and the Relay

As an example, consider the case when both the edges BD and CE are erasure-prone. In this case, we cannot find a solution from the previous section that satisfies both receivers. If A chooses to transmit $(b_1 + b_2, b_2)$ to (AB, AC), the receiver D is satisfied but E still suffers from the degree distribution distortion.

It can be imagined that the ideal solution can be found such that, instead of transmitting b_1 and b_2 with the robust soliton distribution, we use another degree distribution $\nu(i)$ which, after being distorted by erasures, becomes the robust soliton distribution. However, such a distribution is very difficult, if not impossible, to be found. Indeed, it is proven in a similar work [67] that one cannot find a degree distribution $\nu(i)$ for b_1 and b_2 such that $b_1 + b_2$ follows the robust soliton distribution.

Since we cannot achieve this ideal solution, we will offer a cooperative scheme performed by the source node A and the relay node F that corrects the degree distribution distortion only in the positions that most affect the performance. Those positions are at degrees 2, 4, 44, and 84.

The severe distortion at 4, 44, and 84 is due to the high probability that b_1 and b_2 have the degrees of 2 or 42, causing $b_1 + b_2$ to have the degree of 4, 44, or 84 with higher probability than what is required, whereas the distortion at 2 is due to the fact that $b_1 + b_2$ cannot have the degree 2 unless both b_1 and b_2 have the degree of 1, which is unlikely.

According to our cooperative scheme, the distortion at 44 and 84 is reduced by applying a buffering scheme at the source. The scheme arranges the LT-encoder output in such an order that b_1 and b_2 with degrees of 2 and 42, 42 and 2, or 42 and 42 are not allowed to be simultaneously transmitted and mixed at the relay thereafter, as implemented in Step 4) of Algorithm 5.1. In addition, the distortion at 2 and 4 is corrected by selectively discarding some symbols at the relay, i.e., when both b_1 and b_2 have the degree of 2, there is a probability p_c that the relay performs network coding and a probability $1 - p_c$ that either b_1 or b_2 is transmitted while the other is discarded. This will relieve the excess of symbols with degree 4 and the shortage of those with degree 2. The scheme is implemented in Algorithm 5.2.

Algorithm 5.1 *The buffering scheme performed by the source*

Let

- $S_l = \begin{cases} 1 \textit{ if the switch is pushed to the left buffer} \\ 0 \textit{ if the switch is pushed to the right buffer} \end{cases}$,
- λ_n *be the LT-encoder output at the discrete time* $n = 0, 1, 2, ..., N_{max}$,
- $d(\lambda_n)$ *be the degree of* λ_n,
- N_B *be the size of each buffer,*
- $B_l(m), m = 0, 1, 2, ..., N_B - 1$ *be the* m^{th} *element in the left buffer such that* $B_l(m)$ *precedes* $B_l(m+1)$,
- $B_r(q), q = 0, 1, 2, ..., N_B - 1$ *be the* q^{th} *element in the right buffer such that* $B_r(q)$ *precedes* $B_r(q+1)$,
- π_l, π_r *be the pointers of the left and the right buffer, respectively,*
- *and* b_1, b_2 *be the current symbol to be transmitted to the edges* AB *and* AC, *respectively,*

```
1) Initialize
    $n := 0$
    $S_l := 1$
    $B_l(0) := \lambda_0$
    $\pi_l := 1$
    $\pi_r := 0$
    $\{B_l(m) | m = 1, 2, ..., N_B - 1\} := \emptyset$
    $\{B_r(q) | q = 0, 1, 2, ..., N_B - 1\} := \emptyset$
2) $n := n + 1$
3) if $S_l = 0$ {
        $S_l := 1$
        *if* $\pi_l < N_B$ {
            $B_l(\pi_l) := \lambda_n$
            $\pi_l := \pi_l + 1$ }}
*4) while* $S_l = 1$ {
        *if* $(d(\lambda_n), d(B_l(\pi_r))) \notin \{(2,42),(42,2),(42,42)\}$ {
            $S_l := 0$
            *if* $\pi_r < N_B$ {
                $B_r(\pi_r) := \lambda_n$
                $\pi_r := \pi_r + 1$ }}
        *else* {
            *if* $\pi_l < N_B$ {
                $B_l(\pi_l) := \lambda_n$
                $\pi_l := \pi_l + 1$ }}}
*5) if the channel access is allowed* {
        $b_1 := B_l(0)$
        $b_2 := B_r(0)$
        $B_l(m) := B_l(m+1),\ m = 0,1,2,...,N_B - 1$
        $B_r(q) := B_r(q+1),\ q = 0,1,2,...,N_B - 1$
        $\pi_l := \pi_l - 1$
        $\pi_r := \pi_r - 1$
        *Transmit* $b_1$ *and* $b_2$. }
*6) if* $n < N_{max} - 1$ {
        *Go back to 2).* }
    *else* {
        *Exit.* }
```

Algorithm 5.2 *The discarding scheme performed by the relay*

Let

- b_1, b_2 *be the current symbol received by* F *from the edges* BF *and* CF, *respectively,*

- *$d(b_n)$ be the degree of b_n, $n = 1, 2$*
- *and $p_c \in [0, 1]$ be a design parameter used as the probability that F performs network coding.*

```
1) if $(d(b_1), d(b_2)) \neq (2, 2)$ {
        Change the header and transmit $b_1 \oplus b_2$ }.
    else {
        Generate a uniformly distributed random number $r_1$ in the range [0,1].
        if $r_1 < p_c$ {
            Change the header and transmit $b_1 \oplus b_2$ }.
        else {
            Generate a uniformly distributed random number $r_2$ from the set $\{0, 1\}$.
            if $r_2 = 0$ {
                Transmit $b_1$ }
            else {
                Transmit $b_2$ }}}
2) Repeat 1) when new $b_1$ and $b_2$ arrive.
```

From Step 3) in Algorithm 5.1, we can see that when the previous switch position is at the right buffer ($S_l = 0$), it will always be turned to the left one ($S_l = 1$) as the current symbol arrives. On the other hand, in Step 4), the switch position will only change from left to right only if this does not create the degree distribution pair we aim to avoid.

Note that the information regarding the linear combination of original symbols for each LT-coded output is contained in the header. Therefore, in Algorithm 5.2, the header must be changed if we transmit $b_1 \oplus b_2$.

5.7 Results, Conclusions, and Future Works

Figure 5.9 compares the histograms of the number of symbols needed to be transmitted by the source such that all original symbols are recovered in two cases, without the proposed algorithm and with Algorithm 5.1. We can see that Algorithm 5.1 reduces the variance of the number of required symbols, thus making the histogram of the latter case more concentrated near the middle at 1270. When both algorithms are applied, as shown in Fig. 5.10, the histogram resembles a left-shifted version of that in Fig. 5.9 when only Algorithm 5.1 is applied. Instead of simple histograms, Figure 5.11 plots normalized cumulative histograms to clearly show that less symbols are needed to be transmitted when both algorithms are applied. We can conclude that our cooperative scheme improves the receiving performance, not only when the erasure probability at BD is 0.1, but also when it is 0.2 and 0.05. The larger the erasure probability, the greater the improvement.

In a more complicated network, a node can act as both a relay for upstream nodes and a source for downstream ones. Therefore, the subtree analysis as suggested in [14] is needed to identify whether a given node's buffering scheme should follow Algorithm 5.1, Algorithm 5.2, or a modified algorithm combining both of them. In addition, the relationship between erasure probabilities in all edges and the suitable parameter p_c used in Algorithm 5.2 should be studied further. In this work, the value of p_c is 0.7492.

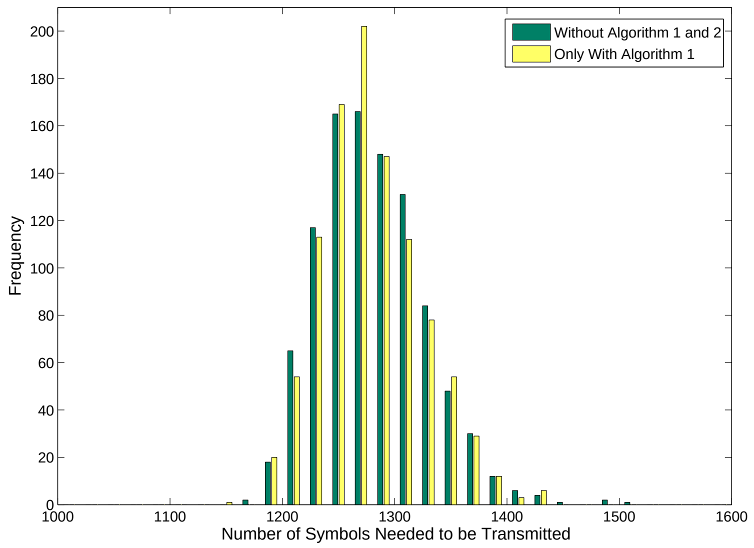

Figure 5.9: Comparison of histograms of the number of LT-encoded symbols needed to be transmitted in a coded butterfly network with the erasure probability of 0.1 at BD such that all original symbols are recovered in two cases, without using the proposed algorithm and with Algorithm 5.1

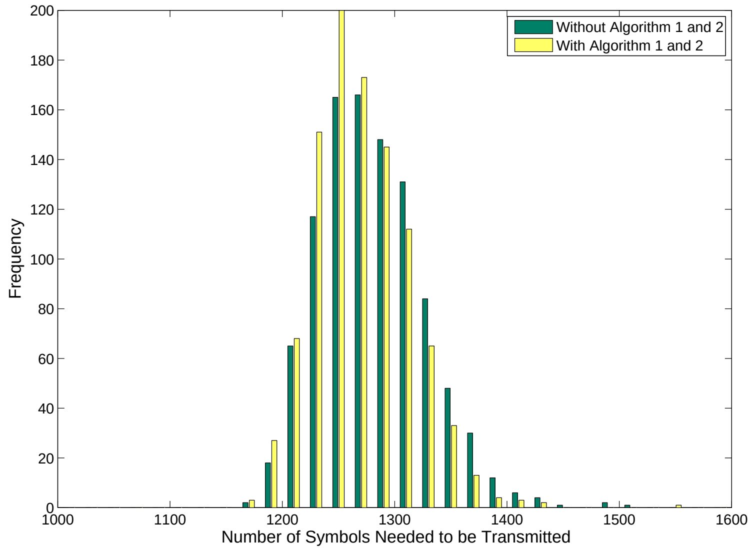

Figure 5.10: Comparison of histograms of the number of LT-encoded symbols needed to be transmitted in a coded butterfly network with the erasure probability of 0.1 at BD such that all original symbols are recovered in two cases, without using the proposed algorithm and with Algorithms 5.1 and 5.2

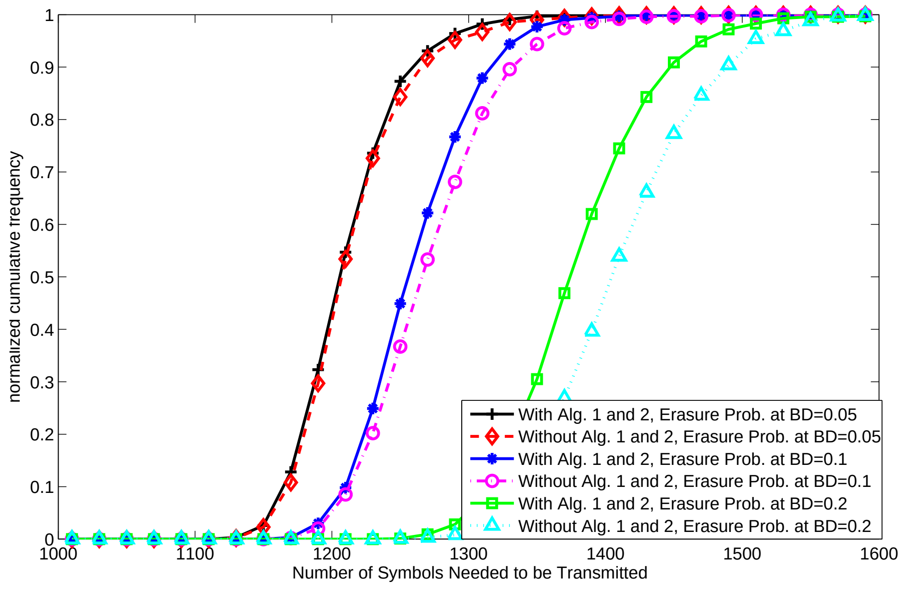

Figure 5.11: Comparison of normalized cumulative histograms of the number of LT-encoded symbols needed to be transmitted in a coded butterfly network with varying erasure probability at BD such that all original symbols are recovered in two cases, without using the proposed algorithm and with Algorithms 5.1 and 5.2

Part III

Wireless Physical-layer Secret-key Generation (WPSG) and Network Coding Techniques for Security Enhancement

Contained in this part are two network-coding-related ideas used for security enhancement of wireless physical-layer secret-key generation (WPSG). The first one employs network-coding-like information mixing in the security protocol for pilot symbol transmission in WPSG. The second one adopts the secret sharing concept used in secure network coding as the basis of what we call "physical-layer key encoding." With this encoding, the eavesdropper may correctly estimate some key symbols, yet knows nothing about the secret message. These two points are discussed in Chapter 6 and 7, respectively.

Some other aspects of WPSG are also discussed. In Chapter 6, an information-theoretic analysis of key generation, key extension in relay networks, and some other security protocols are presented. In Chapter 7, the scalable security concept is formally derived and applied to WPSG.

Chapter 8 summarizes all topics discussed so far and gives some ideas for further research.

Chapter 6

Wireless Physical-layer Secret-key Generation (WPSG) in Relay Networks: Information Theoretic Limits, Key Extension, and Security Protocol

A physical-layer security scheme based on mutual channel-state information (CSI) [8], [71] is discussed in this chapter. Before describing the recent development in our own research, we find ourselves obliged to begin with the basic components of cryptosystems in general. Section 6.1 discusses some general ideas and concepts in cryptology and introduces the cryptosystem model of wireless physical-layer secret-key generation (WPSG). Section 6.2 gives some information-theoretic limits relevant to the key generation process in the case of direct communication without relays as well as some simulation results. This includes the consideration regarding the possibility that the enemy cryptanalyst can estimate some parts of the key. After that, Section 6.3 extends our analysis regarding the amount of generated key further to the cases in which relay nodes are located between transmitter and receiver. Then, Section 6.4 investigates the rate of key generation.

Empirical results from [33, 34] show that, for direct communication, the ratio of the number of vulnerable key bits to the total number of generated key bits is low when there are many wireless transmission paths and the eavesdropper is not within the distance of one wavelength from the transmitter or the receiver. However, as described in Section 6.5, this ratio increases in relay communication. To keep the number of vulnerable key bits as low as possible, some security protocols for relay communication are given in that section.

