TOWARDS ENDURABLE, RELIABLE AND SECURE FLASH MEMORIES – A CODING THEORY APPLICATION

A Dissertation

by

QING LI

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Anxiao (Andrew) Jiang
Committee Members,	Tie Liu
	Andreas Klappenecker
	Jennifer Welch
Department Head,	Dilma Da Silva

December 2015

Major Subject: Computer Science

ABSTRACT

Storage systems are experiencing a historical paradigm shift from hard disk to non-volatile memories due to its advantages such as higher density, smaller size and non-volatility. On the other hand, Solid Storage Disk (SSD) also poses critical challenges to application and system designers. The first challenge is called endurance. Endurance means flash memory can only experience a limited number of program/erase cycles, and after that the cell quality degradation can no longer be accommodated by the memory system fault tolerance capacity. The second challenge is called reliability, which means flash cells are sensitive to various noise and disturbs, i.e., data may change unintentionally after experiencing noise/disturbs. The third challenge is called security, which means it is impossible or costly to delete files from flash memory securely without leaking information to possible eavesdroppers.

In this dissertation, we first study noise modeling and capacity analysis for NAND flash memories (which is the most popular flash memory in market), which gains us some insight on how flash memories are working and their unique noise. Second, based on the characteristics of content-replication codewords in flash memories, we propose a joint decoder to enhance the flash memory reliability. Third, we explore data representation schemes in flash memories and optimal rewriting code constructions in order to solve the endurance problem. Fourth, in order to make our rewriting code more practical, we study noisy write-efficient memories and Write-Once Memory (WOM) codes against inter-cell interference in NAND memories. Finally, motivated by the secure deletion problem in flash memories, we study coding schemes to solve both the endurance and the security issues in flash memories. This work presents a series of information theory and coding theory research studies on the aforesaid three critical issues, and shows that how coding

theory can be utilized to address these challenges.

ACKNOWLEDGEMENTS

Writing this dissertation is a pleasant experience of looking back at my five-year journey of those days and nights spent at Texas A&M University. As I finish it, I am reminded of people who accompanied me during my Ph.D. study, and I would like to take this opportunity to express my gratitude to them for their love, patience and support over the years.

First and foremost, I would like to express my sincere thanks to my advisor, Prof. Anxiao (Andrew) Jiang. As his graduate student, I have been working with him during the last five years. During those days, he showed me how to become a good researcher. As a researcher, he always targets innovative ideas, works extremely hard and is always optimistic. Over those years, he guided me throughout my Ph.D. study and offered me tremendous help. As an advisor, he always gave me freedom to explore interesting and challenging ideas and offered me support and encouragement. Without his support and help, it would have been impossible for me to reach this point.

I would like to express my gratitude to Dr. Hao Zhong at SanDisk and Dr. Erich. F. Haratsch at Seagate Technology. Dr. Hao Zhong offered me the 2014 summer intern at Fusion-io (now part of SanDisk), which presented me a wonderful and interesting world that is very different from university. Dr. Erich F. Haratsch offered me the opportunity to work closely with industry and provided me with some helpful insights into flash memories.

I am also very thankful to my lab-mates and friends, Dr. Yue Wang, Dr. Yue Li, Dr. Jialong Zhang, Dr. Yi Cui, Dr. Huihua Zhao, Dr. Huanlin Zhu, Dr. Zhongzheng Liu, Dr. Wei Li, Dr. Xiong Pu and Dr. Youfeng Zhang at Texas A&M University, Dr. Yangyang Pan at SanDisk, and Dr. Guiqiang Dong at HGST. We have had many inspiring discussions

and joyful moments, which makes the Ph.D. study quite enjoyable and unforgettable.

I would like to thank the committee members of my Ph.D. dissertation: Prof. Tie Liu, Prof. Andreas Klappenecker and Prof. Jennifer L. Welch for serving on my committee and for providing much helpful advice.

Last but not least, I would like to thank my parents, Aiqing Li and Guihua Dai, for offering me endless love, tremendous support and encouragement. I am also specially indebted to my elder brother, Xiaoming Li, and my sister-in-law, Juan He, for taking care of my parents and for their love. I am grateful to my nephew, Yanbing Li, for bringing me joy, warmth and hope.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Non-volatile memories are becoming ubiquitous due to their advantages such as higher density, scaling size and non-volatility, etc. However, challenges also exist, among which the most serious ones are the endurance, the reliability and the security. In this dissertation we present a series of information theory and coding theory studies on those challenges.

1.1 Opportunities and Challenges

The unit of flash memory is a cell. Data is represented by the number of charge in a cell. A flash chip is composed of blocks, each block consists of pages, and each page consists of cells. There are three basic operations on flash memory cells: program/write, read and erase. The granularity of read/program and erasure is a page and a block, respectively.

The writes on flash memory are made in *out-of-place* fas hion, i.e., instead of modifying existing data, the previous data is marked as *invalid* and the new data is written in another block.

There are three main challenges in flash memories:

- *Endurance*. Endurance means flash memory can only experience a limited number of program/erase cycles, beyond which the cell quality degradation can no longer be accommodated by the memory system fault tolerance capacity.

- *Reliability*. Reliability means flash cells are sensitive to various noise and disturbs, i.e., data may change unintentionally after experiencing noise/disturbs.

- *Security*. Security means it is impossible or costly to delete files from flash memory securely without leaking information to possible eavesdroppers.

Those problem are serious especially when flash memories are scaling down, and become even worse with the era of three-dimention flash memories. On the other hand, those

problems are hard to solve by traditional methods due to the dramatic differences between flash memories and traditional storage media. This research not only provides a feasible solution to flash memory challenges but also shows how it expands the coding theory scope.

In this dissertation we present a series of research studies on the aforesaid three critical issues to address these challenges. Our research work makes the following contributions.

1.2 Research Contributions

1.2.1 Noise modeling and capacity analysis for NAND flash memories

In order to solve the reliability challenge in flash memories, we first have to understand various types of error mechanisms in flash memories, which have drastically different characteristics from traditional communication channels. Understanding the error models is necessary for developing better coding schemes in the complex practical setting. This work endeavors to survey the noise and disturbs in NAND flash memories, and construct channel models for them. The capacity of flash memory under these channel models is analyzed, particularly regarding capacity degradation with flash operations, the trade-off of sub-thresholds for soft cell-level information, and the importance of dynamic thresholds.

1.2.2 Joint decoding of content-replication codes for flash memories

In order to solve the reliability challenge in flash memories, we present the content-replication codeword problem, and it leads to a novel joint decoding scheme proposed. We focus on the joint decoding algorithm designs and study their theoretical decoding performances in this work. The proposed scheme is new for storage systems especially for flash memories, and we show their reliability can be enhanced by increasing the diversity of error-correcting codes.

1.2.3 Compressed rank modulation

In order to solve the endurance challenge in flash memories, we present a new data representation scheme, Compressed Rank Modulation (CRM). CRM stores information in the multiset permutation induced by the charge levels of cells. The only allowed charge-placement method is the *minimal-push-up* aiming to minimize the increase of highest charge levels. CRM achieves a higher capacity and a longer endurance.

1.2.4 Polar codes are optimal for Write-Efficient Memories

In order to present efficient code constructions for Write-efficient memory (WEM) to solve the endurance challenge in flash memories, we conduct this research. WEM is a model for storing and updating information on a rewritable medium with constraints. CRM is an example of WEM. A optimal and efficient coding scheme for WEM using polar codes is designed. The coding scheme achieves capacity.

1.2.5 Coding on noisy Write-Efficient Memories

To jointly solve the endurance and the reliability problem, we propose a new coding model, noisy WEM. We construct an efficient coding scheme for it. Its decoding and rewriting operations can be done efficiently.

1.2.6 WOM codes against inter-cell interference in NAND memories

This is another work to jointly solve the endurance and the reliability problem, however, the error type here is specified as inter-cell interference. In this work, we study Write-Once Memories (WOM) codes against cell-to-cell interference. We derive bounds of the rewriting capacity of WOM codes based on the new WOM codes, Delta-WOM, and constrained codes. We also explore efficient WOM code constructions: one construction is based on our Diamond-WOM codes construction, which can be proven to approach its known rewriting capacity; the other one is based on constrained codes.

1.2.7 Coding on secure Write-Efficient Memories

Endurance and security are two serious challenges for non-volatile memories such as flash memories. Write-Efficient Memory (WEM) is an important rewriting code model to solve the endurance problem. Aiming at jointly solving the endurance and the security issues in non-volatile memories, this work focuses on rewriting code with a security constraint. To that end, a novel coding model, secure WEM, is proposed. We explore its rewriting-rate-equivocation region and its secrecy rewriting capacity in this work.

1.2.8 Polar codes for secure Write-Efficient Memories

This work is to present an optimal code construction for secure WEM to solve both the endurance and the secureity challenges in flash memories. This nested code construction is based on optimal code constructions for WEM and wiretap channel codes, and the code construction approaches the secrecy rewriting capacity for a large family of secure WEM.

1.3 Organization of the Dissertation

The rest of this dissertation is organized as follows. In Section 2, we discuss our theoretical study on the noise modeling in flash memories and its impact on the capacity. In Section 3, we present our work to improve the flash memory repliability through joint decoding of content-replicated codes. In Section 4, we discuss Compressed Rank Modulation. In Section 5, we show that Polar codes can be used to construct an optimal code construction for Write-Efficient Memories, and Write-Efficient Memories with error correcting ability is discussed in Section 6. In Section 7, we explore Write-Once Memroies against intra-cell interference. In Section 8, we present Secure Write-Efficient Memories and its efficient code construction in Section 9. Finally, conclusion is obtained in Section 10.

2. NOISE MODELING AND CAPACITY ANALYSIS FOR NANS FLASH MEMORIES

2.1 Introduction

Flash memories have become a significant storage technology, mainly due to their high speed, physical robustness and non-volatility. Various coding techniques have been developed for them, including codes for error correction [37, 77], rewriting [9, 76], rank modulation [36], etc. Despite their wide applications, flash memories are far from ideal. In particular, they have various reliability issues, some of which get more serious with each new generation of flash memories due to the scaling of flash cell sizes [13]. Flash memories have quite a few noise or disturb mechanisms, including retention errors, inter-cell interference, random noise, programming errors, read and write disturbs, and stuck cells [10, 12]. These mechanisms have quite different characteristics from traditional communication channels.

It is important to understand the channel models for the noise and disturbs in flash memories, in order to design better coding schemes in the complex practical setting. However, information-theoretical work on channel modeling for flash memories has been limited. This paper is an endeavor to survey the various noise and disturb mechanisms for NAND flash memories, and build their corresponding channel models. We analyze the capacity of flash memories under these models, and show how it evolves with read and write operations. While conventional storage media (e.g., magnetic and optical recording) typically have noise accumulated over time [54], in flash memories, significant noise accumulates with flash operations and causes the storage capacity to degrade. We also show that there is a trade-off for using sub-thresholds for obtaining more soft information on analog cell-levels due to read disturbs. Furthermore, as the noise in flash cell is highly

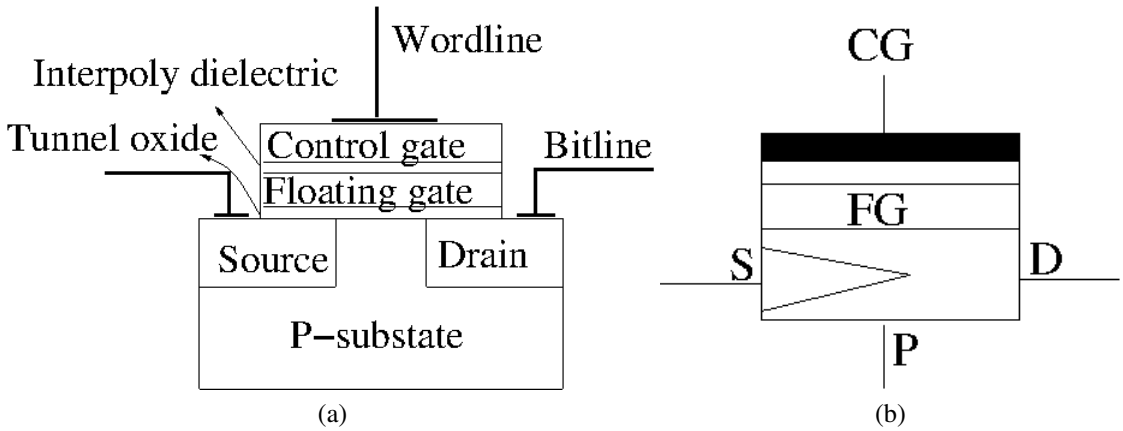


Figure 2.1: (a) The structure of NAND flash cell. (b) A symbol for NAND flash cell, where "CG", " FG", "S", "D" and "P" stand for control gate, floating gate, source, drain and P-substrate, respectively.

correlated and not symmetric, it is important to use dynamic thresholds to achieve higher capacity.

The rest of this work is organized as follows. In Section 2.2, we introduce the fundamental structures and operations in flash memories. In Section 2.3, we model various types of noise and disturbs. In Section 2.4, we analyze the special features of storage capacity in flash memories. In Section 2.5, we present concluding remarks.

2.2 Fundamental Concepts of Flash Memories

In this section, we briefly survey the fundamental concepts on NAND flash memories, which are necessary for understanding the channel models of noise and disturbs.

2.2.1 Structure and operations of flash memory cell

A flash memory cell is a MOS transistor with a floating-gate layer. Its structure is illustrated in Figure 2.1 (a). We represent it with the simplified symbol in Figure 2.1 (b).

A flash cell stores data by storing charge in its floating-gate layer. And the mount

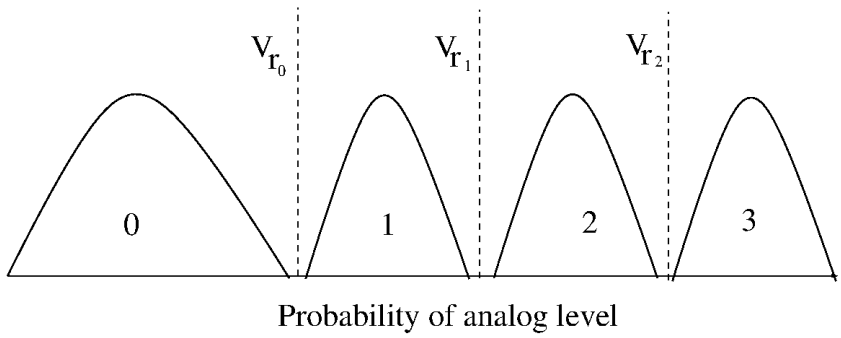


Figure 2.2: A typical distribution of the analog levels of MLC.

of charge affects its threshold voltage, which is a minimum required voltage added to CG to open the gate. When electrons are stored, the more electrons are trapped in the floating-gate layer, the higher the threshold voltage is. We call the analog value of a cell's threshold voltage its *analog level*. In practice, a cell's analog levels are quantized into discrete values to represent one or more bits. We denote the q discrete levels of a cell by level 0, 1, $\cdots$, $q-1$. When $q = 2$, 4, 8, the flash memory cells are called SLC (Single-Level Cell), MLC (Multi-Level Cell) and TLC (Triple-Level Cell), respectively. We illustrate a typical distribution of the analog levels of MLC in Figure 2.2. A cell's discrete level is read by comparing it to several reference levels. For q-level cells, $q-1$ reference levels are needed. (The three reference levels for MLC are shown as dotted vertical lines in Figure 2.2.) If more reference levels are used, more soft information about the analog levels can be obtained, which can be useful for coding schemes, but at the cost of longer read delays and more read disturbs.

There are three basic operations on a flash cell: *read*, *write* and *erase*. To read a cell's level, a reference voltage is applied to its control gate to see if the gate opens or not. (When $q > 2$, multiple such reads with different reference voltages may be needed.) To write a cell (also called programming, which means to inject charge into a cell), the

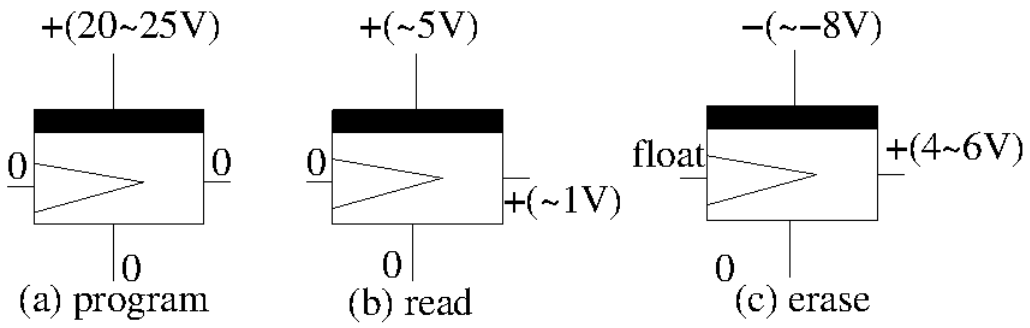


Figure 2.3: The estimates of voltage biases on S, CG, P and D during flash operations

Fowler-Nordheim (FN) [13] tunneling mechanism is used by applying a high voltage to the control gate. To erase a cell (which means to remove all stored charge from the cell), a high negative voltage is applied to the control gate. Some typical configurations of voltages applied to the four ends of a flash cell are shown in Figure 2.3 for read, write and erase, respectively.

2.2.2 Structure and operations of flash-cell array

The cells in a flash memory are organized as (often tens of thousands or more) blocks, where every block is a two dimensional array. We illustrate a block in Figure 2.4. There every vertical wire BL is called a bitline, and every horizontal wire WL is called a wordline. In addition, there are two horizontal wires BSL (Bitline Select Line) and GSL (Grounded Select Line). Each row of a page is called a page, which is the unit of read and write operations. A block usually has 32, 64 or more pages, and a page stores thousands of bits. An erase operation is applied to a whole block, and therefore called block erasure.

More specifically, the read, write and erase operations are performed as follows:

- *Read*. To read a page of cells, a positive voltage V_{read} is applied to the page's corresponding WL (wordline). For the other pages, a higher voltage V_{pass} is applied to their WLs.

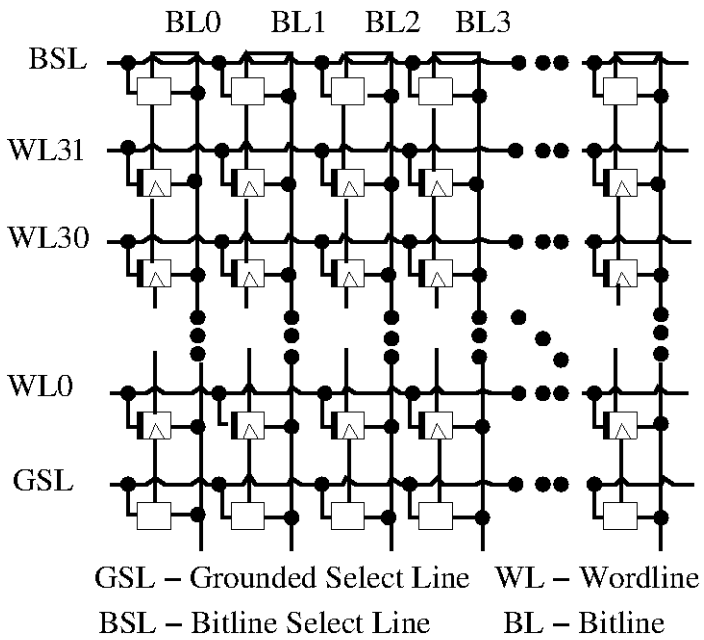
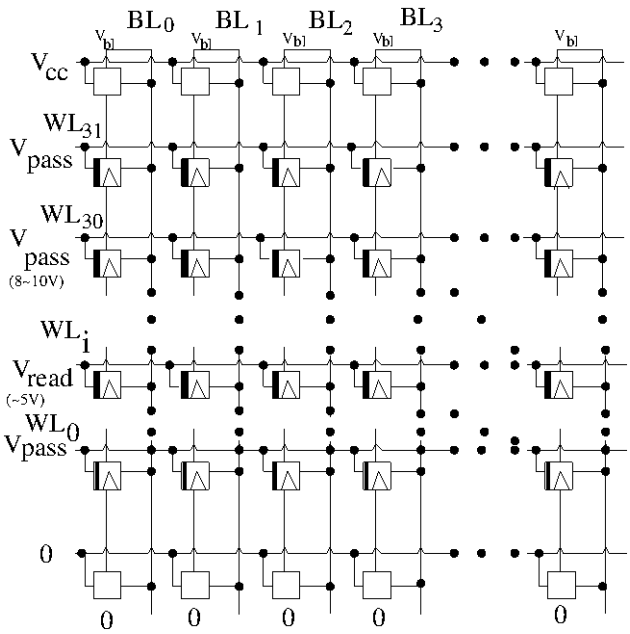


Figure 2.4: Structure of a NAND flash memory block

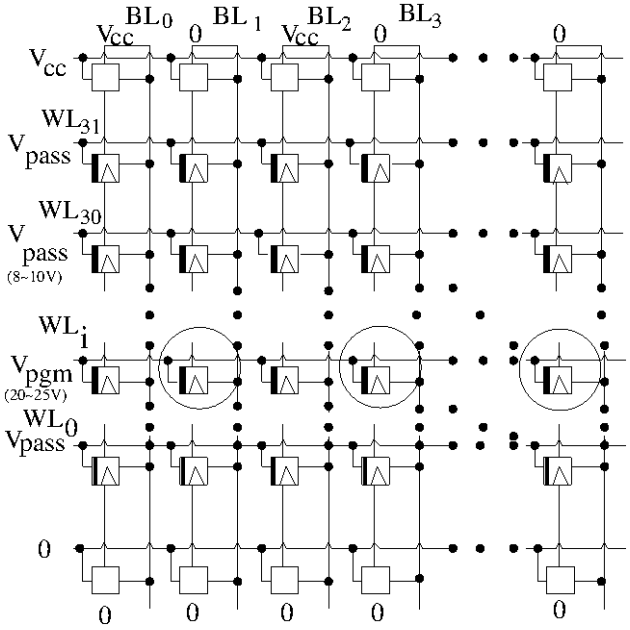
- *Write*. To write a page of cells (more specifically, a subset of the cells in the page), a high voltage V_{pgm} is applied to its corresponding WL. For the programmed cells in that page, their corresponding BLs are grounded; for the remaining cells, their corresponding BLs are set to a positive voltage V_{cc}. For the remaining pages, which are not programmed, a positive voltage V_{pass} is applied to their WLs.

- *Erase*. To erase a block, all the WLs are grounded, and a high positive voltage is applied to all the BLs.

We illustrate the typical voltage configurations for read and write in Figure 2.5 (a), (b), respectively. Please note that the exact voltage numbers may vary from company to company.



(a)



(b)

Figure 2.5: (a) A typical voltage configuration for the read operation. Here the i-th page shown in the figure is being read. (b) A typical voltage configuration for the write operation. Here the i-th page is being programmed. (In particular, the programmed cells are shown in circles)

Table 2.1: Basic notations

Notation	Meaning
q	Number of discrete levels of a cell
W	Number of WLs in a block
$\mathcal{W}_i$	The i-th WL in a block
B	Number of BLs in a block
$\mathcal{B}_j$	The j-th BL in a block
$c_{i,j}$	The cell in the i-th page (corresponding to the i-th WL $\mathcal{W}_i$) and the j-th column (corresponding to the j-th BL $\mathcal{B}_j$)
$V_{i,j}(0)$	The analog level of cell $c_{i,j}$ right after it is programmed
$V_{i,j}(t)$	The analog level of cell $c_{i,j}$ after time t has elapsed since it was programmed
$V_{i,j}$	A simplified notation of $V_{i,j}(t)$
V_i	The average analog level of the i-th discrete level

2.3 Channel Modeling for Errors in Flash Memories

In this section, we survey the various noise and disturb mechanisms, and present channel models for them.

2.3.1 Overview of error models

In this subsection, we briefly overview the errors in flash memories. Their details will be introduced later. Throughout the paper, a number of notations will be used; for convenience, we summarize them in Table 2.1.

The various types of noise and disturbs in flash memories include:

1. *Inaccurate programming*. When cells are programmed, the analog levels they obtain usually deviate from the target level. Even for cells programmed the same way, their obtained analog levels are typically different due to cell heterogeneity, the difference in their original analog levels after the previous erasure operation and other reasons (e.g., inter-cell interference).