6.1 Cryptosystem of Wireless Physical-layer Secret-key Generation (WPSG)

Massey [31] suggests a model of a secret-key cryptosystem as shown in Fig. 6.1. Its concept of security is based on Shannon's idea of perfect secrecy, which means that, for a plain text $\mathbf{X}$ and its cryptogram $\mathbf{Y}$, $P(\mathbf{X}=\mathbf{x}|\mathbf{Y}=\mathbf{y}) = P(\mathbf{X}=\mathbf{x})$ for all possible plain texts $\mathbf{x} = [x_1, x_2, ..., x_M]$ and cryptograms $\mathbf{y}$. In this case, neither the knowledge of the cryptogram $\mathbf{y}$ nor large computational power can help an enemy cryptanalyst to decrypt the message $\mathbf{x}$, unless he or she knows the secret key.

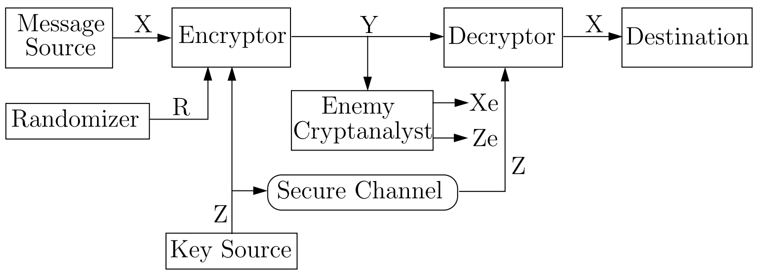

Figure 6.1: A secret-key cryptosystem

In Massey's generalized model, the encryptor mixes the plain text $\mathbf{X}$ with the random message $\mathbf{R}$ and the secret key $\mathbf{Z}$ to achieve perfect secrecy. However, some systems, such as the "one-time pad," which will be discussed in the next chapter, do not require $\mathbf{R}$. In such cryptosystems, the major implementation difficulty lies in the secure distribution of the key.

The scope of this part is on the new idea of transforming wireless channel state information (CSI) into a secret key, which will be called wireless physical-layer secret-key generation (WPSG) from now on. It is widely known that wireless channel coefficients, characterized by their phases and amplitudes, depend heavily on the location, the environment, and the movement of transmitter and receiver to the extent that other terminals except the two can predict almost nothing about their channel parameters, and hence their secret key. The generation of the secret key consists of two steps, deriving the channel estimates before quantizing them into secure key symbols. It is important to distinguish WPSG from the term "physical layer security" used in [42], which requires that the legitimate users' channels have SNR advantages over those of eavesdroppers.

Channel estimates can be derived using a known pilot sequence, which is transmitted back and forth between transmitter and receiver such that they can learn about channel coefficients from the symbols distorted by the channel. The outcome of the channel estimation process is a set of complex channel coefficients which must be quantized into secret key symbols, as shown in Fig. 6.2. We can see that the secure channel in Fig. 6.1 is implemented by the key-generating channel in Fig. 6.2. The key source in Fig. 6.1 is implemented by the combination of a channel estimator and a quantizer in Fig. 6.2. The randomizer in Fig. 6.2 is shown by dashed lines to indicate that it may be needed or not according to the encryptor used. The process of key generation is discussed in more detail in [8, 34].

6.2 Information-Theoretic Limits of the Key Generation

This section studies the amount of key symbols expected to be generated from wireless channels. When there is only one antenna, each for the transmitter and the receiver, the channel is described as single-input single-output (SISO) or a scalar channel. When

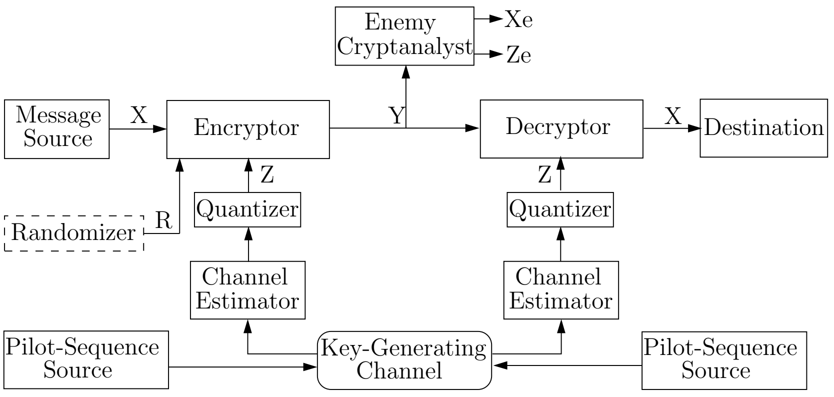

Figure 6.2: A secret-key cryptosystem based on mutual CSI

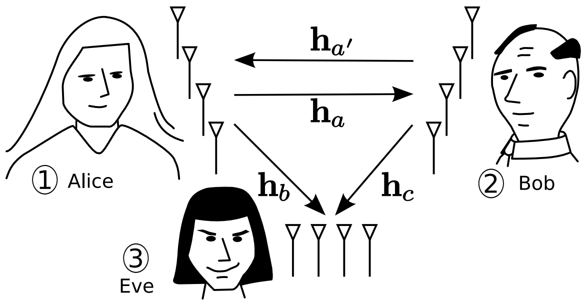

Figure 6.3: A generic communication system model

both the transmitter and the receiver have more than one antenna, secret keys can be generated from more than one channel. These channels are said to be multiple-input multiple-output (MIMO) or vector channels. A scalar channel can be considered as a special case of vector channels.

Now let us consider a communication system model in Fig. 6.3. In this case, there are three parties involved. Alice and Bob are a legitimate transmitter-receiver pair who generate secret keys for their secure communication from reciprocal channel vectors $\mathbf{h}_a$ and $\mathbf{h}_{a'}$, whereas Eve is an eavesdropper who tries to predict the secret key generated by Alice and Bob by using information from channel vectors $\mathbf{h}_b$ and $\mathbf{h}_c$. When Eve is closely located to either Alice or Bob, there might be a correlation between $\mathbf{h}_c$ and $\mathbf{h}_{a'}$ or between $\mathbf{h}_b$ and $\mathbf{h}_a$, respectively, enabling her to estimate some key. According to [33, 34], when there are many scatterers, e.g., if the number of wireless paths between each transmit-receive antenna pair is more than 10, the referred correlation is negligible when the distance between Eve and Alice or Bob is larger than one wavelength. However, when there is only one or two paths, the correlation is significantly high.

In [34], two rigorous metrics have been developed for quantifying the information theoretic limits of key generation in the scenario depicted in Fig. 6.3. Given the users have noisy channel estimates, which are denoted by $\hat{h}_a$, $\hat{h}_{a'}$, $\hat{h}_b$, and $\hat{h}_c$, we refer to the maximum number of independent key bits that can be generated per channel observation as $I_K = I(h_a; h_{a'})$, where $I(x; y)$ denotes the mutual information between x and y. Likewise, the maximum number of independent key bits that can be generated and are secure from an eavesdropper is given by $I_{SK} = I(\hat{h}_a, \hat{h}_{a'}|\hat{h}_b, \hat{h}_c)$ [76]. In [34], closed-form expressions for I_K and I_{SK} are derived for correlated complex Gaussian vector channels. The generated key bits that are not secure are called vulnerable key bits. The number of vulnerable key bits is therefore $I_{VK} = I_K - I_{SK}$.

To illustrate the key generation in Fig. 6.3, consider a simple scenario with scalar channels. Alice sends a pilot signal x to Bob, who derives the channel estimate $\hat{h}_a$ of the channel h_a from the received signal $y_b = h_a x + n_b$, where n_b denotes complex Gaussian noise on Bob's side. After that, Bob sends x to Alice, who derives the estimate $\hat{h}_{a'}$ of $h_{a'}$ in a similar manner. The number of available key bits I_K per channel observation that can be generated by Alice and Bob is given by $I_K = I(\hat{h}_a; \hat{h}_{a'})$. If the time between Alice's transmission and Bob's is short and the same frequency band is used, we can assume reciprocity or $h_a = h_{a'}$. Assuming further that h_a is Rayleigh-distributed with a standard deviation of 0.5, we obtain a simulation result in Fig. 6.4 showing the relationship between I_K and the signal-to-noise ratio, which is the ratio of the power of x to the Gaussian noise power at Bob and Alice. We also compute the result when the key is derived from the envelopes of channel parameters only, such that $I_K = I(|\hat{h}_a|; |\hat{h}_{a'}|)$ [76].

6.3 Possible Key Extensions

In a general wireless network, the length of the key can be extended by having it generated from several transmission routes instead of only one. Although it is obvious that the total key length is the summation of the lengths from all paths, the key length from each path

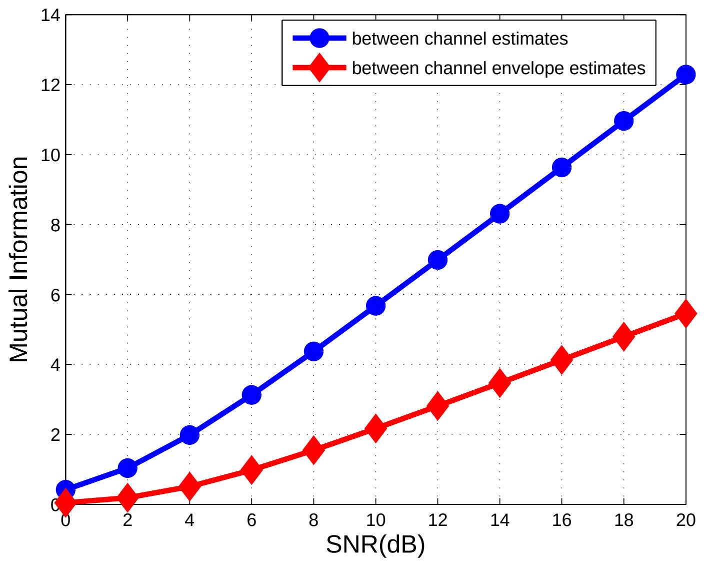

Figure 6.4: A simulation result showing the mutual information between the channel estimates of Alice and Bob, and between their channel envelope estimates

is not easily derived when there are relays in between. For instance, if there is one relay in the path, as illustrated in Fig. 6.5, the channel parameter h_a between the transmitter and the receiver becomes $h_{aT} = h_{a1} \cdot h_{a2}$, which is the product of two complex Gaussian random variables. If we let the real parts of h_{aT}, h_{a1}, and h_{a2} be h_{aTr}, h_{a1r}, and h_{a2r}, respectively, and the imaginary parts be h_{aTi}, h_{a1i}, and h_{a2i}, respectively, we have

$$h_{aTi} = h_{a1i}h_{a2r} + h_{a1r}h_{a2i}. \tag{6.1}$$

If we denote the random variables representing the random processes that generate h_{aTi}, $h_{a1i}h_{a2r}$, and $h_{a1r}h_{a2i}$ by Z, X, and Y respectively, the probability density function $f_Z(z)$ will be the convolution of $f_X(x)$ and $f_Y(y)$.

$$f_Z(z) = \int_{-\infty}^{\infty} f_X(z-y)f_Y(y)dy \ . \tag{6.2}$$

In our case, X and Y have the same probability density function

$$f_X(x) = f_Y(x) = 4\frac{K_0(4x)}{\pi} \ , \tag{6.3}$$

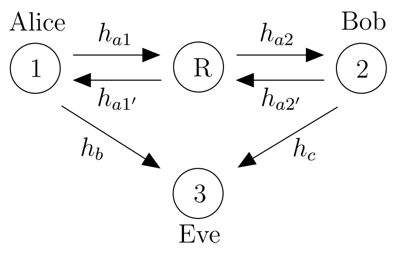

Figure 6.5: A communication system model with one relay

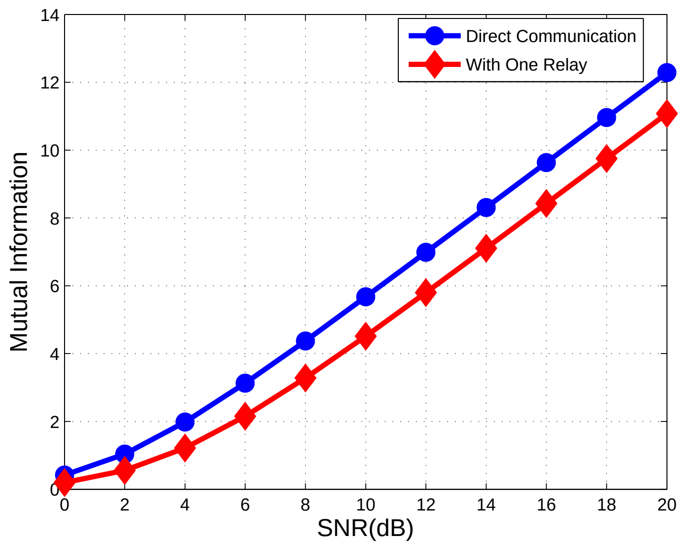

Figure 6.6: A simulation result showing the mutual information between the channel estimates of Alice and Bob in two cases, direct communication and communication with one relay

where $K_0(x) = \int_0^\infty \frac{\cos(xt)}{\sqrt{t^2+1}}dt$ is a modified Bessel function of the second kind [20]. From eqs. (6.2) and (6.3), we can derive the distribution of the imaginary part, which is the same as that of the real one, of channel parameters in the one-relay case [76]. Our simulation result in Fig. 6.6 shows that, due to the altered distribution of channel parameters, the number of possible key bits in the one-relay case is less than the direct communication case. Approximately, it takes 2 dB more pilot transmission power in the one-relay case in order to generate the same amount of key bits as in the direct communication case. Note that this is just the comparison between two exemplary cases. The mobility effect on key generation rate is taken into account in the next section.

6.4 Investigation of the Key Generation Rate

In a direct communication, if the transmitter and the receiver have not moved, no new key can be generated since the channel is static. Therefore, the key generation rate depends heavily on the movements of transmitter and receiver. Assuming that channel parameters are sampled once every sampling time t_s, the average key generation rate becomes

$$R_k = \lim_{n\to\infty} \frac{\sum_{i=0}^{n-1} I_k(it_s)}{nt_s} \ , \tag{6.4}$$

where

$$I_k(it_s) = I\left(\hat{h}_a(it_s); \hat{h}_{a'}(it_s) \,\middle|\, \hat{h}_a((i-1)t_s), \hat{h}_{a'}((i-1)t_s), ..., \hat{h}_a(0), \hat{h}_{a'}(0)\right). \tag{6.5}$$

The mutual information in (6.5) depends on how far the transmitter and the receiver can travel within t_s. It is therefore related to the velocity and the mobility model if the value is to be obtained by simulation [76].