2. *Retention error*. After cells are programmed, the charge stored in them leaks away

gradually. When cells experience more program/erase (P/E) cycles, their quality degrades, and charge leakage becomes more serious.

3. *Cell-to-cell interference*. There is coupling capacitance between adjacent cells. As a result, the analog level of a cell depends not only on its own storage charge, but also the charge in neighboring cells. For a cell $c_{i,j}$, the more charge its neighbors have, the higher its analog levels becomes due to the interference. The interference becomes particularly serious when the neighbors of $c_{i,j}$ are programmed after $c_{i,j}$ itself because that makes it impossible to program $c_{i,j}$ adaptively.

4. *Read disturb*. When the i-th page is read, the other pages are unintentionally and weakly programmed because of the positive voltage V_{pass} applied to their wordlines, making their analog levels higher.

5. *Program disturb*. When a page is programmed, – more specifically, when a subset of the cells in that page are programmed, – the other cells in that page are unintentionally and weakly programmed because of the voltage difference between their control gates and P-substrates, making their analog levels higher.

6. *Pass disturb*. When a page is programmed, the other pages are unintentionally and weakly programmed because of the positive voltage V_{pass} applied to their wordlines.

Flash memory cells also have random noise and stuck-at errors. The latter is caused by the degradation of cell quality, which makes it impossible to change the levels of stuck-at cells.

In the following, we analyze the errors in more detail, and present information-theoretic models.

2.3.2 Inaccurate programming

For cells of q levels, for $k = 0, 1, ..., q-1$, when we program a cell to the k-th level (for $k = 0$ it is actually the erasure state), let V_k denote the target analog level. For a cell $c_{i,j}$ programmed to the k-th level, let $V_{i,j}(0)$ denote its actual programmed analog level. We call

$$Z_k = V_{i,j}(0) - V_k, \tag{2.1}$$

the programming noise. For simplicity, we assume $Z_k \sim \mathcal{N}(0, \sigma_k)$ has a Gaussian distribution. Similar bell-shape models have appeared in [11, 74].

2.3.3 Retention error

It is reported in [17] that the number of leaked electrons depends on the leaking time t and the initial number of electrons $n(0)$. The number of electrons at time t, $n(t)$, can be modeled as $n(t) = n(0)e^{-\upsilon t}$, where υ is a constant parameter. This parameter υ can vary for cells. So for cell $c_{i,j}$, we use $\upsilon_{i,j}$ to denote its corresponding value of υ.

Note that the number of leaked electrons does not always strictly follow the above smooth function. Therefore, we use an additive noise Z_{re} to denote the corresponding noise term. Based on the linear relationship between the analog level and the number of electrons in the cell's FG (see [13] for details), we model $V_{i,j}(t)$ – the analog level of cell $c_{i,j}$ after time t has elapsed since it was programmed – as

$$V_{i,j}(t) \quad = \quad V_{i,j}(0)e^{-\upsilon_{i,j}t} + Z_{re}. \tag{2.2}$$

For simplicity, we assume $Z_{re} \sim \mathcal{N}(0, \sigma_{re})$ has a Gaussian distribution.

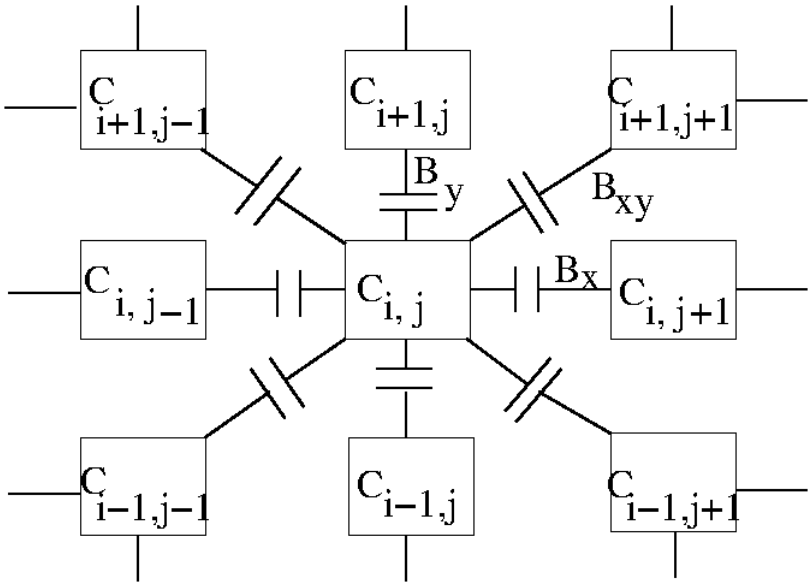


Figure 2.6: Illustration of cell-to-cell interference

2.3.4 Cell-to-cell interference

Cell-to-cell interference is a complex issue for flash memories. The analog level we can read from a cell depends not only on the cell's own level, but also the levels of its neighboring cells. This is due to the parasitic capacitance-coupling effect between neighboring cells. For this reason, we differentiate the concept of *intrinsic analog level* from the *extrinsic analog level* of a cell. By intrinsic (respectively, extrinsic) analog level, we refer to the cell's analog level when there is no (respectively, there is) interference from neighboring cells. For cell $c_{i,j}$, we use $\hat{V}_{i,j}$ to denote its intrinsic analog level, and use $V_{i,j}$ to denote its extrinsic analog level.

A model for cell-to-cell interference is proposed in [11], where a cell is interfered by its eight neighboring cells, as shown in Fig. 2.6. Here B_x, B_y and B_{xy} refer to coupling parameters between neighboring cells in the row direction, the column direction and the diagonal direction, respectively. We model the effect of cell-to-cell interference as

$$
\begin{aligned}
V_{i,j} &= \hat{V}_{i,j} + B_x(\hat{V}_{i,j-1} + \hat{V}_{i,j+1}) + B_y(\hat{V}_{i-1,j} \\
&+ \hat{V}_{i+1,j}) + B_{xy}(\hat{V}_{i-1,j+1} + \hat{V}_{i-1,j-1} \\
&+ \hat{V}_{i+1,j+1} + \hat{V}_{i+1,j-1}) + Z_{inter}, \qquad (2.3)
\end{aligned}
$$

where the noise Z_{inter} accounts for the possible deviation from the above linear model. For simplicity, we assume $Z_{inter} \sim \mathcal{N}(0, \sigma_{inter})$.

For two neighboring cells A and B, if A is programmed before B, when B is programmed, the interference from A can be compensated by programming because the readable level of cell B is already its extrinsic analog level.

2.3.5 Read disturb

When the k-th page is read (for $k \in \{0, 1, \cdots, W-1\}$), the other pages are softly programmed due to the voltage V_{pass} added on their control gates. For a disturbed cell $c_{i,j}$ (for $i \in \{0, 1, \cdots, W-1\} - \{k\}$ and $j \in \{0, 1, \cdots, B-1\}$), we denote its analog level before the read disturb by $V_{i,j}$, and denote that after the read disturb by $V'_{i,j}$. We model read disturb as

$$
V'_{i,j} = V_{i,j} + \gamma^{rd}_{i,j} + Z_{rd}, \qquad (2.4)
$$

where $\gamma^{rd}_{i,j}$ is a parameter that depends on the time interval for the read operation, the strength of the electrical field between cell $c_{i,j}$'s control gate and P-substrate, and the cell's capacitance; and the noise Z_{rd} accounts for the possible deviation from the above simple linear model. For simplicity, we assume $Z_{rd} \sim \mathcal{N}(0, \sigma_{rd})$ has a Gaussian distribution.

2.3.6 Program disturb

When the i-th page is programmed (for $i \in \{0, 1, \cdots, W-1\}$), let $\mathcal{S} \subseteq \{0, 1, \cdots, B-1\}$ denote the indices of those programmed cells in that page. The unprogrammed cells in that page, which have indices in $\{0, 1, \cdots, B-1\} - \mathcal{S}$, will be softly programmed, which is called program disturb. For a disturbed cell $c_{i,j}$ (for $j \in \{0, 1, \cdots, B-1\} - \mathcal{S}$), we denote its analog level before the program disturb by $V_{i,j}$, and denote that after the program disturb by $V'_{i,j}$. We model program disturb as

$$V'_{i,j} = V_{i,j} + \gamma_{i,j}^{prod} + Z_{prod}, \tag{2.5}$$

where $\gamma_{i,j}^{prod}$ has a similar meaning as in function (2.4) (although it may have a different value due to changed parameters such as the time interval for programming). Here the noise Z_{prod} accounts for the possible deviation from the above simple linear model. For simplicity, we assume $Z_{prod} \sim \mathcal{N}(0, \sigma_{prod})$ has a Gaussian distribution.

2.3.7 Pass disturb

When the k-th page is programmed, the other pages are softly programmed due to the voltage V_{pass} added on their control gates. The process is similar to read disturb, and we model it as

$$V'_{i,j} = V_{i,j} + \gamma_{i,j}^{pasd} + Z_{pasd}, \tag{2.6}$$

where $\gamma_{i,j}^{pasd}$ has a similar meaning as in function (2.4) (but with possible different values, as for program disturb). And as before, Z_{pasd} accounts for the additive noise term, and for simplicity we assume $Z_{pasd} \sim \mathcal{N}(0, \sigma_{pasd})$.

2.4 Capacity Analysis for Flash Memories

In this section, we analyze the storage capacity of flash memories. In particular, we focus on its special features: how the storage capacity degrades with flash operations, the trade-off between instantaneous capacity and read disturbs when sub-thresholds are used, and the importance of dynamic thresholds.

2.4.1 Basic model for write and read operations

The storage capacity of flash cells is affected by a number of factors, including the number of cell levels, the specific implementation of read and write operations, etc. In this section, we focus on fundamental features of flash channels. So for simplicity, we consider the following simplistic write/read model for an SLC block of W pages. We first program the W pages sequentially from $\mathcal{W}_0$ to $\mathcal{W}_{W-1}$, and we assume every cell has an equal likelihood of being programmed to 0 or 1. We then have $n-1$ rounds of reading; in each round, we read the pages $\mathcal{W}_0, \mathcal{W}_1, \cdots, \mathcal{W}_{W-1}$ sequentially. Although the noise in cells is correlated (e.g., via inter-cell interference), when we compute capacity, we treat them as having independent noise. (This is, of course, a restrictive model for capacity.) Furthermore, when we analyze capacity, we assume B (the number of cells in a page) approaches infinity.

The above simple model for SLC can be extended to MLC and TLC and to more complex read/write patterns. However, the basic observations derived here still hold for more general cases.

2.4.2 Capacity degradation with flash operations

In conventional storage media such as hard disk, noise typically accumulates over time. In flash memories, however, significant noise is accumulated due to flash operations. So frequent operations will lead to large noise, thus degrade the storage capacity significantly.

In this subsection, we analyze how the analog-level distribution of cells changes with more and more write/read operations (under the model introduced earlier), and compute the corresponding storage capacity. (Note that P/E cycles degrade cells' quality, which is another source of capacity degradation, but we do not consider it here.)

We assume that the W writes and $(n-1)W$ reads in our model happen at time $0, 1, \cdots, W-1, W, W+1, \cdots, (n-1)W-1$, respectively. So for the i-th page $\mathcal{W}_i$, it was programmed at time i and was read at time $i+W, i+2W, \cdots, i+(n-1)W$. For a cell $c_{i,j}$, which is intended to be programmed to $V_k \in \{V_0, V_1\}$, let $V_{i,j}(k,t)$ be the analog level of $c_{i,j}$ after the t-th operation. We first present the recursive formula for $V_{i,j}(0,t)$ below. (For simplicity, we assume $0 < i < W-1$ to avoid boundary cases)

- When $t < i$, $c_{i,j}$ has not been programmed. It is at discrete level 0, and $V_{i,j}(k,t)$ deviates from V_0 due to inaccurate programming (of erasure in this case). Thus, $V_{i,j}(0,t) = V_0 + Z_0$ based on function (2.1).

- When $t = i$, $c_{i,j}$ does not need to be programmed, but it suffers from program disturb. Thus, $V_{i,j}(0,t) = V_{i,j}(0,t-1) + Z_{prod}$ based on function (2.5).

- When $t = i+1$, $\mathcal{W}_{i+1}$ is being programmed. $c_{i,j}$ deviates even further from V_0 by cell-to-cell interference from $c_{i+1,j-1}$, $c_{i+1,j}$ and $c_{i+1,j+1}$ and pass disturb from $\mathcal{W}_{i+1}$. Based on the models of cell-to-cell interference, i.e., function (2.3) and pass disturb, i.e., function (2.6), $V_{i,j}(0,t) = V_{i,j}(0,t-1) + B_y V_{i+1,j}(k_1,t) + B_{xy}(V_{i+1,j-1}(k_2,t) + V_{i+1,j+1}(k_3,t)) + Z_{inter} + \gamma_{i,j}^{pasd} + Z_{pasd}$, where $k_1, k_2, k_3 \in \{0,1\}$.

- When $t \in [i+2, W-1]$, $\mathcal{W}_t$ is being programmed. $c_{i,j}$ is distorted by pass disturb from $\mathcal{W}_t$. Thus based on the model of pass disturb, i.e., function (2.6), $V_{i,j}(0,t) = V_{i,j}(0,t-1) + \gamma_{i,j}^{pasd} + Z_{pasd}$.

- When $t = W + mi$, where $m = 1, 2, \cdots, n-1$, $c_{i,j}$ is being read. There is no noise

for $c_{i,j}$; thus $V_{i,j}(0,t) = V_{i,j}(0,t-1)$.

- When $t \in [W, nW-1] - \{W+mi\}$, where $n \in \mathbb{N}^+$, $n \geq 2$ and $m = 1, 2, \cdots, n-1$, cells of $\mathcal{W}_{t \mod W}$ are being read. $c_{i,j}$ is distorted by read disturb from $\mathcal{W}_{t \mod W}$. Thus based on the model of read disturb, i.e., function (2.4), $V_{i,j}(0,t) = V_{i,j}(0,t-1) + \gamma_{i,j}^{rd} + Z_{rd}$.

We conclude the above as follows: $V_{i,j}(0,t) =$

$$\begin{cases} V_0 + Z_0 & t < i, \\ V_{i,j}(k,t-1) + Z_{prod} & t = i, \\ V_{i,j}(k,t-1) + B_y V_{i+1,j}(k_1,t) & \\ +B_{xy}(V_{i+1,j-1}(k_2,t)+ & \\ V_{i+1,j+1}(k_3,t)) + Z_{inter} & \\ +\gamma_{i,j}^{pasd} + Z_{pasd} & t = i+1, \\ V_{i,j}(k,t-1) + \gamma_{i,j}^{pasd} + Z_{pasd} & t \in [i+2, W-1], \\ V_{i,j}(k,t-1) & t = W+mi, \\ V_{i,j}(k,t-1) + \gamma_{i,j}^{rd} + Z_{rd} & \text{otherwise.} \end{cases} \tag{2.7}$$

The formula for $V_{i,j}(1,t)$ can be obtained similarly with the only difference that $V_{i,j}(1,i) = V_1 + Z_1$ due to the inaccurate programming. Therefore, we know that $V_{i,j}(1,i) = V_1 + Z_1$ and $V_{i,j}(0,i) = V_0 + Z_0 + Z_{prod}$. Furthermore, $V_{i,j}(k,t)$ (for $t = 0, 1, \cdots$) form a Markov chain.

In order to obtain the probability distribution for $V_{i,j}(k,t)$, we make the following assumptions for simplicity: for cell-to-cell interference, B_{xy} is negligible compared to B_y; given any i and j, $\gamma_{i,j}^{pasd}$ and $\gamma_{i,j}^{rd}$ are constant over time, so they have no effect on $V_{i,j}(k,t)$'s

probability distribution; $V_{i,j}(k,t)$ is independent of j, and we write it as $V_i(k,t)$ sometimes. Thus, $V_{i,j}(0,t) \sim \mathcal{N}(V_0(1+B_y), \sigma_i(0,t))$ or $V_{i,j}(0,t) \sim \mathcal{N}(V_0 + B_y V_1, \sigma_i(0,t))$ with equal probability, where $\sigma_i^2(0,t) =$

$$\begin{cases} \sigma_0^2 & t < i, \\ \sigma_i^2(0,t-1) + \sigma_{prod}^2 & t = i, \\ \sigma_i^2(0,t-1) + (B_y \sigma_{i+1}(k,t))^2 & \\ +\sigma_{pasd}^2 + \sigma_{inter}^2 & t = i+1, k \in \{0,1\}, \\ \sigma_i^2(0,t-1) + \sigma_{pasd}^2 & t \in [i+2, W-1], \\ \sigma_i^2(0,t-1) & t = W + mi, \\ \sigma_i^2(0,t-1) + \sigma_{rd}^2 & \text{otherwise.} \end{cases} \tag{2.8}$$

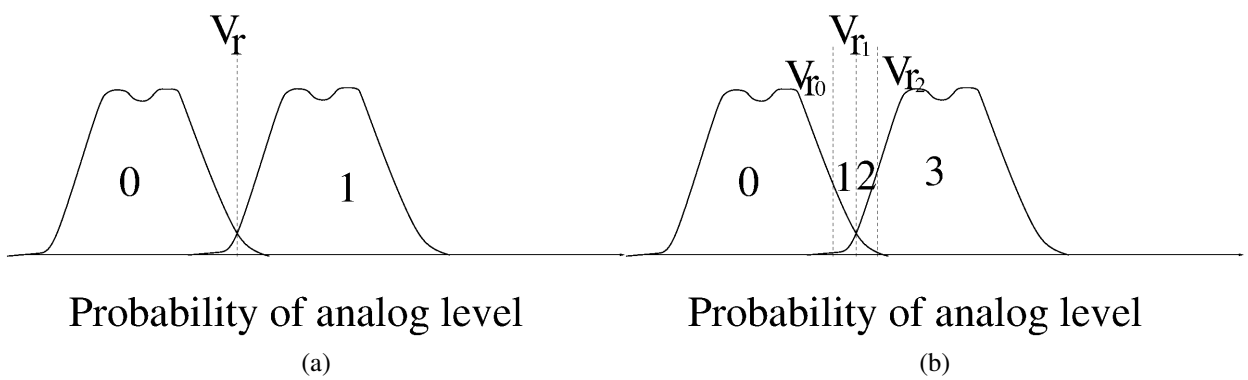


Figure 2.7: (a) SLC with one reference level; (b) SLC with three reference levels (i.e., three sub-thresholds).

Suppose the reference voltage for reading (that separates the two levels) is V_r. Also

suppose $c_{i,j}$ represents 1 if $V_{i,j}(k,t) > V_r$ and 0 otherwise (see Fig. 2.7 (a)). $P^t(Y|X)$ $(Y, X \in \{0,1\})$ is the probability that $c_{i,j}$ (which is intended to be programmed to $V_X \in \{V_0, V_1\}$) represents data Y after t operations. Thus, $P^t(1|0) =$

$$\frac{1}{2}(Q(\frac{V_r - V_0(1+B_y)}{\sigma_i(0,t)}) + Q(\frac{V_r - (V_0 + B_y V_1)}{\sigma_i(0,t)})), \tag{2.9}$$

where $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{+\infty} e^{-t^2/2} dt$.

With a similar process and assumptions as above, we obtain that $V_{i,j}(1,t) \sim \mathcal{N}(V_0, \sigma_0)$ when $t < i$. $V_{i,j}(1,t) \sim \mathcal{N}(V_1(1+B_y), \sigma_i(1,t))$ or $V_{i,j}(1,t) \sim \mathcal{N}(V_1 + B_y V_0, \sigma_i(1,t))$ with equal probability when $t \geq i$, where $\sigma_i^2(1,t) =$

$$\begin{cases} \sigma_1^2 & t = i, \\ \sigma_i^2(1,t-1) + (B_y \sigma_{i+1}(k,t))^2 & \\ +\sigma_{pasd}^2 + \sigma_{inter}^2 & t = i+1, k \in \{0,1\}, \\ \sigma_i^2(1,t-1) + \sigma_{pasd}^2 & t \in [i+2, W-1], \\ \sigma_i^2(1,t-1) & t = W + mi, \\ \sigma_i^2(1,t-1) + \sigma_{rd}^2 & \text{otherwise.} \end{cases} \tag{2.10}$$

$P^t(0|1) =$

$$\begin{cases} 1 - Q(\frac{V_r - V_0}{\sigma_1}) & t < i, \\ 1 - \frac{1}{2}(Q(\frac{V_r - V_1(1+B_y)}{\sigma_i(1,t)}) + Q(\frac{V_r - (V_1 + B_y V_0)}{\sigma_i(1,t)}) & t \geq i. \end{cases} \tag{2.11}$$

Let $\mathcal{X} = \mathcal{Y} = \{0,1\}$, and our channel model is $\mathbb{P} = (\mathcal{X}, \mathcal{Y}, P^t(Y|X))$. An example is presented in Figure 2.8 (a).

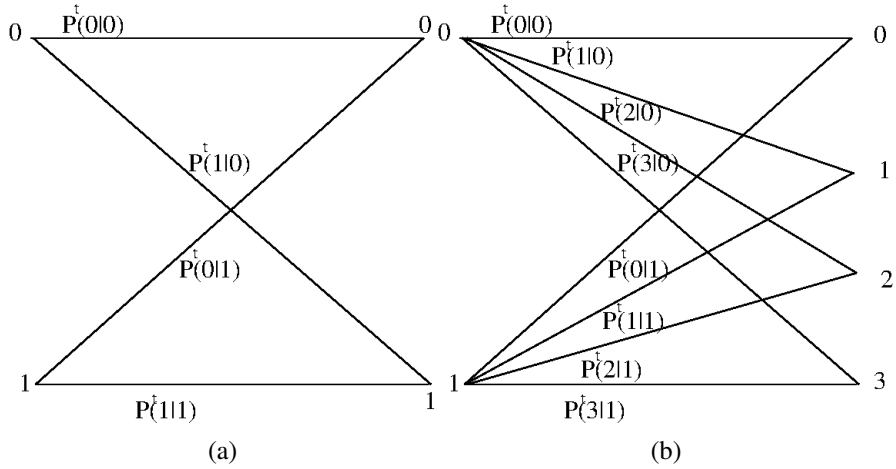


Figure 2.8: (a) Noise channel model with single reference voltage; (b) Noise channel model with three reference voltages.

Thus, the capacity of $\mathcal{W}_i$ after the t-th operation is

$$\begin{aligned} \mathcal{C}_i(t) &= I(X;Y) = H(X) - H(X|Y), \\ &= 1 - H(X|Y), \\ &= 1 - \frac{1}{2} \sum_{X,Y \in \{0,1\}} P^t(Y|X) \log_2 \frac{P^t(Y|X)}{\sum_X P^t(Y|X)}, \end{aligned} \tag{2.12}$$

where equation (2.12) is based on the assumption that X is uniformly distributed over $\{0, 1\}$. Due to data-processing inequality [16, chapter 2], we conclude that $\mathcal{C}_i(t+1) \leq \mathcal{C}_i(t)$ for $t \in \mathbb{N}$.

Let V_r be 1.4 and the remaining parameters be fixed values in Table 2.2. We numerically calculate $\mathcal{C}_2(t)$ for $t = 0, 1, \cdots, 127$ as shown by the solid line of Figure 2.9 (a) and (b). We can clearly see that the storage capacity $\mathcal{C}_2(t)$ decreases with more and more operations.