Note that, when the key is generated according to the one-relay model in Fig. 6.5, the key generation rate is improved by the relay movement, which causes variation in channel parameters even when both transmitter and receiver are static. This compensates for the key loss due to the altered distribution shown in Fig. 6.6.

6.5 Security Protocol

Wireless communication between two nodes in a multi-path, multi-hop network does not require that the forward transmission path and the reverse one are the same. However, if our wireless physical-layer security scheme is used, they at least have to make sure that they use the same channels for secret-key generation. Some protocols are discussed in this section to achieve that purpose.

In an arbitrary network having some routers between Alice and Bob, we need a secure protocol for the transmission of pilot symbols for the key generation. To differentiate

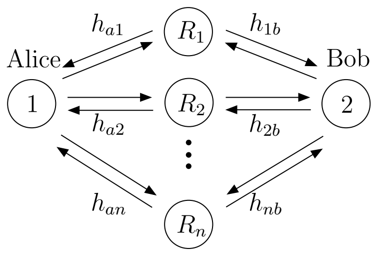

Figure 6.7: A fork-rake network

secure protocols from insecure ones, let us consider a simplified fork-rake network in Fig. 6.7 as an example. Alice broadcasts a pilot packet x to all relays R_1 to R_n within her transmission range. Each relay R_i receives $h_{ai}x$, where h_{ai} is the channel gain between Alice and the relay and x is the pilot sequence. Using an amplify-and-forward scheme, each relay R_i chooses whether to forward $a_i h_{ai} x$, where a_i is the amplification gain to Bob, or not. Then, Bob receives $\sum_{i\in\mathcal{F}} a_i h_{ai} h_{ib} x$, where h_{ib} is the channel gain between the relay R_i and Bob, whereas $\mathcal{F}$ is the index set of relays that choose to forward the pilot packet. After that, assuming that the channel gain has not changed yet, Bob derives the key and broadcasts x back such that every relay receives $h_{ib}x$. However, only the relay R_i that has previously forwarded the packet to Bob will forward $a_i h_{ib} x$ to Alice, who derives the same key from $\sum_{i\in\mathcal{F}} a_i h_{ai} h_{ib} x$ [76].

We assume that no enemy is allowed to be too close to Alice or Bob such that its channel parameters are uncorrelated to theirs. (If there are some correlations, the key encoding scheme discussed in the next chapter can be applied.) Now, we will show that the protocol security depends on the forwarding strategies of the relays. Three strategies are shown in an ascending security order [76].

1. Only one route is taken, i.e., only one relay forwards the packet. This is insecure because an enemy only needs to be close enough to that relay R_i to listen to $a_i h_{ai} x$ sent to Bob and $a_i h_{ib} x$ sent to Alice. If it knows a_i, it can derive $a_i h_{ai} h_{ib} x$ as well as the key with ease.

2. m routes are taken. This is insecure if there are k enemies and $k \geq m$ because each of m out of k enemies can do the same as the single enemy does in 1. After that, they meet and derive $\sum_{i\in\mathcal{F}} a_i h_{ai} h_{ib} x$. The enemy gang has an additional difficulty, however,

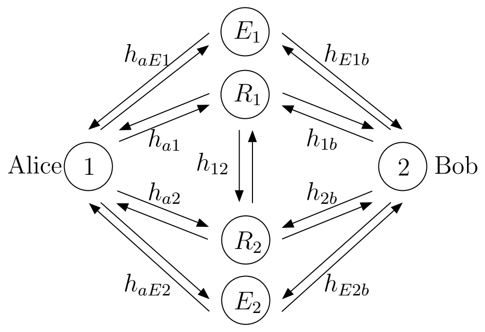

Figure 6.8: A two-relay fork-rake network with two enemies

since interference among relays may make it necessary for the gang to solve a set of linear equations.

3. l routes $0 \leq l \leq n$ are taken at random at each time slot. To do so, each relay chooses to forward the packet with a probability p. This is more secure than 2, especially if the total number of relays $n \gg k$, since those k enemies do not know exactly where to listen, although there can possibly be some flukes, which can be managed by key encoding discussed in the next chapter.

6.5.1 Protocol Evaluation in a Scenario of a Two-Relay Fork-Rake Network with Two Enemies

As a practical example, consider a two-relay fork-rake network shown in Fig. 6.8 with two enemy cryptanalysts and the following assumptions:

1. The two relays are located far enough from each other so that h_{a1} is uncorrelated to h_{a2} and h_{1b} is uncorrelated to h_{2b}.

2. The two relays are close enough to each other so that each of them can hear the signal transmitted from the other.

3. Enemies E_1 and E_2 are located far enough from R_1 and R_2, respectively, such that h_{a1}, h_{a2}, h_{1b}, and h_{2b} are uncorrelated to h_{aE1}, h_{aE2}, h_{E1b}, and h_{E2b}, respectively.

4. Alice transmits x in the first time slot to R_1 and R_2. If R_1 or R_2 will forward the packet further to Bob, depending on its strategy, it will do so in the second time slot. Bob transmits x back in the third time slot. Finally, those relays who forwarded the packet

to Bob in the second time slot will do that to Alice in the fourth one.

5. Enemies E_1 and E_2 are located near enough to R_1 and R_2, respectively, so that, if both relays forward x,

5.1 in the second time slot, E_1 hears $a_1 h_{a1} x$ from R_1 plus the interference $a_2 h_{a2} h_{R2,E1} x$ from R_2, where $h_{R2,E1}$ is the channel gain between R_2 and E_1.

5.2 In the same manner, E_2 hears $a_2 h_{a2} x$ from R_2 plus the interference $a_1 h_{a1} h_{R1,E2} x$ from R_1, where $h_{R1,E2}$ is the channel gain between R_1 and E_2.

5.3 In the fourth time slot, E_1 hears $a_1 h_{1b} x$ from R_1 plus the interference $a_2 h_{2b} h_{R2,E1} x$ from R_2.

5.4 At the same time, E_2 hears $a_2 h_{2b} x$ from R_2 plus the interference $a_1 h_{1b} h_{R1,E2} x$ from R_1.

These assumptions are realistic and will hold except in an unlikely situation where either the relays or the eavesdroppers are only few wavelengths away from the transmitter or the receiver. Now, we would like to evaluate the security of each strategy.

Strategy 1 of Section 6.5 is totally unsafe since, when only one route is taken, there is no interference from another route according to 5.1-5.4, therefore, the enemy E_i closest to the chosen route receives clear signal $a_i h_{ai} x$ and $a_i h_{ib} x$ and can derive $\sum_{i \in \mathcal{F}} a_i h_{ai} h_{ib} x$ as well as the whole key.

Strategy 2 is more secure since there is some interference according to 5.1-5.4. Even when both enemies help each other, it is not easy to derive $\sum_{i \in \mathcal{F}} a_i h_{ai} h_{ib} x$ if the interference is strong and they do not know $a_1 h_{R1,E2}$ and $a_2 h_{R2,E1}$. (Of course, if they do know $a_1 h_{R1,E2}$ and $a_2 h_{R2,E1}$, they simply have to solve a system of two linear equations.) An interference cancellation scheme can be applied, but they still have some disadvantages as compared with Alice and Bob. They may recover some, but not the whole key.

Strategy 3 with the forwarding probability of, let us say 0.5, is even more secure since it is now impossible for each enemy to know whether it receives the desired signal plus interference or merely the interference, especially if the interference is strong and the enemies do not know $a_1 h_{R1,E2}$ and $a_2 h_{R2,E1}$. With some rational guesses, they may recover some parts of the key, though.

In conclusion, Strategies 2 and 3 use interference to make the protocol safer. This is similar to the network coding approach discovered by Shimizu et al. [74]. If the enemies can recover some, but not the whole key, we can apply the key encoding scheme discussed in the Chapter 7 so that the enemies will have no idea at all about the encoded key.

6.5.2 RSF Protocol in Arbitrary Networks

The security protocol for a fork-rake network can be applied sequentially so that it can be used for any multi-path multi-hop network in general. When Strategy 3 is adopted to a more complicated network such that every router forwards the packet with a probability p, it is even more secure. We call this "randomly selective flooding (RSF) protocol." The protocol is similar to that proposed in our Globecom paper [4]. The advantage of this scheme is that it supports datagram networks in which the transmission route might not be known in advance by the transmitter and the receiver. The transmitter and the receiver always generate the secret key from the same channel without knowing from which channel they do. This lack of explicit routing information makes life very hard for the eavesdropper. Now, we will give an example of how the RSF protocol is applied to an arbitrary network.

The scheme proceeds as follows. First, the transmitter broadcasts a pilot packet to every next node on the way to the receiver. Each next node generates a uniformly random real value in the range of zero to one. If the generated value is less than the designated forwarding probability p, the node broadcasts the packet further to its neighbors, who follow the same procedure until the packet reaches the receiver. The receiver uses the packet to estimate channel parameters and derive the secret key. After that, a packet with the same pilot sequence must be sent back along the same path. To do so, the receiver broadcasts it to every neighbor, who, after reading the header, only sends the packet further upstream if it belongs to the previous packet's forward path from the transmitter to the receiver. Again, this repeats until the transmitter gets the packet back

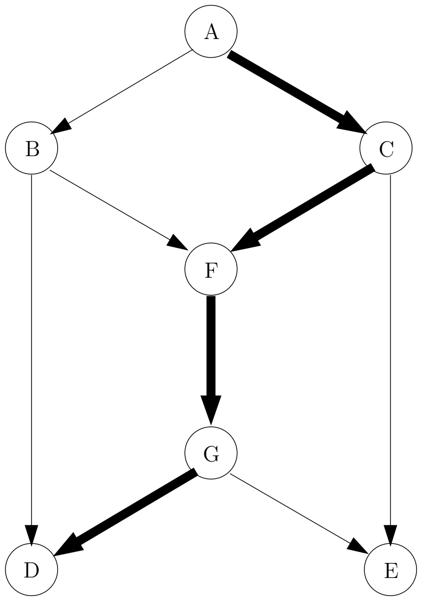

Figure 6.9: A butterfly network

and derives the secret key accordingly.

To illustrate the scheme, let us consider the butterfly network in Fig. 6.9, where A and D are transmitter and receiver, respectively. $ACFGD$ can be randomly chosen by intermediate nodes as a forward path from A to D according to the following scenario.

Forward:

1. A broadcasts a packet to C and B.

2. Only C randomly decides to forward the packet to F and E.

3. Only F randomly decides to forward the packet to G and B.

4. Only G randomly decides to forward the packet to D and E.

Reverse:

1. D broadcasts the packet to B and G.

2. After reading the header, B discards the packet, whereas G sends it to F and E.

3. After reading the header, E discards the packet, whereas F sends it to B and C.

4. After reading the header, B discards the packet, whereas C sends it to A and E.

Source address	Request ID	Destination address	Source sequence no.	Destination sequence no.	Hop count
Pilot sequence					

Figure 6.10: The proposed pilot packet format for the RSF Protocol

5. After reading the header, E discards the packet, whereas A successfully receives it, together with the correct channel information.

Apart from the randomness employed in the protocol, the RSF protocol is similar to the ad hoc on-demand distance vector (AODV) algorithm used for routing in mobile ad hoc networks (MANETs) in many aspects. One aspect is that both algorithms determines a route to a destination only when someone wants to send a packet to that destination [7, 13]. Therefore, we propose a pilot packet format in Fig. 6.10 which is similar to the ROUTE REQUEST packet used for the AODV algorithm. The source address, which is the transmitter's address, together with the request ID, which is a counter incremented whenever a pilot packet is sent, uniquely identify each pilot packet so that intermediate nodes know whether the packet should be discarded. The destination address is the receiver's address. The source sequence number and the destination sequence number are local counters updated by the transmitter and the receiver, respectively, for each pilot packet transmission. The hop-count field is introduced so that flooding is limited to a reasonably narrow region.

Chapter 7

Physical-layer Key Encoding for Wireless Physical-layer Secret-key Generation (WPSG) with Unequal Security Protection (USP)

The previous chapter concerns mainly the process of secret key generation. It does not tell how the encryptor uses the key. In this chapter, we discuss the one-time-pad encryptor as well as a technique called "physical-layer key encoding" used to enhance security when that kind of encryptor is used.

An information-theoretic analysis of physical-layer key generation given in the previous chapter states that there are vulnerable key symbols that might be estimated by eavesdroppers. To protect those key symbols, we introduce physical-layer key encoding in Section 7.1. After that, in Section 7.2, we provide necessary and sufficient conditions on the code in order to achieve perfect secrecy as a function of the number of vulnerable bits and derive the asymptotic code rate accordingly. This perfect secrecy is guaranteed even when the eavesdropper knows the code. When the number of vulnerable symbols is unknown but the ratio between the number of vulnerable key symbols and the total number of generated key symbols is given instead, we suggest an equivalent design parameter used to achieve perfect secrecy in Section 7.3.

However, perfect secrecy for the key encoding scheme may require key input that is too long to be economical, especially if the number of vulnerable key bits is large. In the case of scalable data, such as video data, perfect secrecy is not required for the whole data. Even though low-priority parts in the data do not have perfect secrecy, the enemy cryptanalyst will have no idea at all about the data if high-priority parts are properly protected. From now on, we will call this concept "scalable security" or "unequal security protection (USP)."

Previous research on scalable security is discussed in Section 7.4. After that, we propose a new framework in Section 7.5 to provide scalable security services to the application layer. This framework allows the application layer to specify some design parameters according to its scalable security requirements, which, as shown in Sections 7.6 and 7.7, can be realized in the contexts of secure network coding and physical-layer key encoding, respectively. In the end, Section 7.8 concludes the chapter and suggests some future research.

7.1 Introduction to Physical-layer Key Encoding for a One-Time-Pad Encryptor

As demonstrated in Sections 6.2 and 6.5 of Chapter 6, vulnerable key symbols that can be estimated by an eavesdropper exist in high proportion if there are too few scatterers in the environment, the eavesdropper is located near either the transmitter or the receiver, or, in the case of relay communication, the pilot transmission protocol is insecure. Moreover, although the eavesdropper is nowhere near the transmitter or receiver during transmission, he or she may have been there before and it is possible that channel coefficients (especially the amplitudes) do not change much. Therefore, assuming that he or she can correctly predict some key symbols, we should try to find some countermeasures. To do so, we adopt a similar concept to Shamir's secret sharing [9] and Cai and Yeung's secure network coding [46]. We call our scheme “physical-layer key encoding,” which is designed such that, up to a certain threshold, partial knowledge of key symbols derived from the channel leaves the encoded key completely undetermined. This is performed at the transmitter prior to the one-time pad encryptor block in our wireless physical-layer secret-key cryptosystem shown in Fig. 7.1.