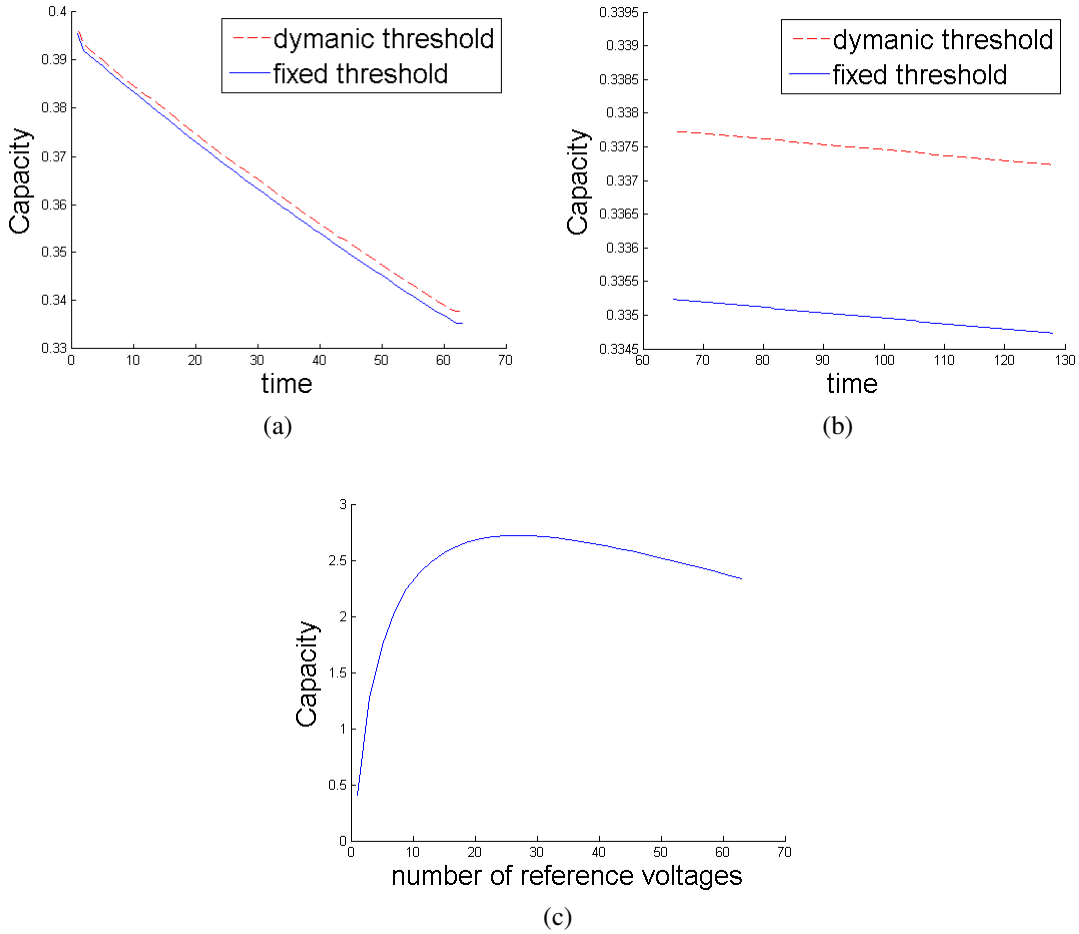


Figure 2.9: (a), (b) Comparison between the fixed-reference-voltage scheme and the dynamic-reference-voltage scheme during sequential write operations ((a)) and sequential read operations ((b)). Here the x-axis is the discrete time when write or read operations happen, the y-axis is the storage capacity, the solid curve corresponds to $\mathcal{C}_2(t)$ $t = 0, 1, \cdots, 127$ for fixed reference voltage, and the dashed curve corresponds to $\mathcal{C}_2^d(t)$ for dynamic reference voltage. (c) The trade-off between the number of sub-thresholds (reference voltages) and capacity (note that the y-axis is scaled).

Table 2.2: Parameters used in computing $\mathcal{C}(t)$

σ_0^2	σ_1^2	σ_{inter}^2	σ_{pasd}^2	σ_{rd}^2	σ_{prod}^2
2	1	9×10^{-3}	5×10^{-3}	10^{-4}	8×10^{-3}
$(B_y\sigma_{i+1}(0,t))^2$	$(B_y\sigma_{i+1}(1,t))^2$	B_y	V_0	V_1	W
10^{-3}	10^{-3}	0.01	0	2.5	64

2.4.3 The impact of sub-threshold for flash capacity

Recently, the usage of sub-thresholds (e.g., [69, 23, 24]) in flash memories has attracted great research interest. With sub-thresholds, there are multiple reference voltages between adjacent discrete levels (e.g., in Figure 2.7 (b) there are three reference voltages $V_{r_0}, V_{r_1}, V_{r_2}$ between two adjacent cell level distributions). The purpose of sub-thresholds is to obtain more soft information on cell levels, and improve coding performance (e.g., for LDPC codes). However, we observe that there is also a trade-off. Sub-thresholds lead to more reads, and therefore causes more read disturbs. Although having sub-thresholds can increase the precision of reading at the moment, the additional noise caused by read disturbs also distorts cell levels and is accumulated for future reading. Therefore, there is an optimal way to set sub-thresholds to maximize capacity over the flash memory's lifetime (which is not necessarily the more sub-thresholds the better).

In this subsection, we explore the impact of sub-thresholds for flash capacity, focusing on the trade-off between read precision and read disturbs. (How to set the positions of sub-thresholds is beyond the scope of this paper, and there is already a significant body of work on it [69, 24, 23].) Consider SLC, for $l = 1, 3, 5, \cdots$, let $\mathcal{V}(l) = \{V_{r_0}, V_{r_1}, \cdots, V_{r_{l-1}}\}$ denote the set of l sub-thresholds we use for separating discrete level 0 from level 1. Let V_r be the single sub-threshold when $l = 1$. We require the sub-thresholds in $\mathcal{V}(l)$ be symmetric with respect to V_r; we also require $\mathcal{V}(l-1) \subset \mathcal{V}(l)$ so that more soft information

can be obtained with more sub-thresholds (if no read disturb is considered). Specifically, we make all the sub-thresholds fall in the region $[V_0 + \frac{(1+B_y)(V_0+V_1)}{2}, V_1 + \frac{(1+B_y)(V_0+V_1)}{2}]$. Let L be the maximum number of sub-thresholds used. Let $\delta = \frac{V_1-V_0}{2L}$. We set $\mathcal{V}(l-1)$ as $\{V_{r_k} = V_r - (\lfloor \frac{l}{2} \rfloor - k)\delta | k = 0, 1, \cdots, l-2\}$.

An SLC with l sub-thresholds can be modeled by a 2-input $(l+1)$-output channel, where the $(l+1)$ outputs– $0, 1, \cdots, l$– corresponds to $l+1$ regions separated by the l sub-thresholds. Let $\mathcal{X} = \{0, 1\}$, $\mathcal{Y} = \{0, 1, \cdots, l\}$, and $P^t(Y|X)$ $(X \in \mathcal{X}, Y \in \mathcal{Y})$ be the probability that $c_{i,j}$ (which is intended to be programmed to $V_X \in \{V_0, V_1\}$) is read as $Y \in \mathcal{Y}$ after t operations. $P^t(Y|X)$ can be obtained in a similar way as before. (With the same setting and the similar analysis of the previous subsection, we obtain that $V_{i,j}(k,t) = V_{i,j}(k,t-1) + \gamma_{i,j}^{rd} + l \times Z_{rd}$ when it is suffered from read disturbs. The remaining cases of $V_{i,j}(k,t)$ are the same as those of the previous subsection.) Our proposed channel model is $\mathbb{P}^m = (\mathcal{X}, \mathcal{Y}, P^t(Y|X))$. Figure 9.4 (b) presents an illustration of the channel model with three sub-thresholds.

Let the capacity of the i-th page $\mathcal{W}_i$ (with l sub-thresholds) after t write/read operations be $\mathcal{C}_i(l,t) = I(X;Y)$. With parameters listed in Table 2.2 except $\sigma_0^2 = \sigma_1^2 = 1$, we present $\mathcal{C}_2(l, 500)$ for different l in Figure 2.9 (c). (The capacity for other values of i and t has similar shapes.) As shown in Figure 2.9(c), there is a trade-off between the number of sub-thresholds and storage capacity. When there are too many sub-thresholds, the impact of read disturbs becomes dominant, and the corresponding capacity decreases.

2.4.4 Dynamically adjust reference threshold voltages

It can be seen from the error models that flash disturbs are highly correlated (both in time and space), and the noise has a tendency to be non-symmetric (e.g., disturbs tend to increase cell levels). Therefore, it is important to set reference voltages adaptively over time to reduce errors and maximize capacity. Such a scheme is called *dynamic threshold*,

and has been studied before [80, 75]. In this subsection, we study how dynamic thresholds can help improve storage capacity based on our flash models.

Let $V_r(t)$ be the reference voltage we adaptively choose for the t-th operation. Let $ER_k(t)$ denote the error probability of quantizing $V_i(k,t)$ (for simplicity, we assume $t \geq i$). Therefore $ER_k(t) =$

$$\begin{cases} \frac{1}{2}(Q(\frac{V_r(t)-V_0(1+B_y)}{\sigma_i(0,t)}) + Q(\frac{V_r-(V_0+B_yV_1)}{\sigma_i(0,t)})) & k=0, \\ 1-\frac{1}{2}(Q(\frac{V_r(t)-V_1(1+B_y)}{\sigma_i(1,t)}) + Q(\frac{V_r-(V_1+B_yV_0)}{\sigma_i(1,t)})) & k=1. \end{cases} \tag{2.13}$$

Assume that k is uniformly distributed over $\{0,1\}$. Thus the total quantizing error probability for the t-th operation is $TER(t) = \frac{1}{2}+$

$$\begin{aligned} &\frac{1}{4}(Q(\frac{V_r(t)-V_0(1+B_y)}{\sigma_i(0,t)}) + Q(\frac{V_r(t)-(V_0+B_yV_1)}{\sigma_i(0,t)})) \\ &-\frac{1}{4}(Q(\frac{V_r(t)-V_1(1+B_y)}{\sigma_i(1,t)}) + Q(\frac{V_r(t)-(V_1+B_yV_0)}{\sigma_i(1,t)})). \end{aligned}$$

The objective of dynamic reference voltage is to choose $V_r(t)$ such that $TER(t)$ is minimized, therefore $V_r(t)$ should satisfy

$$\begin{aligned} &\frac{\partial(Q(\frac{V_r(t)-V_0(1+B_y)}{\sigma_i(0,t)}) + Q(\frac{V_r(t)-(V_0+B_yV_1)}{\sigma_i(0,t)}))}{\partial V_r(t)} = \\ &\frac{\partial(Q(\frac{V_r(t)-V_1(1+B_y)}{\sigma_i(1,t)}) + Q(\frac{V_r(t)-(V_1+B_yV_0)}{\sigma_i(1,t)}))}{\partial V_r(t)}. \end{aligned} \tag{2.14}$$

Similarly, we can obtain the probability distribution of $V_{i,j}(k,t)$, and $P^t(Y|X)$ for $X \in \mathcal{X} = \{0,1\}$ and $Y \in \mathcal{Y} = \{0,1\}$. The channel model of $\mathcal{W}_i$ for dynamic reference voltages is denoted by $\mathbb{P}^d = (\mathcal{X}, \mathcal{Y}, P^t(Y|X))$, and its capacity is $\mathcal{C}_i^d(t) = I(X;Y)$. With parameters of Table 2.2, we numerically compute $\mathcal{C}_2^d(t)$, which is shown by the dashed curve in Figure 2.9 (a) and (b). We can see that after dynamically adjusting the refer-

ence voltages, the channel (with quantization) becomes less noisy and the storage capacity increases correspondingly.

2.5 Concluding Remarks

In this paper, we explore various noise and disturb mechanisms in NAND flash memories, and build their corresponding information-theoretic channel models. We further study the storage capacity of flash memory under these channels. In particular, we show the impact of read/write operations on flash capacity, as well as the intriguing effect of sub-thresholds and dynamic thresholds. It is important to design coding schemes adaptively corresponding to the special properties of flash memories. That remains as our future work.

3. JOINT DECODING OF CONTENT-REPLICATION CODES FOR FLASH MEMORIES

3.1 Introduction

One challenge for flash memories is the data reliability as several types of noise [49] exist. Besides a strong error correcting code, e.g., LDPC code, another mechanism to protect flash memories is *memory scrubbing* [64], i.e., while errors accumulate in a codeword, with the next block erasure, the codeword is corrected and a new error-free codeword is written back to the memory. However, in flash memory rewrites are made in an *out-of-place* fashion, i.e., an updated codeword is stored at a new physical address and the original codeword remains in the memory. Those mechanisms lead to multiple copies of codewords, i.e., the *content-replicated codeword* problem. Besides memory scrubbing, other factors also may cause the content-replication problem such as garbage collection, weal-leveling, etc, and it is estimated that on average $3 \sim 13$ (i.e., the exact number depends on the work load traffic and various algorithms used) copies of content-replicated codewords can be generated [21].