In this section, we shall present physical-layer key encoding which generates “the encoded key” $\mathbf{Z} = [Z_1, Z_2, ..., Z_J]$ as a codeword from “an original quantized key” $\mathbf{K} = [K_1, K_2, ..., K_{I_K}]$ and feeds it into the one-time pad encryptor. The encoder aims at protecting the secret key in case the enemy can correctly estimate some channel coefficients, and hence some original quantized key symbols. Three parameters, I_K, I_{SK}, and I_{VK} are of interest to the encoder. The first one, I_K, is the number of symbols that can be generated by the quantizer in Fig. 7.1. The second one, I_{SK}, is the number of secure key symbols that the enemy cannot correctly estimate, whereas the third, I_{VK}, is the number of vulnerable ones that he or she can correctly estimate, such that $I_K = I_{SK} + I_{VK}$.

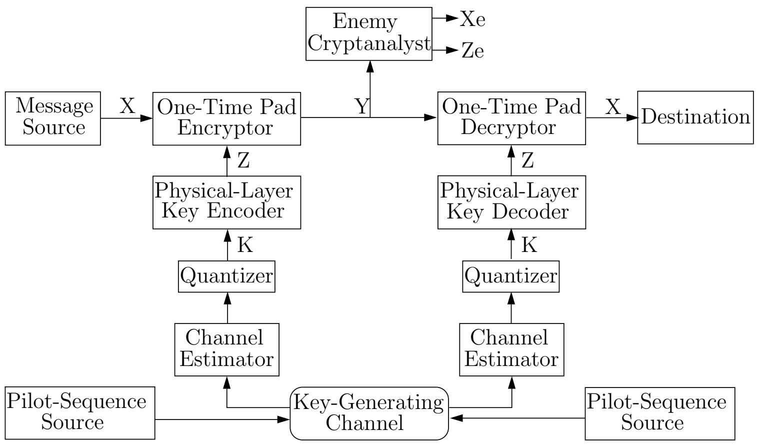

Figure 7.1: A modified secret-key cryptosystem based on mutual CSI with physical-key encoding and decoding

The information-theoretic derivation of I_K, I_{SK}, and I_{VK} is discussed in [34] but not in this chapter, where we are more interested in the following questions: Given I_K, I_{SK}, and I_{VK}, what is the important property of the physical-layer key encoding that ensures perfect secrecy as well as efficiency? How can we choose the code rate? And how can we derive the optimal code? Some questions will be answered by Theorems 7.1-7.3 proposed in the next section.

7.2 Perfect Secrecy in Physical-Layer Key Encoding for a One-Time-Pad Encryptor

Consider a non-randomized cipher in which elements in the plaintext $\mathbf{X} = [X_1, X_2, ..., X_M]$, ciphertext $\mathbf{Y} = [Y_1, Y_2, ..., Y_N]$, and secret key $\mathbf{Z} = [Z_1, Z_2, ..., Z_J]$ all take values in an L-ary alphabet and $J = N = M$. Suppose that the key is chosen to be completely random, i.e., $P(Z_i = z) = L^{-M}$, $i = 1, 2, ..., M$, for all possible values z of the secret key, and that the enciphering transformation is

$$Y_i = (X_i + Z_i) \bmod L, \; i = 1, 2, ..., M. \tag{7.1}$$

Since, for each possible choice x_i and y_i of X_i and Y_i, respectively, there is a unique z_i such that $Z_i = z_i$ satisfies (7.1), it follows that $P(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{x}) = L^{-M}$ for every particular $\mathbf{y} = [y_1, y_1, ..., y_M]$ and $\mathbf{x} = [x_1, x_2, ..., x_M]$, no matter what the statistics of $\mathbf{X}$ may be. Thus, $\mathbf{X}$ and $\mathbf{Y}$ are statistically independent, and hence this system provides perfect secrecy [31]. The system is called a modulo-L Vernam system or one-time pad and is used in our model in Fig. 7.1 to combine the message and the encoded key together.

Since, out of the total I_K quantized key symbols, I_{VK} symbols are vulnerable symbols, we can see that, had a one-time pad encryptor been used without physical-key encoding, I_{VK} ciphertext symbols would have been decrypted by the eavesdropper. Thus, in order to construct a secure Y_1, we use the following linear combination for Z_1.

$$Z_1 = (K_1 + K_2 + ... + K_{I_{VK}+1}) \bmod L \tag{7.2}$$

After substituting (7.2) into (7.1) using $i = 1$, we can see that even if the set of all vulnerable key symbols is a subset of $\{K_1, K_2, ..., K_{I_{VK}+1}\}$, there is still one symbol that is unknown to the eavesdropper. If every key symbol is statistically independent of others, perfect secrecy of Y_1 is achieved.

Now, in order to construct Z_2, Z_3, and so on, we perform a linear combination of I_{VK} key symbols similar to (7.2). Therefore, our physical-layer key encoding can be defined by a linear block code C_p transforming an I_K-tuple $\mathbf{K}$ over $GF(q^{I_K})$ of key symbols obtained from the quantizer into an M-tuple codeword $\mathbf{Z}$ over $GF(q^M)$. We propose the following theorem providing two necessary and sufficient conditions for perfect secrecy.

Theorem 7.1 *If I_{VK} out of I_K physical-layer key symbols generated by the quantizer can be correctly estimated by the eavesdropper, the cryptosystem in Fig. 7.1 still maintains perfect secrecy if and only if each member in the I_K-dimensional vector $\mathbf{K}$ is statistically independent of one another and the physical-layer code C_p has the following properties. [4]*
7.1.1. Every code symbol is a linear combination of at least $I_{VK} + 1$ input symbols.
7.1.2. Every linear combination of any subset of code symbols results in a linear combination of at least $I_{VK} + 1$ input symbols.

Proof From earlier discussion it should be clear that the first condition is necessary. The necessity of the second condition can be proved by contradiction as follows. If the combination of ν code symbols $\sum_{i=1}^{\nu} Z_i$ yields a combination of $I_{VK}+1-\delta$ input symbols, where δ is a positive integer, the eavesdropper who calculates $\sum_{i=1}^{\nu} Y_i$ may know the exact value of $\sum_{i=1}^{\nu} Z_i$, since the number of input symbols in the combination does not exceed the number of vulnerable symbols. If $\sum_{i=1}^{\nu} Z_i$ is known, perfect secrecy is not achieved because these ν encoded key symbols are not independent.

As for the sufficiency, we can also prove it by contradiction. Now, we have to prove that if perfect secrecy is not achieved, either the condition 7.1.1 or 7.1.2 is not satisfied. By definition, perfect secrecy in a one-time-pad system is not achieved only if either 1) at least one Z_i, $i = 1, 2, ..., M$ is known or 2) any linear combination of some Z_i is known implying linear dependence among key symbols. Since 1) and 2) will not happen if the condition 7.1.1 and 7.1.2 are, respectively, satisfied, the two conditions are sufficient. $\square$

Now, we focus our attention on the special case when $\mathbf{K}$ and $\mathbf{Z}$ are vectors of binary data and present the second theorem.

Theorem 7.2 *If $\mathbf{K} \in GF(2^{I_K})$, in order to generate $\mathbf{Z} \in GF(2^n)$, which is a codeword of n encoded key bits providing perfect secrecy in our system, the following conditions on I_K are necessary and sufficient. [4]*

7.2.1. If $I_{VK}+1$ is even,

$$I_K \geq \frac{(n+1)}{2}(I_{VK}+1) \tag{7.3}$$

7.2.2 If $I_{VK}+1$ is odd,

$$I_K \geq \frac{(n+1)}{2}(I_{VK}+1) + \frac{(n-1)}{2} \tag{7.4}$$

Proof We prove this theorem by mathematical induction. We first demonstrate that the theorem is valid for $n = 1$ and $n = 2$ before showing that if the theorem is valid for any n, it will hold true for $n + 1$. The case of $n = 1$ can be easily validated by substituting it into (7.3) and (7.4) and observing that the resulting I_K corresponds to that in Theorem 7.1.

Given Z_1 in Eq. (7.2), when $L = 2$ and $I_{VK} + 1$ is an even number. Let

$$Z_2 = K_{\frac{1}{2}(I_{VK}+3)} \oplus K_{\frac{1}{2}(I_{VK}+5)} \oplus ... \oplus K_{\frac{3}{2}(I_{VK}+1)}. \tag{7.5}$$

Now, the generator matrix generating Z_1 and Z_2 can be written as follows.

$$\mathbf{G_p} = \begin{bmatrix} \overbrace{1\ 1\ ...\ 1\ 1\ 1\ ...\ 1}^{I_{VK}+1}\ \overbrace{0\ 0\ ...\ 0\ 0\ 0\ ...\ 0}^{I_K-I_{VK}-1} \\ \underbrace{0\ 0\ ...\ 0}_{\frac{1}{2}(I_{VK}+1)}\underbrace{1\ 1\ ...\ 1\ 1\ 1\ ...\ 1}_{I_{VK}+1}\ 0\ 0\ ...\ 0 \end{bmatrix}^T = [\mathbf{g}_1\ \mathbf{g}_2] \tag{7.6}$$

such that

$$\mathbf{Z} = [Z_1, Z_2] = \mathbf{K} \cdot \mathbf{G_p}, \tag{7.7}$$

where

$$\mathbf{K} = [K_1, K_2, ..., K_{I_K}]. \tag{7.8}$$

When $n = 2$, i.e., only two encoded key bits are to be constructed, we can see from (7.6) that $I_K = \frac{3}{2}(I_{VK} + 1)$ original key bits from the quantizer are sufficient, i.e., the zero padding after the $\frac{3}{2}(I_{VK} + 1)^{th}$ is trivial. This sufficient value of I_K holds for any given $\mathbf{g}_1$ with Hamming weight of $I_{VK} + 1$ because, in order to choose $I_{VK} + 1$ rows of $\mathbf{g}_2$ with 1-valued elements, we should have $\frac{I_{VK}+1}{2}$ rows where those elements of $\mathbf{g}_1$ in the same positions have the value 1. In other words, if we pile $\mathbf{g}_2$ up onto $\mathbf{g}_1$, at most $\frac{I_{VK}+1}{2}$ positions of 1-valued elements may overlap. If there are more overlapping positions, the second condition in Theorem 1 will be violated. If there are less overlapping positions, no condition is violated, but it is not an economical use of original quantized key bits because I_K must now be greater than $\frac{3}{2}(I_{VK} + 1)$.

We can therefore conclude that with $n = 2$ and an even $I_{VK} + 1$, $I_K \geq \frac{(n+1)}{2}(I_{VK} + 1)$ is a necessary and sufficient condition on I_K. If we follow the same line of reasoning with $n > 2$, we see that when each $\mathbf{g}_{i+1}$ is piled up onto the heap of $\mathbf{g}_j$, $j = 1, 2, ..., i$, the most economical way in terms of the number of original quantized key bits used is still to have $\frac{I_{VK}+1}{2}$ 1-valued elements in overlapped positions with any 1-valued elements in any of those $\mathbf{g}_j$, $j = 1, 2, ..., i$. This means at least $\frac{I_{VK}+1}{2}$ more original quantized key bits are needed for each increment of n, thus completing the proof of 7.2.1.

A similar line of reasoning can be used to prove 7.2.2. With odd $I_{VK} + 1$, $\mathbf{G_p}$ in (7.6) becomes

$$\mathbf{G_p} = \begin{bmatrix} \overbrace{1\,1\,...\,1\,1\,1\,...\,1}^{I_{VK}+1}\ \overbrace{0\,0\,...\,0\,0\,0\,...\,0}^{I_K-I_{VK}-1} \\ \\ \underbrace{0\,0\,...\,0}_{\frac{1}{2}(I_{VK}+2)}\underbrace{1\,1\,...\,1\,1\,1\,...\,1}_{I_{VK}+1}\ 0\,0\,...\,0 \end{bmatrix}^T = [\mathbf{g}_1\ \mathbf{g}_2] \qquad (7.9)$$

With the overlapped positions reduced from $\frac{I_{VK}+1}{2}$ in the even case to $\frac{I_{VK}}{2}$ in the odd case, the value of necessary I_K for each n increases. One can verify Eq. (7.4) when $n = 2$ by looking at the $\mathbf{G_p}$ in (7.9). With $n \geq 2$, at least $\frac{I_{VK}+2}{2}$ more original quantized key bits are needed for each increment of n, thus completing the proof of 2.2. $\square$

The next theorem concerns the asymptotic rate of the code.

Theorem 7.3 *The minimum asymptotic code rate of C_p as $n \to \infty$ is $\frac{I_{VK}+1}{2}$ if $I_{VK}+1$ is even. Otherwise, it is $\frac{I_{VK}+2}{2}$. [4]*

Proof The code rate is the ratio between the number of originial bits to that of encoded ones. Therefore, the asymptotic code rate as $n \to \infty$ is derived by dividing (7.3) and (7.4) by n and finding the limit of the results as as $n \to \infty$. □

From the proof of Theorem 7.2, we have derived our generator matrix prototype of a physical-layer key encoder for any value of I_{VK}, when $I_{VK}+1$ is even or odd in equations (7.10) or (7.11), respectively.

$$\mathbf{G_p} = \begin{bmatrix} \overbrace{1\ 1\ \ldots\ 1\ 1\ \ldots\ 1}^{I_{VK}+1}\ \overbrace{0\ \ldots\ 0\ 0\ \ldots\ 0\ 0\ 0\ \ldots\ 0}^{I_K-I_{VK}-1} \\ \underbrace{0\ 0\ \ldots\ 0}_{\frac{1}{2}(I_{VK}+1)}\underbrace{1\ \ldots\ 1\ 1\ \ldots\ 1}_{I_{VK}+1}\ 0\ \ldots\ 0\ 0\ 0\ \ldots\ 0 \\ \vdots \\ \underbrace{0\ 0\ \ldots\ 0\ 0\ \ldots\ 0\ 0\ \ldots\ 0\ 0\ \ldots\ 0}_{I_K-I_{VK}-1}\underbrace{1\ 1\ \ldots\ 1}_{I_{VK}+1} \end{bmatrix}^T \tag{7.10}$$

$$\mathbf{G_p} = \begin{bmatrix} \overbrace{1\ 1\ \ldots\ 1\ 1\ \ldots\ 1}^{I_{VK}+1}\ \overbrace{0\ \ldots\ 0\ 0\ \ldots\ 0\ 0\ 0\ \ldots\ 0}^{I_K-I_{VK}-1} \\ \underbrace{0\ 0\ \ldots\ 0}_{\frac{1}{2}(I_{VK}+2)}\underbrace{1\ \ldots\ 1\ 1\ \ldots\ 1}_{I_{VK}+1}\ 0\ \ldots\ 0\ 0\ 0\ \ldots\ 0 \\ \vdots \\ \underbrace{0\ 0\ \ldots\ 0\ 0\ \ldots\ 0\ 0\ \ldots\ 0\ 0\ \ldots\ 0}_{I_K-I_{VK}-1}\underbrace{1\ 1\ \ldots\ 1}_{I_{VK}+1} \end{bmatrix}^T \tag{7.11}$$

7.3 The Equivalent Number of Vulnerable Bits for the Design of Physical-Layer Key Encoding

The analysis given in the last section is based on the assumption that the encoder knows I_{VK}. However, empirical results in [33, 34] show that the encoder normally can only estimate the ratio I_{VK}/I_K instead of I_{VK} alone. In this section, we derive an equivalent number of vulnerable bits, denoted by I'_{VK}, as a design parameter that can substitute I_{VK} in the theorems discussed in the last section. To do so, we first propose a model of an enemy cryptanalyst based on the ratio I_{VK}/I_K estimated by the legitimate transmitter.