In this work, we enhance the flash memory reliability by utilizing the existence of *two* content-replicated codewords for decoding, including an old codeword and a new codeword storing the same information. We aim at designing a *joint decoding* scheme having access to both content-replicated codewords, and explore its decoding performance. This leads to reliability improvement in flash memories. We further study a new paradigm where the two content-replicated codewords have different forms for better performance. The significance of this paper is two-fold: on the practical side, the new coding scheme utilizes the unique properties of flash memories; on the theoretical side, we show that increasing the diversity of error-correcting codes in the storage system can improve the re-

liability of replicated data even if there exist constraints in their joint decoding algorithms.

The rest of this paper is organized as follows, in Section 3.2 we present the problem statement; we then present various joint decoding algorithms and study their theoretical decoding performances in Section 3.3 for Binary Erasure Channel and Additive White Gaussian Noise channel in Section 3.4, respectively; the conclusion and future work are presented in Section 3.5.

3.2 Problem Statement

In this section, we first define some notations used throughout this paper, and then formally define the problem.

Let $\mathcal{D} = \{0, 1, \cdots, M-1\}$ be the message set for $M \in \mathbb{N}$, and let $\mathcal{X}$ and $\mathcal{Y}$ be two alphabets of the symbols stored in a cell. Let two encoders be $f_1 : \mathcal{D} \to \mathcal{X}^N$ and $f_2 : \mathcal{D} \to \mathcal{X}^N$, and the desired joint decoder be $h : \mathcal{Y}^N \times \mathcal{Y}^N \to \mathcal{D}$, where N is the length of codewords. Let $\mathbb{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{P}_{Y|X})$ and $\mathbb{Q} = (\mathcal{X}, \mathcal{Y}, \mathcal{Q}_{Y|X})$ be two independent channels.

We illustrate the model in Figure 3.1. Here, m is a common message to both encoders, the N-dimensional vectors $x_0^{N-1}(1)$, $x_0^{N-1}(2) \in \mathcal{X}^N$ are two codewords obtained through two encoders (those encoders are not necessarily identical), and $y_0^{N-1}(1)$, $y_0^{N-1}(2)$ are two noisy codewords through $\mathbb{P}$ and $\mathbb{Q}$. The task is to design a joint decoder to give a *reliable* estimation the message m, which is denoted as $\hat{m}$, giving $y_0^{N-1}(1)$ and $y_0^{N-1}(2)$.

The problem statement is presented below:

Definition 1. Given two $(N, 2^{NR})$ error-correcting codes, a message set $\mathcal{D} = \{0, 1, \cdots, 2^{NR} - 1\}$, their encoding functions $f_1 : \mathcal{D} \to \mathcal{X}^N$ and $f_2 : \mathcal{D} \to \mathcal{X}^N$, and two idependent channels $\mathbb{P}$ and $\mathbb{Q}$, the task is to design a joint decoding scheme $h : \mathcal{Y}^N \times \mathcal{Y}^N \to \mathcal{D}$ such that $Pr(h(y_0^{N-1}(1), y_0^{N-1}(2)) \neq i | x_0^{N-1}(1) = f_1(i), x_0^{N-1}(2) = f_2(i))) \to 0$ for $i \in \mathcal{D}$ as $N \to \infty$.

That is a new decoding algorithm $h(\cdot)$ is required to design which is due to the presence of content-replicated codewords. For simplicity we assume that such a joint decoding scheme always exists, i.e., $Pr(h(y_0^{N-1}(1), y_0^{N-1}(2)) \neq i | x_0^{N-1}(1) = f_1(i), x_0^{N-1}(2) = f_2(i))) \to 0$ for $i \in \mathcal{D}$ as $N \to \infty$, and leave the exact condition under which this holds as our future work.

We point out two implicit requirements for the joint decoder in the above definition: The first is the rate of the given code should be larger than the capacities of underline two channels, i.e., $R > C(\mathbb{P})$ and $R > C(\mathbb{Q})$, therefore reliable decoding is impossible for individual decoders, i.e., $\nexists\ g_1 : \mathcal{X}^N \to \mathcal{D}$ and $\nexists\ g_2 : \mathcal{X}^N \to \mathcal{D}$ such that $Pr(g_1(y_0^{N-1}(1)) \neq i, g_2(y_0^{N-1}(2)) \neq i | x_0^{N-1}(1) = f_1(i), x_0^{N-1}(2) = f_2(i))) \to 0$ for $i \in \mathcal{D}$ as $N \to \infty$. Otherwise, the joint decoder degenerates to the individual decoder in channel coding model; The second one is the same encoders to which reliable individual decoders corresponding exist when channels are not degrading too much are required here. More precisely, given the same parameters N, R, we require $f_1(\cdot)$ and $f_2(\cdot)$ meet up with the condition that when $R < C(\mathbb{P}_1)$ and $R < C(\mathbb{Q}_1)$ for some $\mathbb{P}_1$ and $\mathbb{Q}_1$, there exist $g_1 : \mathcal{X}^N \to \mathcal{D}$ and $g_2 : \mathcal{X}^N \to \mathcal{D}$ such that $Pr(g_1(y_0^{N-1}(1)) \neq i, g_2(y_0^{N-1}(2)) \neq i | x_0^{N-1}(1) = f_1(i), x_0^{N-1}(2) = f_2(i))) \to 0$ for $i \in \mathcal{D}$ as $N \to \infty$.

The above requirements are due to the motivations of joint decoders: the joint decoder is not to replace existing individual decoders (as it is possible that individual decoders suffice to reliably decode when channels do not degrade too much, and also the content-replicated codewords can not always be guaranteed to exist) but to replace individual decoders when they fail. It is also those requirements that differentiate the joint decoder from other coding models like Multiple Access Channels with correlated sources by Splepian and Wolf [66] and Fountain code [52].

In the following, assume $\mathbb{P}$ and $\mathbb{Q}$ are Binary Erasure Channels with the same parameter ϵ, and both encoders are LDPC encoders. The following notations will be used: let

the rate of two LDPC codes be $\frac{K}{N}$, let $\mathbf{G}_1$, $\mathbf{G}_2$ be the encoding matrices, and $\mathbf{H}_1$, $\mathbf{H}_2$ denotes their parity check matrices. Let $y_0^{N-1}(1), y_0^{N-1}(2) \in \{0, 1, ?\}^N$ be two codewords received.

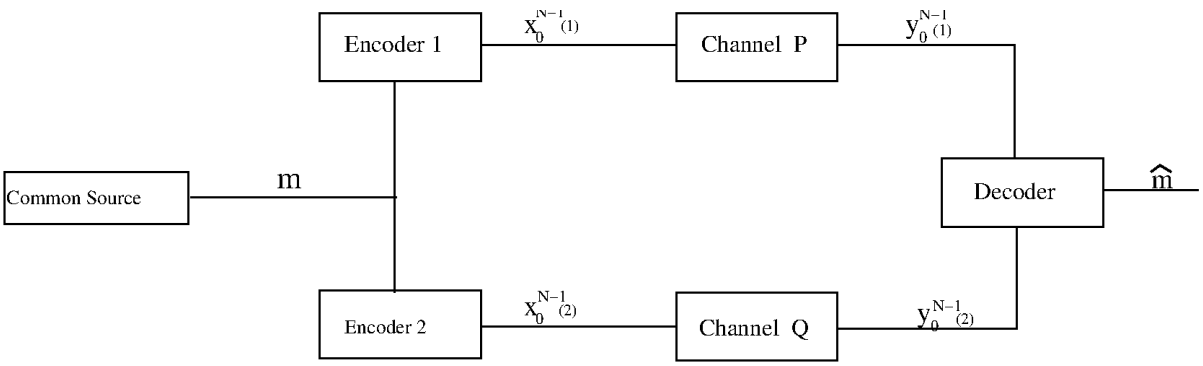


Figure 3.1: Illustration of joint decoding content-replicated codewords.

3.3 Joint Decoder for BEC Channels

In this section, we present various joint decoder designs when $\mathbb{P}$ and $\mathbb{Q}$ are Binary Erasure Channels with the same parameters.

3.3.1 Joint decoder of identical content-replicated codes

In this subsection, we start the joint decoder design with identical content-replicated codes, i.e., the two encoders are identical.

3.3.1.1 Joint decoder design and its performance

The given codes are *identical* in this case, i.e., $\mathbf{G}_1 = \mathbf{G}_2$ and $\mathbf{H}_1 = \mathbf{H}_2$.

Given $y_0^{N-1}(1)$ and $y_0^{N-1}(2)$, a *combined* codeword y_0^{N-1} is obtained as follows, for

$i = 0, 1, \cdots, N-1$, $y_i =$

$$\begin{cases} ? & \text{if } y_i(1) = y_i(2) = ?, \\ y_i(1) & \text{if } y_i(2) = ? \text{ and } y_i(1) \neq ?, \\ y_i(2) & \text{else} \end{cases}$$

The parity check matrix for y_0^{N-1} is $\mathbf{H}_1$. The decoding result is obtained by applying belief propagation to y_0^{N-1} with $\mathbf{H}_1$ and initial erasure probability ϵ^2.

Let $\lambda(x)$ and $\rho(x)$ be degree distributions for the LDPC codes used, let $\epsilon^{BP}(\lambda, \rho)$ be its original threshold as in [60], and $\epsilon^{BP}_{iden}(\lambda, \rho)$ be the threshold for our joint decoder. The comparison of $\epsilon^{BP}_{iden}(\lambda, \rho)$ and $\epsilon^{BP}(\lambda, \rho)$ for some regular LDPC codes is presented in the second and the third columns of Table 3.1, and we have $\epsilon^{BP}_{iden} > \epsilon^{BP}$.

Note that the above scheme can be generalized to cases when $\mathbb{P}$ and $\mathbb{Q}$ are with different ϵ, and due to space limitation we do not present that here.

Table 3.1: Comparison of ϵ^{BP}, ϵ^{BP}_{iden} and ϵ^{BP}_{dif}

(d_v, d_c)	ϵ^{BP}	ϵ^{BP}_{iden}	ϵ^{BP}_{dif}
(3,4)	0.6474	0.8046	0.8741
(3,5)	0.5176	0.7194	0.7594
(3,6)	0.4294	0.6553	0.6600
(4,6)	0.5061	0.7114	0.7335
(4,8)	0.3834	0.6192	0.5814

3.3.2 Joint decoder of different content-replicated codes

In the above subsection, the two codes are identical, which are effectively repetition codes, and this motivates us to explore another joint decoder design when the two encoders

are different.

3.3.2.1 Joint decoder design

The given codes are *different* in this case, i.e., $\mathbf{G}_1 \neq \mathbf{G}_2$ and $\mathbf{H}_1 \neq \mathbf{H}_2$, but codewords carry identical systematic information bits, that is, two encoding functions are $x_0^{N-1}(1) = u_0^{K-1}\mathbf{G}_1$ and $x_0^{N-1}(2) = u_0^{K-1}\mathbf{G}_2$.

Let $\mathcal{I}_1$, $\mathcal{I}_2 \subseteq \{0, 1, \cdots, N-1\}$ be the information bit index sets for $y_0^{N-1}(1)$ and $y_0^{N-1}(2)$, and let $\mathcal{P}_1$ and $\mathcal{P}_2$ be their parity check bit index sets. Let $y_0^{N-1}(1)_{\mathcal{I}_1} = (y_i(1) : i \in \mathcal{I}_1)$, i.e., information bits of $y_0^{N-1}(1)$, and similar notations apply to $y_0^{N-1}(2)_{\mathcal{I}_2}$, $y_0^{N-1}(1)_{\mathcal{P}_1}$ and $y_0^{N-1}(2)_{\mathcal{P}_2}$. Let $g(\cdot) : \mathcal{I}_1 \to \mathcal{I}_2$ be a one-to-one mapping such that $x_i(1) = x_{g(i)}(2)$ for $i \in \mathcal{I}_1$. Similar to the previous section, we define $(y_0^{N-1})_{\mathcal{I}_1}$, where $y_i =$

$$\begin{cases} ? & \text{if } y_i(1) = y_{g(i)}(2) = ?, \\ y_i(1) & \text{if } y_{g(i)}(2) = ? \text{ and } y_i(1) \neq ?, \\ y_{g(i)}(2) & \text{else} \end{cases}$$

Then, a constructed *combined* codeword is $y_0^{2N-K-1} = [(y_0^{N-1})_{\mathcal{I}_1}, y_0^{N-1}(1)_{\mathcal{P}_1}, y_0^{N-1}(2)_{\mathcal{P}_2}]$. That is, y_0^{2N-K-1} is constructed by extracting information bits from $y_0^{N-1}(1)$ and $y_0^{N-1}(2)$, and appending parity check bits from $y_0^{N-1}(1)$ and $y_0^{N-1}(2)$. An example is illustrated in Figure 3.2.