According to Fig. 7.2 (a), the enemy is modeled to have almost the same structure as the legitimate transmitter and receiver. However, since it can only estimate channel coefficients from an imaginary channel which differs from the key-generating channel used by the legitimate terminals, we propose an equivalent model in Fig. 7.2 (b). In the equivalent model, the enemy does estimate the key-generating channel, but the quantized key $\mathbf{K}$ is distorted by a binary symmetric channel (BSC), erring each estimated key bit with a probability p. The legitimate transmitter has to guess this error probability p based on its estimate of $1-(I_{VK}/I_K)$. For security, the estimated value p should not be more than the estimated $1-(I_{VK}/I_K)$.

Based on the equivalent model, we can specify the relationship between the estimate of p and the design parameter I'_{VK} needed for perfect secrecy in Theorem 7.4.

Theorem 7.4 *If the enemy cryptanalyst behaves according to the model in Fig. 7.2 (b) having p as the error probability of the binary symmetric channel, and the generator matrix prototype in the form of equations (7.10) or (7.11) is used, when $I'_{VK}+1$, being even or odd, respectively, substitutes $I_{VK}+1$, the following conditions on I'_{VK} are sufficient for perfect secrecy. [4]*

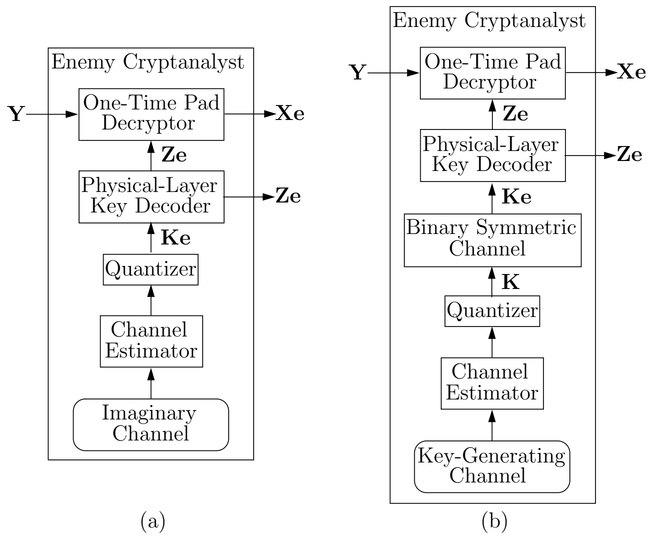

Figure 7.2: The model of an enemy cryptanalyst (a), and its equivalent (b)

7.4.1. If $I'_{VK}+1$ is even,

$$I'_{VK}+1 \geq \begin{cases} \lceil -\frac{2}{\log_2(1-p)} \rceil \ , & \textit{if } \lceil -\frac{2}{\log_2(1-p)} \rceil \textit{ is even} \\ \lceil -\frac{2}{\log_2(1-p)} \rceil + 1 \ , & \textit{if } \lceil -\frac{2}{\log_2(1-p)} \rceil \textit{ is odd.} \end{cases} \tag{7.12}$$

7.4.2. If $I'_{VK}+1$ is odd,

$$I'_{VK}+1 \geq \begin{cases} \lceil -\frac{2}{\log_2(1-p)} \rceil - 1 \ , & \textit{if } \lceil -\frac{2}{\log_2(1-p)} \rceil \textit{ is even} \\ \lceil -\frac{2}{\log_2(1-p)} \rceil \ , & \textit{if } \lceil -\frac{2}{\log_2(1-p)} \rceil \textit{ is odd,} \end{cases} \tag{7.13}$$

where $\lceil -\frac{2}{\log_2(1-p)} \rceil$ is the smallest integer that is larger than $-\frac{2}{\log_2(1-p)}$.

Proof For perfect secrecy, the probability that the enemy can successfully decrypt x bits in the plaintext is at most $(\frac{1}{2})^x$, which equals the success probability of a clueless guess. Therefore, according to equations (7.10) or (7.11), if $x = 1$, the probability that the enemy correctly predicts $I'_{VK}+1$ original key bits, which is $(1-p)^{I'_{VK}+1}$, must not exceed $\frac{1}{2}$.

$$(1-p)^{I'_{VK}+1} \leq \frac{1}{2} \tag{7.14}$$

$$I'_{VK}+1 \geq -\frac{1}{\log_2(1-p)} \tag{7.15}$$

Now, if $x = n$, which is the number of bits in the generated codeword as well as the number of bits in the plaintext, and $I'_{VK} + 1$ is even,

$$(1-p)^{\frac{n+1}{2}(I'_{VK}+1)} \leq (\frac{1}{2})^n \tag{7.16}$$

For very large n, (7.16) becomes

$$\begin{aligned}
\lim_{n\to\infty}(1-p)^{\frac{n+1}{2}(I'_{VK}+1)} &\leq \lim_{n\to\infty}(\frac{1}{2})^n && (7.17)\\
(1-p)^{\frac{n}{2}(I'_{VK}+1)} &\leq (\frac{1}{2})^n && (7.18)\\
I'_{VK}+1 &\geq -\frac{2}{\log_2(1-p)} && (7.19)
\end{aligned}$$

We can see that, as n increases, the minimal value of $I'_{VK} + 1$ that should be set also increases. If we denote by $\lceil -\frac{2}{\log_2(1-p)} \rceil$ the smallest integer that is larger than $-\frac{2}{\log_2(1-p)}$, the direct consequence of (7.19), when $I'_{VK} + 1$ is constrained to be an even number, will be the condition 7.4.1.

In case $I'_{VK}+1$ is odd, we use the prototype in Eq. (7.11) and follow the same reasoning.

$$\begin{aligned}
(1-p)^{\frac{n+1}{2}(I'_{VK}+1)+\frac{n-1}{2}} &\leq (\frac{1}{2})^n && (7.20)\\
\lim_{n\to\infty}(1-p)^{\frac{n+1}{2}(I'_{VK}+1)+\frac{n-1}{2}} &\leq \lim_{n\to\infty}(\frac{1}{2})^n && (7.21)\\
(1-p)^{\frac{n}{2}(I'_{VK}+2)} &\leq (\frac{1}{2})^n && (7.22)\\
I'_{VK}+1 &\geq -\frac{2}{\log_2(1-p)} - 1 && (7.23)
\end{aligned}$$

This results in the condition 7.4.2. Therefore, with the same p, we can set $I'_{VK} + 1$ to be smaller by 1 bit if it is odd, as compared with the even case. □

In the last section, we have suggested two simple generating matrix prototypes for our physical-layer key encoding for two specific cases, when $I_{VK}+1$ is even and when it is odd, where I_{VK} is the number of vulnerable key bits. $I_{VK} + 1$ is related to I_K, the number of original key bits needed from the quantizer, by Theorem 7.2. In case $I_{VK}+1$ is unknown, we use Theorem 7.4 to derive $I'_{VK}+1$ as an equivalent of $I_{VK}+1$ from the probability p that the eavesdropper incorrectly estimates a key bit. For example, $I'_{VK}+1$ is at least 5 when p is 0.25, yielding an asymptotic code rate of 3, as predicted by Theorem 7.3.

The next section discusses scalable security, which will be related to the physical-layer key encoding in the end.

7.4 Literature Review of Scalability in Practical Security Domain

Scalable security has been previously investigated in [22,27,32,72] using selective encryption schemes. The basic idea of the scheme is to encrypt only some important parts of the scalable data. These works consider several video coding standards, especially MPEG-I, from which scalable video data is used to evaluate their encryption scheme.

MPEG encoding of a video sequence requires compression in two dimensions. In the time dimension, a combination of blocked-based motion compensation is applied to remove inter-frame temporal redundancy. In the space dimension, discrete cosine transform (DCT)-based compression is used to remove intra-frame spatial redundancy. After compression, frames are formed by several compressed blocks. Then, a number of frames are grouped together to form a random access unit, called a group of pictures (GOP), so that the video can be viewed either forward or backward. Each GOP has little or no dependence on other GOPs [27].

A GOP consists of three types of frames, which are intracoded frames (I-frames), motion-estimated forward predicted frames (P-frames), and motion-estimated bidirectional predicted frames (B-frames). The encoder of the I-frame uses the same scheme as JPEG encoding of still frames. The I-frame is used as the motion-estimated reference for the P- and B- frames. The P-frame is encoded with reference to the most recently previous I- or P- frame, whereas the B-frame is encoded with reference to both the most recently previous as well as the most immediately succeeding I- or P- frame [27].

In [22, 32], a selective encryption scheme is used such that only the I-frames are encrypted. However, empirical evidence in [27, 72] shows that the scheme is insecure. The authors of [27] suggest that improvements can be made if some more parts in the data are encrypted. Although this is a good compromise, the basic idea is the same.

It is important to note that the encryption scheme used in these works, such as Data Encryption Standard (DES) and Rivest, Shamir, and Adleman (RSA) algorithm, only provides practical, but not theoretical security. Any cryptanalyst who posseses an incredibly large computing power can decrypt the secret data. Since we are interested in theoretical security, our method used to achieve scalable security is different. The next section provides the framework to our approach.

7.5 Scalability Framework in Theoretical Security

One normally thinks of a security concept as a dichotomy of being secure or insecure, with nothing in between. In order to understand our framework, it is most important to understand that absolute security and absolute insecurity are the two extremes of a continuum of the guessing success probability.

Consider a piece of two-bit data, when we say that the data is absolutely secure, we mean that the probability that the enemy can correctly guess the encrypted data is 0.25. In contrast, it is absolutely insecure if such a probability, which will be called the guessing success probability from now on, is 1.

Earlier frameworks based on selective encryption have only concepts of (practically) absolute security and absolute insecurity. According to our example, any encryption scheme providing the guessing success probability p_g such that $0.25 < p_g < 1$ cannot be incorporated into those frameworks.

When the concept of guessing is included in the scalable security framework, the guessing success probability p_g becomes one security benchmark so that the security levels of encryption schemes can be measured and compared. n-symbol data is absolutely secure if $p_g = (\frac{1}{|\mathbb{F}|})^n$, where $|\mathbb{F}|$ is the field size of each symbol. It is absolutely insecure if $p_g = 1$.

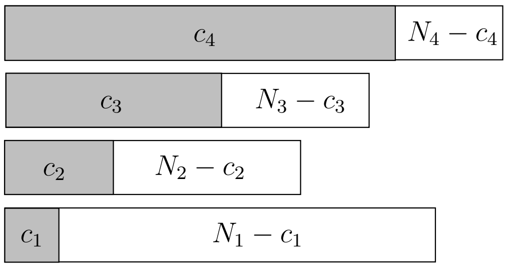

Figure 7.3: An illustration of combinatorial weak security limit

It is weakly secure if $\frac{1}{|\mathbb{F}|^n} < p_g < 1$. This p_g should be determined by the application layer who has a better idea about how weak the data security in each priority class should be allowed to be.

In our scalable security framework, the guessing success probability is not the only benchmark, since, for a given data priority i having N_i data symbols, we are interested in making sure that only up to a limited number c_i of symbols are weakly secure with a guessing success probability p_g not exceeding a threshold p_{ti}, whereas the other $N_i - c_i$ symbols are perfectly secure. Therefore, we propose in Definition 7.4 another benchmark called the combinatorial weak security limit (CWSL) which provides the parameter c_i for each priority class. Prior to that, the concepts of a priority class, a priority classification function, and an ordered scalable message are defined in Definitions 7.1-7.3, respectively.

A combinatorial weak security limit c_i for each priority class i is illustrated in Fig. 7.3. The smaller the i, the higher the priority and therefore the lower the proportion c_i/N_i of weakly secure symbols. Note that c_i symbols do not need to stick together at the front of the data, but can be arbitrarily distributed.

Definition 7.1 *Given a scalable message* $\mathbf{M}= [m_1, m_2, ..., m_\omega]$, *a set of symbols with priority class i,* $i > 0$, *is given by* $\mathcal{Q}_i= \{q_{i1}, q_{i2}, ..., q_{i\upsilon_i}\}$, *where each* q_{ik}, $1 \leq k \leq \upsilon_i$, *is a distinct element in* $\mathbf{M}$. *Each member in* $\mathcal{Q}_i$ *is less important than any member in* $\mathcal{Q}_{i-1}$ *and may be useless without the recovery of some members in* $\mathcal{Q}_{i-1}$.

Definition 7.2 $\pi(m_j)$ *is a priority classification function of a scalable symbol* m_j *belonging to the scalable message* $\mathbf{M}= [m_1, m_2, ..., m_\omega]$ *if and only if* m_j *is of the priority class* $\pi(m_j) \in \mathbb{Z}^+$, $1 \leq j \leq \omega$.

Definition 7.3 *A scalable message* $\mathbf{M}= [m_1, m_2, ..., m_\omega]$ *classified by the vector* $\Pi(\mathbf{M}) = [\pi(m_1), \pi(m_2), ..., \pi(m_\omega)]$ *is said to be ordered if and only if* $\pi(m_v) \leq \pi(m_u), 1 \leq v < u \leq \omega$.

Definition 7.4 *A combinatorial weak security limit (CWSL) vector* $\mathbf{C}= [c_1, c_2, ..., c_\psi]$ *associated with a guessing success probability threshold (GSPT) vector* $\mathbf{P_t}= [p_{t1}, p_{t2}, ..., p_{t\psi}]$ *of an ordered scalable message* $\mathbf{M}= [m_1, m_2, ..., m_\omega]$ *is a vector whose each element* c_i *represent the maximum number of weakly secure symbols permitted by the application layer for the priority class* i*, such that the ordered scalable message* $\mathbf{M}$ *is considered insecure if, for some* i*, at least one of the following conditions holds.*
7.4.1. The eavesdropper can decrypt at least one set of symbols $\mathbf{N_{c_i}}= \{n_1, n_2, ..., n_{c_i}\}$ *belonging to the* i^{th} *priority with the guessing success probability* p_g*, where* $p_g > p_{ti} \geq (\frac{1}{|\mathbb{F}|})^{c_i}$.
7.4.2. The eavesdropper can decrypt at least one set of symbols $\mathbf{N_{\chi_i}}= \{n_1, n_2, ..., n_{\chi_i}\}$, $\chi_i > c_i$*, belonging to the* i^{th} *priority with the guessing success probability* $p_g > p_{ti} \cdot (\frac{1}{|\mathbb{F}|})^{\chi_i - c_i}$.

The first condition in Definition 7.4 states that, with a CWSL vector and a GSPT vector given by the application layer, it is required that the encryption must not allow the eavesdropper to decrypt c_i symbols of priority i with the guessing success probability exceeding p_{ti}. Therefore, the GSPT vector specifies the limit of weakness in our security scheme.

The second condition in Definition 7.4 states that, beyond the limit of c_i symbols of priority i, it is required that the guessing success probability is reduced by the factor $\frac{1}{|\mathbb{F}|}$ for each symbol beyond the limit. In other words, we only allow weak security for any arbitrary set of up to c_i symbols. Any more symbols added to the set must be regarded as perfectly secure.

This scalable security framework is mainly designed for cryptosystems using Shamir's concept of secret sharing. We will discuss two such systems, which are weakly secure network coding and our physical-layer key encoding in a one-time pad cryptosystem in Sections 7.6 and 7.7, respectively.

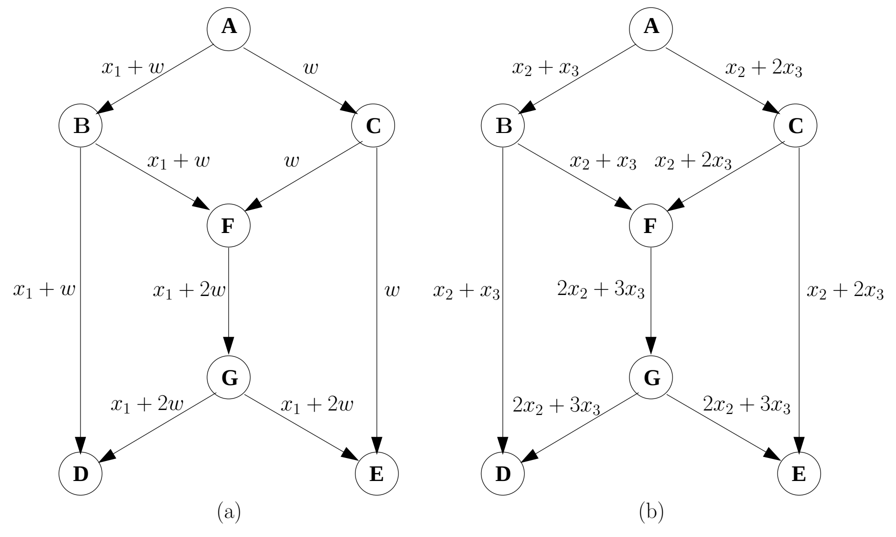

Figure 7.4: Secure network coding in a butterfly network with (a) Shannon security and (b) weak security

7.6 Weakly Secure Network Coding in the Scalable Security Framework

Bhattad and Narayanan explain the difference between Shannon security and weak security in their work on weakly secure network coding [35]. They state that the source information is Shannon secure if $I(\mathbf{X};\mathbf{Y}) = 0$, where $\mathbf{X} = [x_1, x_2, ..., x_r]$ is the source information and $\mathbf{Y}$ is the set of messages to which an eavesdropper can listen. In the case of Shannon security, the eavesdropper has no information about the source at all. On the other hand, the source information is weakly secure if $I(x_i;\mathbf{Y}) = 0$, $1 \leq i \leq r$.

The difference between two security concepts can be illustrated by secure network coding in Fig. 7.4. The goal of secure network coding is to make sure that an eavesdropper who has access to a limited number k of edges cannot decrypt the source information. Figures 7.4 (a) and (b) are an example of secure network coding with $k = 1$. According to the figures, the node F performs network coding by adding the symbol from the edge BF to that from CF, yielding a symbol in FG. In Fig. 7.4 (a), the source symbol x_1 is mixed with a random symbol w, whereas in Fig. 7.4 (b), two source symbols x_2 and x_3 are mixed together. Observe that, for both figures 7.4 (a) and (b), the eavesdropper who has access only to a single edge cannot decrypt x_1, x_2, or x_3.

Although the source symbols in both Fig. 7.4 (a) and (b) cannot be decrypted by wiretapping a single edge, the degrees of security in the two figures differ. If we check the definition of Shannon security and weak security, we find that secure network coding in Fig. 7.4 (a) is Shannon secure since $I(\mathbf{X} = [x_1]; \mathbf{Y}) = 0$, where $\mathbf{Y}$ is the information symbol at any single edge. However, that in Fig. 7.4 (b) is only weakly secure since, although $I(x_i; \mathbf{Y}) = 0$, $i = 2, 3$, $I(\mathbf{X} = [x_2, x_3]; \mathbf{Y}) = \log_2 |\mathbb{F}|$, where $|\mathbb{F}|$ is the field size of the symbols x_2 and x_3. (In this case, x2 and x3 can take their values from the Galois field with the size of 5 or greater.)

Secure network coding in Fig. 7.4 can be seen as a special case of the scalable security framework proposed in the previous section. We can see that, when the field size of x_1, x_2, x_3, and w is $|\mathbb{F}|$, the guessing success probability p_g of x_1 is $\frac{1}{|\mathbb{F}|}$, and that of $[x_2, x_3]$ is $\frac{1}{|\mathbb{F}|}$ as well. The degree of security in Fig. 7.4 (b) is weaker since, with the same p_g, two symbols are decrypted as compared with one symbol in Fig. 7.4 (a).

The practical implication is that the high-priority symbols should be encrypted according to Fig. 7.4 (a), whereas the low-priority ones may consider the encryption in Fig. 7.4 (b). For example, if we have an ordered scalable message $\mathbf{M}= [m_1, m_2, m_3]$ such that $\Pi(\mathbf{M}) = [1, 2, 2]$, the symbol m_1 should be encrypted according to Fig. 7.4 (a), whereas m_2 and m_3 may follow Fig. 7.4 (b), if the CWSL vector $\mathbf{C}= [1, 2]$ and the GSPT vector $\mathbf{P_t}= [\frac{1}{|\mathbb{F}|}, \frac{1}{|\mathbb{F}|}]$ are specified by the application layer. (See Definition 7.4.)

7.7 Physical-layer Key Encoding in the Scalable Security Framework

Scalable security can be realized in physical-layer key encoding by reducing the number of 1-elements in (7.10) or (7.11) and shifting the groups of 1-elements to the left. This will save the number of original key symbols needed. We will give an example problem as follows.

Problem 7.1. The application layer requires that at most three data bits in the second priority class can be weakly secure with the eavesdropper's guessing success probability of 0.25. If the number of vulnerable key bits is 3, specify the generator matrix of the physical-layer key encoding generating four encoded key bits.

Solution. In this problem, we start from the generator matrix in (7.10) for perfect secrecy with parameters $I_{VK} = 3$, and $n = 4$. The overlapping part of the groups of 1-elements in two adjacent columns consists of two elements such that

$$\mathbf{G_p} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}^T . \tag{7.24}$$

For scalable security satisfying the conditions $c_2 = 3$ and $p_{t2} = 0.25$, we reduce the number of 1-elements in each column from 4 to 3. We also shift each row of the transpose of the generator matrix to the left such that it becomes

$$\mathbf{G_p} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}^T . \tag{7.25}$$

Now, we will verify that this generator matrix results in weak security corresponding to Definition 7.4. According to eqs. (7.1), (7.2), (7.7), (7.8), and (7.24), the ciphertext Y_i can be written as

$$Y_i = K_i \oplus K_{i+1} \oplus K_{i+2} \oplus X_i, \;\; i = 1, 2, 3, 4. \tag{7.26}$$

Next, we will check the first condition in Definition 7.4 where $c_2 = 3$. The best way for the eavesdropper to guess three symbols is to add together three adjacent ciphertext symbols, for example,

$$\sum_{i=1}^{3} Y_i = K_1 \oplus K_3 \oplus K_5 \oplus \sum_{i=1}^{3} X_i. \tag{7.27}$$

With the summation $\sum_{i=1}^{3} Y_i$ known by wiretapping, and $K_1 \oplus K_3 \oplus K_5$ known if K_1, K_3, and K_5 are vulnerable symbols, the enemy knows the summation $\sum_{i=1}^{3} X_i$. This means, by guessing two bits out of three bits X_i, $i = 1, 2, 3$, correctly, the enemy knows all the three bits. The probability of guessing two bits correctly is 0.25, corresponding to the requirement $p_{t2} = 0.25$. We can see that there is no way to derive any three bits with more guessing success probability than 0.25. Therefore, the first condition is not met, meaning that our encryption will be considered scalably secure if the second condition is not met either.

By checking every possible way for the enemy to derive all four plaintext bits X_i, $i = 1, 2, 3, 4$, we see that the maximum guessing success probability is 0.125. Since $0.125 = p_{ti} \cdot (\frac{1}{|\mathbb{F}|})^{\chi_i - c_i} = 0.25 \cdot (\frac{1}{2})^{4-3}$, the second condition is not met and our encryption is therefore scalably secure according to the requirement.

One may observe that, by using the given scalable security requirement instead of the perfect secrecy requirement, we reduce the number of original symbols I_K needed by 40%.

7.8 Conclusion and Future Research

We have proposed physical-layer key encoding for the WPSG cryptosystem with one-time pad encryptor. Four theorems indicate the required properties of the codes in order to achieve perfect secrecy of the secret data. After that, we discuss a scalable security framework specifying the degrees of security weakness that can be allowed in lower-priority data. Our framework applies to weakly secure network coding as well as our physical-layer key encoding in a WPSG cryptosystem and, indeed, any key encoding schemes for one-time pad cryptosystems having vulnerable key bits. We show that the number of original key bits needed can be significantly reduced if weak security is allowed.

In future, we hope that some algorithms are developed such that, given any scalable security requirement, the code can be systematically designed.

Chapter 8

Summary, Conclusion, and Future Works

8.1 Summary and Conclusion

Physical key encoding has been presented in Chapter 7 as a means to protect security in the presence of vulnerable key symbols. The encoded output key is shorter than the input, thus sacrificing some key length for the sake of security. The counterpart of physical key encoding is Slepian-Wolf coding, which adds some redundancy to the quantized key such that legitimate receivers are protected against key mismatch due to channel estimation error.

The relationship between physical key encoding and Slepian-Wolf coding in WPSG is similar to source coding and channel coding in a communication system in such a way that the latter expands what the former contracts. It is proved that optimality can be achieved when the source coding and the channel coding in a communication perform their tasks separately. It is yet to be proved whether such optimality holds with the separation of physical key encoding and Slepian-Wolf coding in WPSG.

Just as channel coding has an unequal-error-protection (UEP) capability, physical key encoding has an unequal-security-protection (USP) capability. The concept of USP or scalable security has been previously discussed [22, 27, 32, 72], but it is more precisely defined in terms of generalized weak security in our work. Since weak security is also inherent in secure network coding, it follows that secure network coding possesses USP capability as well.

Secure network coding is a class of network coding used for cryptographic purposes. The earlier purpose of network coding is to facilitate multicast transmission. Although both deterministic and random network coding can improve the multicast rate, only deterministic network coding can guarantee max-flow transmission. Another advantage of deterministic network coding is that we are better in control of the unequal-erasure-protection (UEP) capability, as discussed in Chapter 4. This UEP property can lead to conflicts among the multicast receivers in terms of received data quality. When the conflicts are considered as economic problems of resource distribution, an auction algorithm can be used to resolve them, as proposed in Chapter 4. If they are considered as political problems, a voting algorithm could be an interesting solution.

In practice, network coding will probably be used in a system in which channel coding is present. A problem occurs when the channel coding performance depends on the degree distribution of the codes, which may be affected by network coding and erasures in the network. A particular problem in LT-codes is discussed in Chapter 5 and a cooperative buffering scheme is proposed as a solution.

In conclusion, this thesis highlights several interrelationships among several subjects in the fields of coding and cryptography. We pictorially summarize it in Fig. 8.1. Apart from this, network protocol aspects are also considered for WPSG in generalized networks in Chapter 6, showing that network coding can enhance the security of the WPSG protocol.

8.2 Future Works

In the following, we shortly address some possibilities for future research.

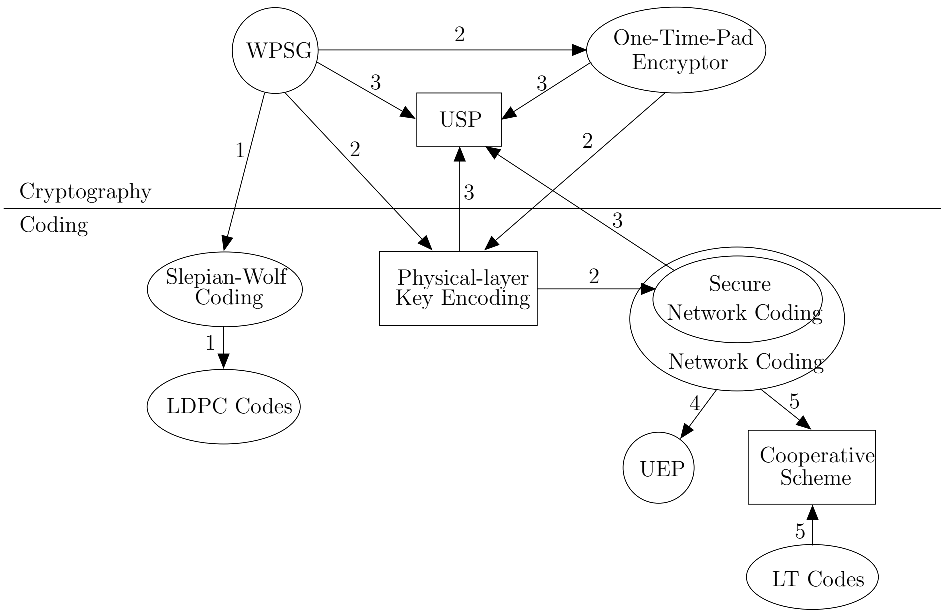

Figure 8.1: Interrelationships among mentioned subjects

8.2.1 Network Coding for WPSG

Apart from our research project, there exists an important contribution on one-relay WPSG made by Shimizu et al.. They show that network coding helps make one-relay WPSG more secure [74]. Figure 8.2(b) gives an illustration of their scheme, which they call "multiple-access amplify-and-forward (MA-AF)," as compared with the normal amplify-and-forward scheme in Fig. 8.2(a). For simplicity, we do not consider the effect of noise here and assume that the amplification factor at the relay is 1. The effects of these two factors are discussed in detail in [74].