Let $\mathbf{H}_1 = [\mathbf{H}_{1,0}, \mathbf{H}_{1,1}, \cdots, \mathbf{H}_{1,N-1}]$, let $\mathbf{H}_{1,\mathcal{I}_1} = [\mathbf{H}_{1,i} : i \in \mathcal{I}_1]$, and let $\mathbf{H}_{1,\mathcal{P}_1} = [\mathbf{H}_{1,i} : i \in \mathcal{P}_1]$. Similarly, we divide $\mathbf{H}_2$ into $\mathbf{H}_{2,\mathcal{I}_2}$ and $\mathbf{H}_{2,\mathcal{P}_2}$. Then, the parity check matrix $\mathbf{H}$ for y_0^{2N-K-1} is of the form in Figure 3.3. An example is illustrated in Figure 3.2.

The decoding result is obtained by applying belief propagation to y_0^{2N-K-1} with $\mathbf{H}$, the initial erasure probability ϵ^2 for $(y_0^{N-1})_{\mathcal{I}_1}$, and ϵ for $y_0^{N-1}(1)_{\mathcal{P}_1}$ and $y_0^{N-1}(2)_{\mathcal{P}_2}$.

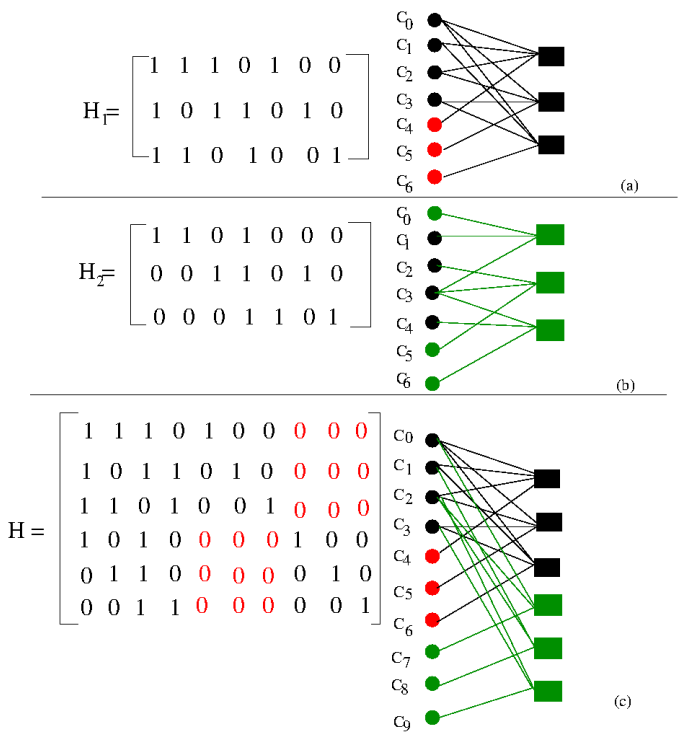


Figure 3.2: Illustration of constructed y_0^{2N-K-1} and $\mathbf{H}$. (a) The Tanner graph and $\mathbf{H}_1$ for $y_0^{N-1}(1)$, where information bits are black and parity check bits are red; (b) The Tanner graph and $\mathbf{H}_2$ for $y_0^{N-1}(2)$, where information bits are black and parity check bits are green; (c) The constructed Tanner graph and $\mathbf{H}$ based on (a) and (b), where information bits are black, parity check bits from $y_0^{N-1}(1)$ are red, and parity check bits from $y_0^{N-1}(2)$ are green.

$$H = \begin{array}{c} \\ N-K \\ N-K \end{array} \begin{array}{c} \begin{array}{ccc} K & N-K & N-K \end{array} \\ \left[\begin{array}{ccc} H_{1,I} & H_{1,P} & 0 \\ H_{2,I} & 0 & H_{2,P} \end{array}\right] \end{array}$$

Figure 3.3: Illustration of the parity check matrix **H**.

3.3.2.2 Performance analysis by density evolution

In a Tanner graph of an LDPC code, for an edge if its one end connects to an information bit of variable nodes, we call it *information edge*; If it connects to a parity check bit of variable nodes, we call it *parity edge*. For example, in Fig. 3.2 (c), the edges connecting to c_0, c_1, c_2, c_3 are information edges and the remaining edges are parity edges.

For information edges (resp. parity edges), let $\lambda_i^{(i)}$ (resp. $\lambda_i^{(p)}$) be the fraction of edges connecting to an variable node with degree i. Let $\lambda^{(i)}(x) = \sum_{i=1}^{d_v^i} \lambda_i^{(i)} x^{i-1}$, where $\sum_{i=1}^{d_v^i} \lambda_i^{(i)} = 1$, and $\lambda^{(p)}(x) = \sum_{i=1}^{d_v^p} \lambda_i^{(p)} x^{i-1}$, where $\sum_{i=1}^{d_v^p} \lambda_i^{(p)} = 1$, be the degree distribution functions from the edge perspective. For example, $\lambda^{(i)}(x) = \frac{6}{15}x^2 + \frac{4}{15}x^3 + \frac{5}{15}x^5$ and $\lambda^{(p)}(x) = 1$ in Figure 3.2 (c).

Let $\rho_{j,k}$ be the fraction of edges connecting to a check node with degree $j+k$, of which j edges are information edges and k edges are parity edges. Let $\rho(x,y) = \sum_{j,k} \rho_{j,k} x^j y^k$, where $\sum_{j,k} \rho_{j,k} = 1$, denote the edge degree distribution functions from the check node perspective. For example, $\rho(x,y) = \frac{12}{21}x^3y + \frac{9}{21}x^2y$ in Figure 3.2 (c).

Let $\rho_{j,k}^{(p)} = \frac{\rho_{j,k}}{1-\rho_{0,j+k}}$ and $\rho_{j,k}^{(i)} = \frac{\rho_{j,k}}{1-\rho_{j+k,0}}$, let $\rho^{(i)}(x,y) = \sum_{j,k} \rho_{j,k}^{(i)} x^{j-1} y^k$ where $\sum_{j,k} \rho_{j,k}^{(i)} = 1$ and $j \geq 1$, $k \geq 0$, and $\rho^{(p)}(x,y) = \sum_{j,k} \rho_{j,k}^{(p)} x^j y^{k-1}$ where $\sum_{j,k} \rho_{j,k}^{(p)} = 1$ and $j \geq 0, k \geq 1$. For example, $\rho^{(i)}(x,y) = \frac{12}{21}x^2y + \frac{9}{21}xy$ and $\rho^{(p)}(x,y) = \frac{12}{21}x^3 + \frac{9}{21}x$ in Figure 3.2 (c), where $\rho^{(p)}(x,y)$ happens to be the same as $\rho^{(i)}(x,y)$ for this example.

Lemma 2. Given two regular (d_v, d_c) LDPC codes (which are not necessarily the same), the edge degree distributions of constructed the *combined* LDPC code are: $\lambda^{(i)}(x) = x^{2d_v-1}$, $\lambda^{(p)}(x) = x^{d_v-1}$, and $\rho_{j,k} = \binom{j+k}{j}(\frac{d_c-d_v}{d_c})^j(\frac{d_v}{d_c})^k$, where $j+k=d_c$.

Proof. Based on the construction presented, for the Tanner graph of y_0^{2N-K-1}, both check nodes of $y_0^{N-1}(1)$ and $y_0^{N-1}(2)$ connect to information bits of variable nodes of y_0^{2N-K-1}, thus those node degrees are doubled; The degree of parity check bit of variables nodes of y_0^{2N-K-1} remains the same as those of $y_0^{N-1}(1)$ and $y_0^{N-1}(2)$.

The result for $\rho_{j,k}$ follows from that for a random edge it is an information edge with probability $\frac{d_c-d_v}{d_c}$, a parity edge with probability $\frac{d_v}{d_c}$, and the probability distribution that j out of $j+k$ edges are from information edges is a binomial distribution. □

From Lemma 2, we know that $\lambda^{(i)}(x)$ and $\lambda^{(p)}(x)$ are not identical, the initial effective erasure probability is ϵ^2 for information bits of y_0^{2N-K-1} and ϵ for parity bits of y_0^{2N-K-1}, thus the probabilities of a parity bit and an information bit being an erasure at the l-round of belief propagation decoding are not the same (We show this point in Figure 3.4 through a simulation with both $(3,6)$ LDPC code and initial erasure probability 0.6).

Let $x_i^{(l)}$ be the average probability of an information bit of y_0^{2N-K-1} being an erasure after the l-round of belief propagation decoding, and similarly let $x_p^{(l)}$ be that for a parity check bit of y_0^{2N-K-1}.

Our main result based on density evolution [60] is presented below:

Theorem 3. For our joint decoding of different content-replicated codes, the average erasure probabilities after l-round of belief-propagation decoding are given by

$$\begin{aligned} x_i^{(l)} &= \epsilon^2\lambda^{(i)}(1-\rho^{(i)}(1-x_p^{(l-1)}, 1-x_i^{(l-1)}))), \\ x_p^{(l)} &= \epsilon\lambda^{(p)}(1-\rho^{(p)}(1-x_i^{(l-1)}, 1-x_p^{(l-1)}))), \end{aligned}$$

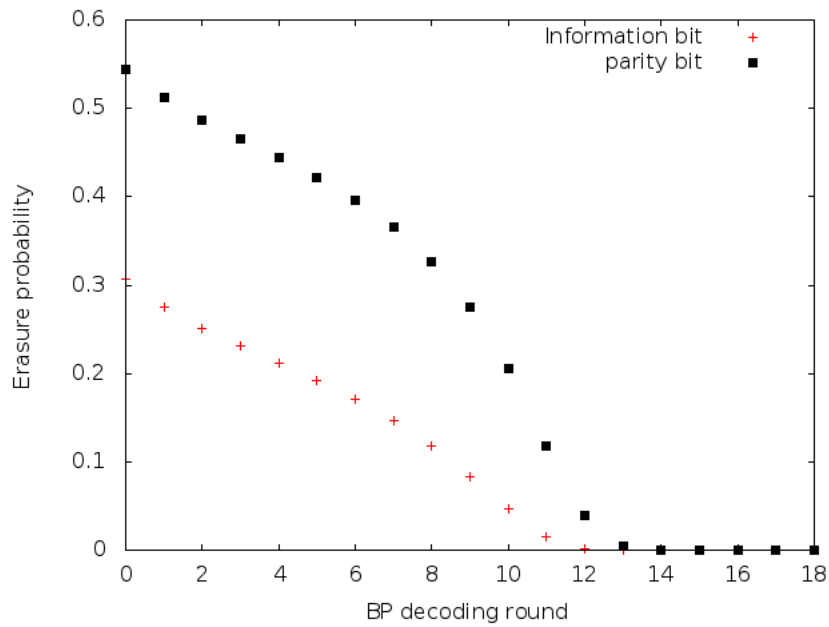


Figure 3.4: Density evolution comparision of information bits and parity bits for joint decoder of different content-replicated $(3, 6)$ LDPC codes with initial erasure probability 0.6.

Proof. We break the proof into two steps.

First, let $y_i^{(l)}$ be the average probability of being an erasure under belief-propagation decoding after l rounds for an output edge from a check node to an information bit of variable node. It is given by $y_i^{(l)} = \sum_{j,k} \rho_{j,k}^{(i)}(1 - (1 - x_i^{(l)})^{j-1}(1 - x_p^{(l)})^k) = 1 - \rho^{(i)}(1 - x_i^{(l)}, 1 - x_p^{(l)})$.

Similarly, let $y_p^{(l)}$ be the average probability of erasure under belief-propagation decoding after l-round for an output edge from a check node to a parity check bit of variable node. It is given by $y_p^{(l)} = 1 - \rho^{(p)}(1 - x_i^{(l)}, 1 - x_p^{(l)})$.