The normal amplify-and-forward scheme is the same as Protocol 1 in Section 2.2.10, which is the least secure. The relay, after receiving the pilot packet x multiplied by the channel gain from Alice in the first time slot, amplifies and forwards it to Bob in the second one. The process repeats itself in the third and fourth time slot with the roles of Alice and Bob interchanged.

In the first time-slot of the MA-AF scheme, Alice receives xh_{ar} whereas Bob receives xh_{rb}, where h_{ar} and h_{rb} are Alice-relay and relay-Bob channel gains, respectively. In the second time slot, the signal from Alice and Bob adds together such that the relay receives $x(h_{ar} + h_{rb})$. The relay then forwards this to Alice and Bob in the third time slot. Alice

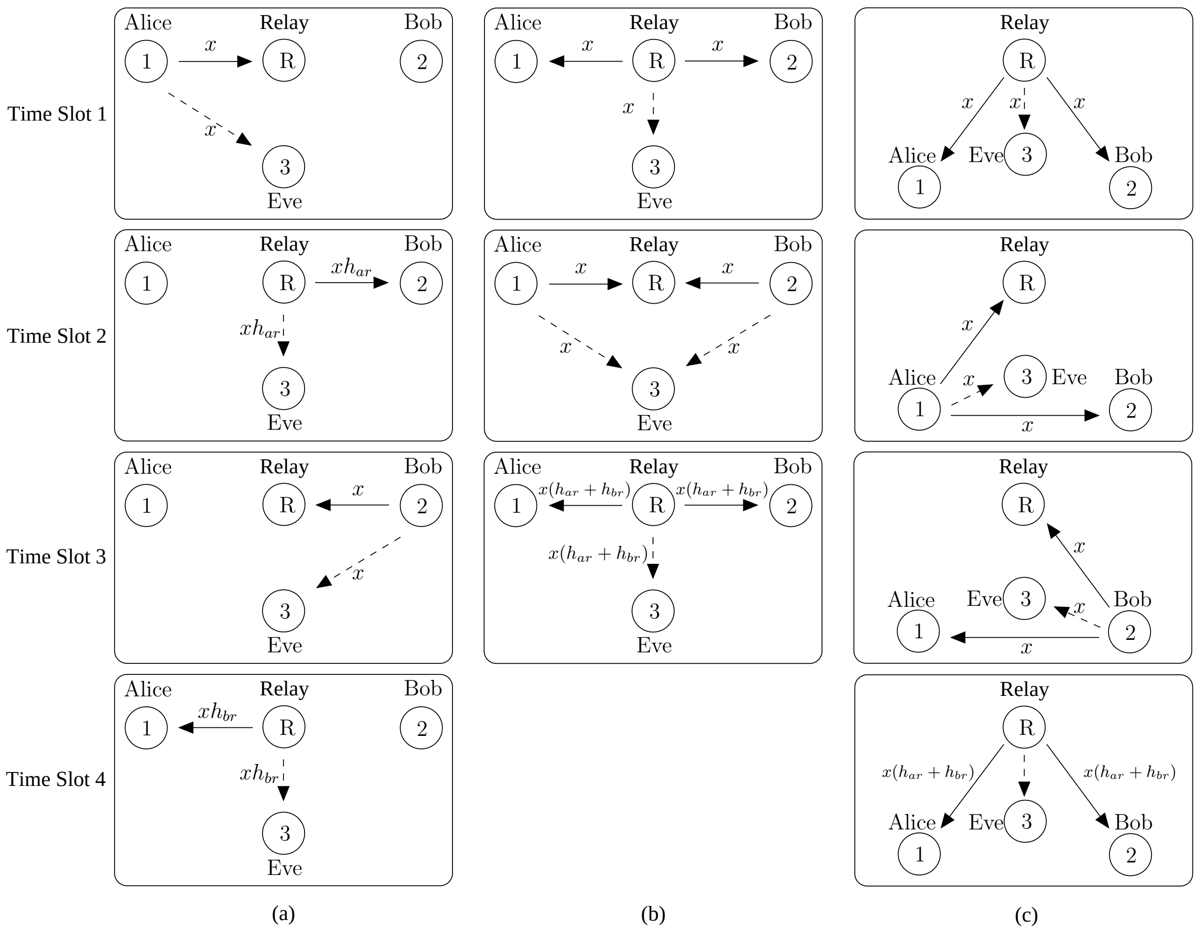

Figure 8.2: One-relay WPSG schemes, each arrow representing a transmitted signal: (a) amplify and forward, (b) MA-AF, (c) our proposed scheme

and Bob can then derive the overall channel gain $h_{ar}h_{rb}$ from the signal they receive in the first and the third time slots. Since the channel information sent out by the relay is neither h_{ar} nor h_{rb} but a network-coded combination, which is $h_{ar} + h_{rb}$, the enemy will find it more difficult to derive $h_{ar}h_{rb}$.

Investigation of Different Network Coding and Relaying Patterns

When Alice and Bob are within each other's transmission range, the efficiency of key generation can be enhanced as shown by our proposed scheme in Fig. 8.2(c). With an extension by one time slot, Alice and Bob can now generate a secret key from two paths, the direct one and the relayed one.

Apart from the triangular shape in Fig. 8.2(c), it is interesting to investigate the graphs with other shapes, such as a quadrilateral or a polygon in general, and compare the key generation efficiency. Also, one might design a scheme to generate secret keys not only for Alice and Bob, but also for the relays. One may further ask such graph-theoretic questions as how the convexity of the shape, or the completeness of the graph, affect key generation. (A shape is convex if all the points along the straight line connecting any two points within it lie inside. A graph is complete if every pair of nodes is connected by an edge.)

Generalizing the MA-AF Scheme

When the distance between Alice and Bob is larger, we need more than one relay and thus a generalized network coding scheme. The main question is whether a corresponding scheme can be derived by using Shimizu's procedure as a building block, and how.

Protocols for WPSG with Network Coding

If some local key is generated by the triangular geometry in Fig. 8.2(c), how can we make it compatible with the key generation schemes using different geometry and network coding patterns in other locations?

8.2.2 Effects of Mobility on Key Extension, Regeneration Rate, and Protocols

Key Extension and Regeneration Rate

Node mobility in a wireless network has profound effects on the key length and key regeneration rate due to two reasons. The obvious one is that the faster the nodes move, the faster the change in channel parameters and therefore the higher the key regeneration rate. The less obvious reason is that the mobility model affects the spatial node distribution. According to Bettstetter, Resta, and Santi, the node distribution $f_{\mathbf{x}}(\mathbf{x})$ of the random way point (RWP) mobility model, where $\mathbf{x} = (x, y)$ is a coordinate in two-dimensional space, consists of three components such that [12]

$$f_{\mathbf{x}}(\mathbf{x}) = f_s(\mathbf{x}) + f_p(\mathbf{x}) + f_m(\mathbf{x}) \ . \tag{8.1}$$

The nodes that remain static for the whole simulation time account for the static component $f_s(\mathbf{x})$. Those who are pausing between their moves make up the pause component $f_p(\mathbf{x})$, whereas those who are moving are responsible for the mobility component $f_m(\mathbf{x})$. In the RWP model, the distribution eventually reaches the steady state after some simulation time. The analytical expression of each component in the distribution is discussed in detail in [12]. Figure 8.3 shows the mobility component normalized such that the integral over the one-unit-squared region equals one.

We aim at deriving some empirical results from simulation regarding key regeneration rate as well as the optimal channel sampling rate as functions of the node mobility.

Protocols

As the nodes move and the topology changes, some wireless channels used for key generation are disconnected and other channels are added to replace them. We therefore need an adaptive protocol to determine the followings: 1) When will a channel be dropped from being used for key generation? 2) Which nodes are authorized to drop it? 3) How

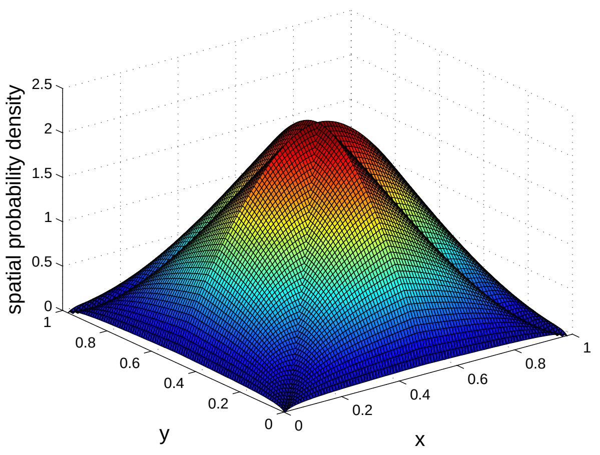

Figure 8.3: The normalized mobility component of the spatial node distribution in the RWP model

do other nodes know that the channel is dropped?

8.2.3 Economics of UEP Network Coding

The auction problem considered in our previous work [3] is only one specific economic problem among many. There are two more problems that we would like to investigate, the bargaining problem and the hierarchical network coding game.

The Bargaining Problem of UEP Network Coding

The bargaining problem is a non-zero-sum game which allows some cooperation among players. Let us consider network coding in a butterfly network in Fig. 8.4(a). Our knowledge about UEP network coding tells us that D obtains better data quality if b_1 is of higher priority than b_2 and every edge has the same erasure probability. Indeed, D may have won an auction over E to obtain this network coding pattern. However, D and E may reach an agreement that, when the edge AB alone is failing or having lots of erasures, the network coding pattern in Fig. 8.4(b) is used instead. This is beneficial for both D and E.

We aim at exploring the conditions under which bargaining is beneficial as well as the optimal bargain in numerical values for each receiver in a generalized network.

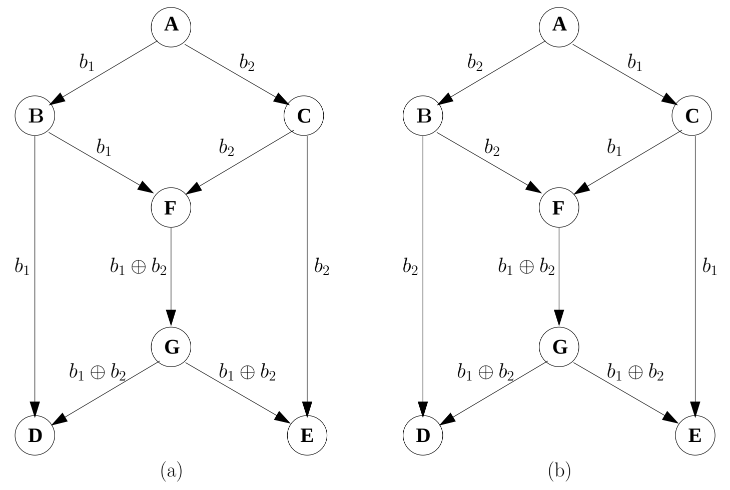

Figure 8.4: Two patterns of network coding in a butterfly network

The Hierarchical Network Coding Game

In more complicated networks, transmission paths from one source to certain sinks may need to pass some intermediate nodes which are also information sources themselves, as shown in Fig. 8.5. In such a case, network coding functions cannot be assigned by a single source. Instead, every source plays in a Stackelberg game based on von Stackelberg's "Marktform und Gleichgewicht [24]." According to Fig. 8.5, the source node A is considered the leader in the game since it has to make the first decision about global encoding kernels (GEKs) used for the edges AB and BC. After that, B and C can observe A's action and derive their optimal strategies.

Since more than one source node is considered, this game approach can be considered as a generalization of our previous analyses in [3, 5].

8.2.4 Joint LT-Network Coding

We would like to extend, generalize, and synthesize our previous works in UEP network coding [3, 5] and joint LT-network coding [2]. We aim at finding joint LT-network coding solutions in an arbitrary network instead of just a butterfly network, as well as introducing the UEP concept into joint LT-network coding.

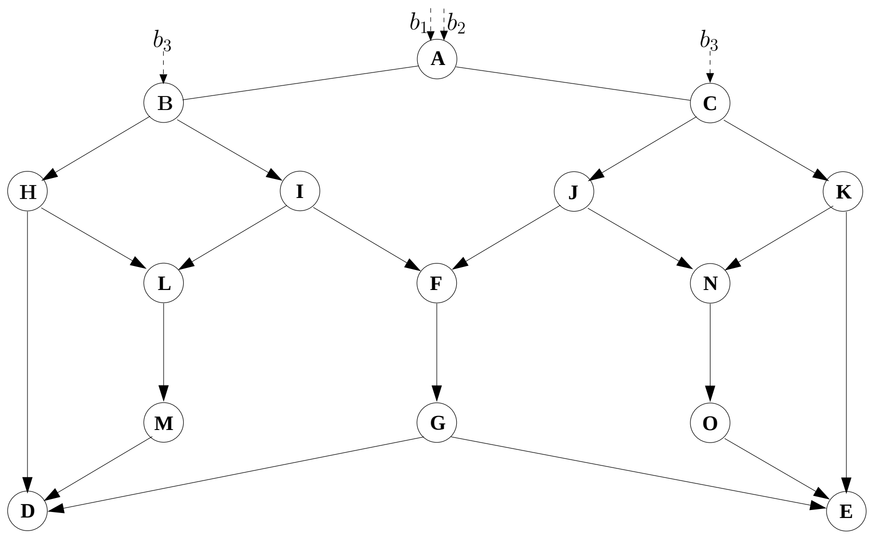

Figure 8.5: A double-butterfly network illustrating hierarchical network coding

Bibliography

[1] 3GPP, “3GPP; Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service; Protocol and Codecs,” 26.346 v6.3.0 (2005-12).

[2] A. Limmanee and W. Henkel, “A Cooperative Scheme for Shaping Degree Distribution of LT-Coded Symbols in Network Coding Multicast,” *Int. ITG Conf. Source and Channel Coding*, Siegen, Jan. 2010.

[3] A. Limmanee and W. Henkel, “Ascending-bid auction for unequal-erasure-protected network coding,” *IEEE Information Theory Workshop*, Taormina, Oct. 2009.

[4] A. Limmanee and W. Henkel, “Secure Physical-layer Key Generation Protocol and Key Encoding in Wireless Communications,” *IEEE GLOBECOM Workshop on Heterogeneous, Multi-hop Wireless and Mobile Networks*, Miami, Dec. 2010.

[5] A. Limmanee, W. Henkel, “UEP Network Coding for Scalable Data,” 5^{th} *Int. Symp. on Turbo Codes and Related Topics*, Lausanne, Sep. 2008.

[6] A.P. Lerner, *The Economics of Control*, Augustus M. Kelley Publishers, NY, 1970.

[7] A.S. Tanenbaum, *Computer Networks*, Prentice Hall PTR, New Jersey, 2003.

[8] A. Sayeed and A. Perrig, “Secure Wireless Communications: Secret Keys Through Multipath,” *Proc. 2008 IEEE Int. Conf. Acoustics, Apeech, and Signal Processing*, Las Vegas, Mar.-Apr. 2008.

[9] A. Shamir, “How to Share a Secret,” *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, Nov. 1979.

[10] A. Shokrollahi, “Raptor Codes,” *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2551-2567, Jun. 2006.

[11] A. Shokrollahi, “Raptor Codes,” *Int. Symp. Inform. Theory (ISIT 2004)*, Chicago, Jun.-Jul. 2004.

[12] C. Bettstetter, G. Resta, and P. Santi, “The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks,” *IEEE Trans. Mobile Computing*, vol. 2, no. 3, pp. 257-269, Jul.-Sep., 2003.

[13] C.E. Perkins and E.M. Royer, “Ad-hoc On-demand Distance Vector Routing,” *IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, Feb. 1999.

[14] C. Fragouli, E. Soljanin, and A. Shokrollahi, “Network Coding as a Coloring Problem,” in *Proc. Conf. Information Sciences and Systems*, Princeton, NJ, Mar. 2004.

[15] C. Hausl and J. Hagenauer, “Iterative Network and Channel Decoding for The Two-way Relay Channel,” *IEEE Int. Conf. Communications*, Istanbul, Jun. 2006.

[16] C. Yao, J.K. Zao, C.-H Wang, S.-Y.R. Li, N.A. Claude, and K.-K. Yen, “On Separation Vectors of Static Linear Network Codes with UEP Capability,” *Int. Symp. Network Coding*, Beijing, Jul. 2011.

[17] D.P. Bertsekas, *Network Optimization: Continuous and Discrete Models*, Athena Scientific, Belmont, Massachusetts, 1998.

[18] D. Slepian and J.K. Wolf, “Noiseless Coding of Correlated Information Sources,” *IEEE Trans. on Information Theory*, vol. 19, no. 4, pp. 471-480, Jul. 1973.

[19] E. van Damme and S. Hurkens, “Endogeneous Stackelberg Leadership,” *Games and Economic Behavior*, vol. 28, iss. 1, pp. 105-129, Jul. 1999.

[20] E.W. Weisstein, “Normal Product Distribution.” From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/NormalProductDistribution.html

[21] F.A. Hayek, "Competition as A Discovery Procedure," *Quarterly journal of Austrian economics*, vol. 5, iss. 3, pp. 9-24, Sep. 2002.

[22] G.A. Spanos and T.B. Maples, "Performance Study of a Selective Encryption Scheme for the Security of Networked Real-time Video," *Int. Conf. Computer and Communications*, Las Vegas, 1995.

[23] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103-1120, Sep. 2007.

[24] H. von Stackelberg, *Marktform und Gleichgewicht*, Springer-Verlag, Berlin, 1934.

[25] http://personal.ie.cuhk.edu.hk/ITIP/ISIT02/secure.ps

[26] I.A. Glover and P.M. Grant, *Digital Communications*, Prentice Hall, 2004.

[27] I. Agi and L. Gong, "An Empirical Study of Secure MPEG Video Transmissions," *ISOC Symp. Network and Distributed System Security*, San Diego, 1996.

[28] J.F. Nash, Jr., "The Bargaining Problem" *Econometrica*, vol. 18, no. 2, pp. 155-162, Apr. 1950.

[29] J. Feldman, T. Malkin, R.A. Servedio, and C. Stein, "On the Capacity of Secure Network Coding," *Proc. 42nd Annual Allerton Conf. Communication, Control, and Computing*, Sep. 2004.

[30] J.G. Proakis, *Digital Communications*, McGraw-Hill, Boston, 2001.

[31] J.L. Massey, "An Introduction to Contemporary Cryptology," *Proceedings of the IEEE*, vol. 76, no. 5, May 1998.

[32] J. Meyer and F. Gadegast, "Security Mechanisms for Multimedia Data with the Example MPEG-1 Video," SECMPEG Project Report, 1995.

[33] J.W. Wallace and R.K. Sharma, "Automatic Secret Keys From Reciprocal MIMO Wireless Channels: Measurement and Analysis," *IEEE Tran. Information Forensics and Security*, vol. 5, no. 3, pp. 381-391, Sep. 2010.

[34] J. Wallace, "Secure Physical Layer Key Generation Schemes: Performance amd Information Theoretic Limits," *IEEE Int. Conf. Communications*, Dresden, Jun. 2009.

[35] K. Bhattad and K.R. Narayanan, "Weakly Secure Network Coding," in *Proc. NetCod 2005*, Riva del Garda, Apr. 2005.

[36] K. Hassan and W. Henkel, "UEP MIMO-OFDM with Beamforming-Combining for Imperfect Channel Information," *OFDM-Workshop 2007*, Hamburg, Aug. 2007.

[37] K. Hassan and W. Henkel, "UEP with Eigen Beamforming for Partial Channel Information MIMO-OFDM," *2007 IEEE Sarnoff Symposium*, Princeton, Apr.-May 2007.

[38] L.H. Ozarow and A.D. Wyner, "Wire-Tap Channel II," *AT&T Bell Laboratories technical journal*, vol. 63, no. 10, pp. 2135-2157, 1984.

[39] L.M. Ausubel, "An Efficient Ascending-bid Auction for Multiple Objects," *The American Economic Review*, vol. 94, no. 5, pp. 1452-1475, Dec. 2004.

[40] L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols," *Computer Communication Review*, Apr. 1997.

[41] L. Sassatelli, W. Henkel, and D. Declercq, "Check-Irregular LDPC Codes for Unequal Error Protection under Iterative Decoding," *4th Int. Symp. Turbo Codes & Related Topics in connection with the 6th Int. ITG-Conf. on Source and Channel Coding*, Munich, April 4-7, 2006.

[42] M. Bloch, J. Barros, M.R.D. Rodrigues, and S.W. McLaughlin, "Wireless Information-Theoretic Security," *IEEE Trans. Information Theory*, vol. 54, no. 6, pp. 2515-2534, Jun. 2008.

[43] M. Friedman and L.J. Savage, "The Utility Analysis of Choices Involving Risk," *The Journal of Political Economy*, vol. 56, no. 6, pp. 279-304, 1948.

[44] M. Luby, "LT codes," *Proc. 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002.

[45] N. Cai and R.W. Yeung, "Network Error Correction, Part II: Lower Bounds" *Communications in Information and Systems*, vol. 6, no 1, pp. 37-54, 2006.

[46] N. Cai and R.W. Yeung, "Secure Network Coding," *Int. Symp. Information Theory*, Lausanne, Jun.-Jul. 2002.

[47] N. Rahnavard, B.N. Vellambi, and F. Fekri, "Rateless Codes with Unequal Error Protection Property," *IEEE Trans. Inform. Theory*, Vol. 53, no 4, pp. 1521-1532, Apr. 2007.

[48] N. von Deetzen and S. Sandberg, "Design of Unequal Error Protection LDPC Codes for Higher Order Constellations," *IEEE International Conference on Communications (ICC)*, Glasgow, Jun. 2007.

[49] N. von Deetzen and W. Henkel, "On Code Design for Unequal Error Protection Multilevel Coding," *7th ITG Conference on Source and Channel Coding 2008 (SCC'08)*, Ulm, Jan. 2008.

[50] N. von Deetzen and W. Henkel, "Unequal Error Protection Multilevel Codes and Hierarchical Modulation for Multimedia Transmission," *Int. Symp. Information Theory (ISIT 2008)*, Toronto, Jul. 2008.

[51] P.A. Chou, Y. Wu, and K. Jain, "Practical Network Coding," 51^{st} *Allerton Conf. Communication, Control, and Computing*, Oct. 2003.

[52] P. Elias, "Coding for Two Noisy Channels," 3^{rd} *London Symp. Information Theory*, London, 1955.

[53] P. Popovski and H. Yomo, "Physical Network Coding in Two-way Wireless Relay Channels," *IEEE Int. Conf. Communications*, Jun. 2007.

[54] P. Popovski and H. Yomo, "Wireless Network Coding by Amplify-and-forward for Bi-directional Traffic Flows," *IEEE Communication Letter*, vol. 11, no. 1, pp. 16-18, Jan. 2007.

[55] R. Ahlswede, N. Cai, S.-Y.R. Li, and R.W. Yeung, "Network Information Flow," IEEE Trans. Inform. Theory, vol. 46, no. 4, pp. 1204-1216, Jul. 2000.

[56] R. Diestel, *Graph Theory*, Springer-Verlag, Berlin, 2006.

[57] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, New Jersey, 1993.

[58] R. Koetter and F.R. Kschischang, "Coding for Errors and Erasures in Random Network Coding," *IEEE Trans. Inform. Theory*, vol. 54, no. 8, pp. 3579-3591, Aug. 2008.

[59] R. Koetter and M. Médard, "An Algebraic Approach to Network Coding," *IEEE/ACM Trans. Networking*, vol. 11, pp. 782-795, Oct. 2003.

[60] R.W. Yeung and N. Cai, "Network Error Correction, Part I: Basic Concepts and Upper Bounds," *Communications in Information and Systems*, vol. 6, no. 1, pp. 19-36, 2006.

[61] R.W. Yeung, S.-Y.R. Li, N. Cai, and Z. Zhang, "Network Coding Theory," *Foundation and Trends in Communications and Information Theory*, vol. 2, nos. 4 and 5, pp. 241-381, 2005.

[62] S. Fu, T. Zhang, and M. Colef, "Secrecy in Two-way Relay Systems," *IEEE Global Communications Conf. (GLOBECOM)*, Miami, Dec. 2010.

[63] S. Jaggi, P. Sanders, P.A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, “Polynomial Time Algorithms for Multicast Network Code Construction,” *IEEE Trans. Inform. Theory*, vol. 51, no. 6, pp. 1973-1982, Jun. 2005.

[64] S.-K. Chang, K.-C. Yang, and J.-S. Wang, “Unequal-Protected LT Code for Layered Video Streaming,” *IEEE Int. Conf. on Communications*, Beijing, May 2008.

[65] S. Katti, and D. Katabi, “Embracing Wireless Interference: Analog Network Coding,” *Proc. Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 397-408, 2007.

[66] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, “XORs in The Air: Practical Wireless Network Coding,” *IEEE/ACM Trans. Networking*, vol. 16, no. 3, pp. 497-510, Jun. 2008.

[67] S. Puducheri, J. Kliewer, and T.E. Fuja, “Distributed LT Codes,” *IEEE Int. Symp. Information Theory*, Seattle, Jul. 2006.

[68] S. Sandberg and N. von Deetzen, “Design of Bandwidth-Efficient Unequal Error Protection LDPC Codes,” *IEEE Trans. Communications*, vol. 58, no. 3, pp. 802-811, Mar. 2010.

[69] S.-Y.R. Li, R.W. Yeung, and N. Cai, “Linear Network Coding,” *IEEE Trans. Inform. Theory*, vol. 49, no. 2, pp. 371-381, Feb. 2003.

[70] S. Zhang, S.C. Liew, and P.P. Lam, “Hot Topic: Physical-layer Network Coding,” *Proc. MobiCom’06*, pp. 358-365, ACM, 2006.

[71] T. Aono, K. Higuchi, T. Ohira, B. Komiyama, and H. Sasaoka, “Wireless Secret Key Generation Exploiting Reactance-Domain Scalar Response of Multipath Fading Channels,” *IEEE Trans. Antennas and Propagation*, vol. 53, no. 11, pp. 3776-3784, Nov. 2005.

[72] T. Kunkelmann and R. Rainema, "A Scalable Security Architecture for multimedia communication standards," *Int. Conf. Multimedia Computing and Systems*, Ottawa, Jun. 1997.

[73] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, John Wiley & Sons, New Jersey, 2006.

[74] T. Shimizu, H. Iwai, H. Sasaoka, and A. Paulraj, "Secret Key Agreement Based on Radio Propagation Characteristics in Two-way Relaying Systems," *IEEE Global Communications Conf. (GLOBECOM)*, Miami, Dec. 2010.

[75] W. Cheng, L. Yu, F. Xiong, and W. Wang, "Trusted Network Coding in Wireless Ad Hoc Networks," *IEEE Global Communications Conf. (GLOBECOM)*, Miami, Dec. 2010.

[76] W. Henkel and J. Wallace, "Unequal Error Protection and Security Approaches in Wireless and Network Coding- A Study of Continuous and Discrete Number Designs," Extension Application in the Framework of a "Schwerpunktprogramm" (HE-3654/11-2), 2011.

[77] W. Henkel and K. Hassan, "OFDM (DMT) Bit and Power Loading for Unequal Error Protection," *OFDM-Workshop 2006*, Hamburg, Aug. 2006.

[78] W. Henkel and N. von Deetzen, "Path Pruning for Unequal Error Protection Turbo Codes," *2006 International Zurich Seminar on Communications*, Zurich, Feb. 2006.

[79] W. Kocay and D.L. Kreher, *Graphs, Algorithms, and Optimization*, Chapman & Hall/CRC, Florida, 2005.

[80] W. Vickrey, "Counterspeculation, Auctions, and Competitive Sealed Tenders," *Journal of Finance*, vol. 16, no. 1, pp. 8-37, Mar. 1961.

[81] X. Sun, X. Wu, C. Zhao, M. Jiang, and W. Xu, “Slepian-Wolf Coding for Reconciliation of Physical Layer Secret Keys,” *Proc. IEEE WCNC 2010*, Sydney, Apr. 2010.

[82] Y. Lin, B. Li, and B. Liang, “Differentiated Data Persistence with Priority Random Linear Codes,” 27th *Int. Conf. on Distributed Computing Systems*, Toronto, Jun. 2007.

[83] Y. Wu, P.A. Chou, and S.-Y. Kung, “Information Exchange in Wireless Networks with Network Coding and Physical-layer Broadcast,” *2005 Conf. Information Sciences and Systems*, Mar. 2005.

[84] Y. Wu, P.A. Chou, and S.-Y. Kung, “Information Exchange in Wireless Networks with Network Coding and Physical-layer Broadcast,” *Technical Report MSR-TR-2004-78*, Aug. 2004.