Second, the average probability of erasure for the output message of an information bit of variable nodes is given by $x_i^{(l)} = \epsilon^2 \sum_i \lambda_i^{(i)}(y_i^{(l-1)})^{i-1} = \epsilon^2 \lambda^{(i)}(y_i^{(l-1)})$.

Similarly, the average probability of erasure for the output message of an parity check bit of variable nodes is given by $x_p^{(l)} = \epsilon\lambda^{(p)}(y_p^{(l-1)})$.

Combining the above two steps, we obtain the desired results. $\square$

The following theorem presents us the existence of density evolution threshold.

Theorem 4. Based on Theorem 3, one sees density evolution updates are given by $f_i(\epsilon, x, y) = \epsilon^2\lambda^{(i)}(1-\rho^{(i)}(1-x, 1-y))$ and $f_p(\epsilon, x, y) = \epsilon\lambda^{(p)}(1-\rho^{(p)}(1-x, 1-y))$. We observe the following:

1. $f_i(\epsilon, x, y)$ and $f_p(\epsilon, x, y)$ are non-decreasing in all arguments for $\epsilon, x, y \in [0, 1]$ and strictly increasing if $\epsilon, x, y \in (0, 1)$.

2. For any $x_0, y_0, \epsilon \in [0, 1]$, the sequence $x_{l+1} = f_i(\epsilon, x_l, y_l)$ and $y_{l+1} = f_p(\epsilon, x_l, y_l)$ are monotonic in l.

3. Let $x_{l+1}(\epsilon)$ and $y_{l+1}(\epsilon)$ be defined recursively by $x_{l+1}(\epsilon) = f_i(\epsilon, x_l(\epsilon), y_l(\epsilon))$, $y_{l+1}(\epsilon) = f_p(\epsilon, x_l(\epsilon), y_l(\epsilon))$, $x_0(\epsilon) = \epsilon^2$ and $y_0(\epsilon) = \epsilon$. Then, $x_{l+1}(\epsilon)$ and $y_{l+1}(\epsilon)$ are non-decreasing in ϵ.

4. The function $x_\infty(\epsilon) = \lim_{l\to\infty} x_l(\epsilon)$ and $y_\infty(\epsilon) = \lim_{l\to\infty}(y_l(\epsilon))$ exist and are non-decreasing for all $\epsilon \in [0, 1]$.

Proof. For 1), we observe that $\frac{d}{d\epsilon}f_i(\epsilon, x, y) = 2\epsilon\lambda^{(i)}(1-\rho^{(i)}(1-x, 1-y))$ is not negative for $\epsilon, x, y \in [0, 1]$, and $\frac{d}{d\epsilon}f_p(\epsilon, x, y) = \lambda^{(p)}(1-\rho^{(p)}(1-x, 1-y))$ are positive for $x, y \in [0, 1]$. $\frac{d}{dx}f_i(\epsilon, x, y) = \epsilon^2\lambda^{(i)'}(1-\rho^{(i)}(1-x, 1-y)))\rho^{(i)'}(1-x, 1-y)$ is positive for $\epsilon, x, y \in (0, 1)$ and $\frac{d}{dx}f_p(\epsilon, x, y) = \epsilon\lambda^{(p)'}(1-\rho^{(p)}(1-x, 1-y))\rho^{(p)'}(1-x, 1-y)$ is also positive for $\epsilon, x, y \in (0, 1)$. Similarly, we can prove $\frac{d}{dy}f_i(\epsilon, x, y)$ and $\frac{d}{dy}f_p(\epsilon, x, y)$ are also positive for $\epsilon, x, y \in (0, 1)$.

For 2), the monotonicity of $f_i(\epsilon, x, y)$ and $f_p(\epsilon, x, y)$ implies that $x_{l+1} = f_i(\epsilon, x_l, y_l) \gtreqless x_l$ and $x_{l+2} = f_i(\epsilon, x_{l+1}, y_{l+1}) \gtreqless x_{l+1}$. Therefore, monotonicity holds inductively and the

direction of x_l depends only on the first step. Similarly, we can prove $y_{l+1} = f_p(\epsilon, x, y)$ are monotonic.

For 3), we first observe that $x_0(\epsilon)$ and $y_0(\epsilon)$ are non-decreasing in ϵ. Next, we proceed by induction, for any $\epsilon \leq \epsilon'$, to see that $x_{l+1}(\epsilon) = f_i(\epsilon, x_l(\epsilon), y_l(\epsilon)) \leq f_i(\epsilon', x_l(\epsilon'), y_l(\epsilon')) = x_{l+1}(\epsilon')$. Similarly, we can prove that $y_{l+1}(\epsilon)$ is non-decreasing in ϵ.

For 4), the limit exists because 2) implies the sequence $x_l(\epsilon)$ is monotonic and bounded for all $\epsilon \in [0, 1]$. The limit function is non-decreasing because 3) implies that, for any $\epsilon \leq \epsilon'$, we have $x_\infty(\epsilon) = \lim_{l\to\infty} x_l(\epsilon) \leq \lim_{l\to\infty} x_l(\epsilon') = x_\infty(\epsilon')$. The same process applies for the sequence $y_l(\epsilon)$. □

Let $\epsilon_{dif}^{BP}(\lambda^{(i)}, \rho^{(i)}) = \sup\{\epsilon \in [0, 1] : x_\infty(\epsilon) = 0\}$ (which is clearly equal to $\sup\{\epsilon \in [0, 1] : y_\infty(\epsilon) = 0\}$) be the threshold defined by the density evolution. We compute $\epsilon^{BP}, \epsilon_{iden}^{BP}, \epsilon_{dif}^{BP}$, where ϵ_{dif}^{BP} is based on the recursive functions defined in Theorem 3, for some regular LDPC codes in the fourth column of Table 3.1. Comparing with previous results, we can see that $\epsilon_{dif}^{BP} > \epsilon_{iden}^{BP}$ is possible.

3.3.3 Joint decoder of related content-replicated codes

In this section, we further explore another joint decoder when the two encoders are related, i.e., not only their parity check bits but also information bits are related.

3.3.3.1 Related encoder design

The two codes are *related* in this case. More specially, let $\mathbf{G}_3$ be an *intermediate* LDPC generator matrix with the rate $\frac{1}{2}$. Similarly, let $\mathcal{I}_i$ and $\mathcal{P}_i$ denote the information bit index set and parity check bit index set for codes with $\mathbf{G}_i$, $i = 1, 2, 3$. The encoding algorithm is below, where $(x_0^{N-1})_{\mathcal{P}_3}$ denotes the subvector $(x_i : i \in \mathcal{P}_3)$.

1. f_1: $x_0^{N-1}(1) = u_0^{K-1}\mathbf{G}_1$.

2. $v_0^{K-1} = (u_0^{K-1}\mathbf{G}_3)_{\mathcal{P}_3}$.

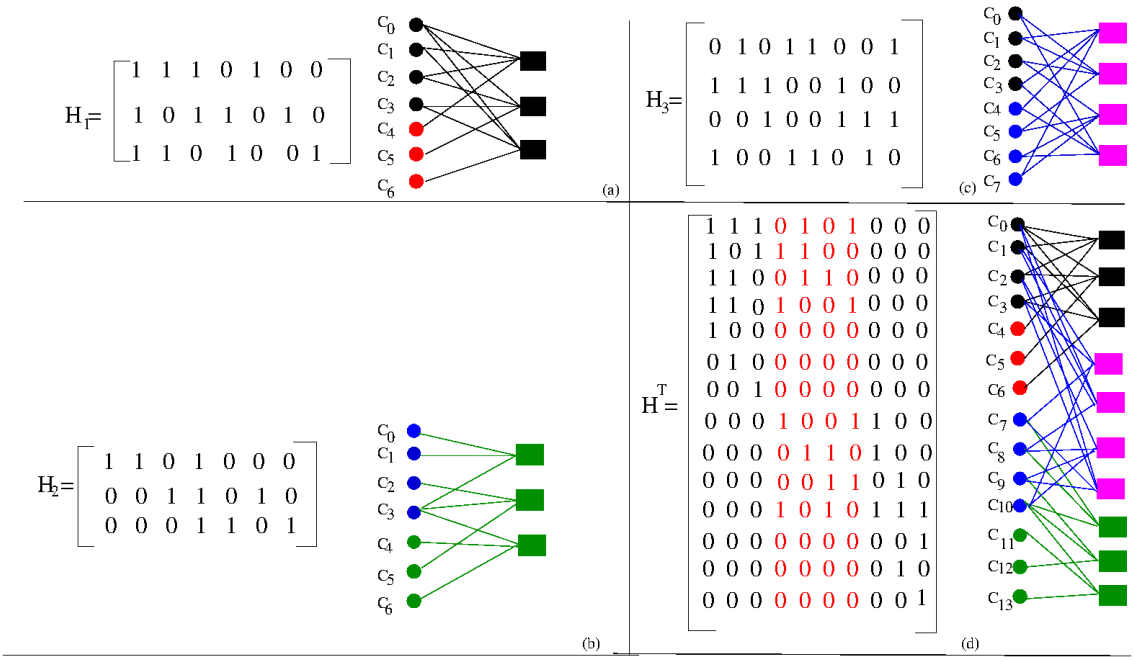


Figure 3.5: Illustration of constructed y_0^{2N-1} and $\mathbf{H}$. (a) The Tanner graph and $\mathbf{H}_1$ for $y_0^{N-1}(1)$, where information bits are black and parity check bits are red; (b) The Tanner graph and $\mathbf{H}_2$ for $y_0^{N-1}(2)$, where information bits are green and parity check bis are blue; (c) The Tanner graph and $\mathbf{H}_3$ for v_0^{K-1}, where information bits are black and parity check bits are blue; (d) The constructed Tanner graph and $\mathbf{H}$ for y_0^{2N-1}.

3. f_2: $x_0^{N-1}(2) = v_0^{K-1}\mathbf{G}_2$.

That is, $(x_0^{N-1}(1))_{\mathcal{I}_1}$ and $(x_0^{N-1}(2))_{\mathcal{I}_2}$ are related through $\mathbf{G}_3$. Refer to Figure 3.5 for an example.

3.3.3.2 Joint decoder design

A *combined* codeword is obtained by assembling $y_0^{N-1}(1)$ and $y_0^{N-1}(2)$ in the following way, $y_0^{2N-1} = (y_0^{N-1}(1)_{\mathcal{P}_1}, y_0^{N-1}(1)_{\mathcal{I}_1}, y_0^{N-1}(2)_{\mathcal{P}_2}, y_0^{N-1}(2)_{\mathcal{I}_2})$.

Let $\mathbf{H}_3$ be the parity check matrix corresponding to $\mathbf{G}_3$. Then, the parity check matrix $\mathbf{H}$ for y_0^{2N-1} is of the form in Figure 3.6. An example is presented in Figure 3.5.

The decoding result is obtained by applying belief propagation to y_0^{2N-1} with $\mathbf{H}$ and initial erasure probability ϵ.

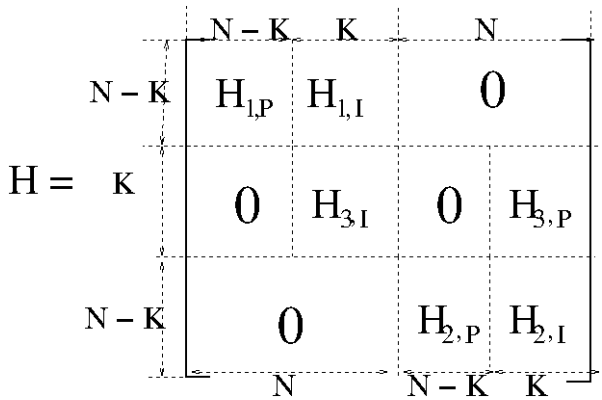


Figure 3.6: Illustration of parity check matrix $\mathbf{H}$

3.3.3.3 Performance analysis by density evolution

For individual LDPC codes x_0^{N-1}, let $\lambda^{(i)}(x) = \sum \lambda_i x^{i-1}$, $\lambda^{(p)}(x)$, $\rho^{(p)}(x,y)$ and $\rho^{(i)}(x,y)$ be the same notations used as subsection B of the previous section. For the intermediate LDPC code u_0^{K-1}, let $\lambda_3(x) = \sum \lambda_{3,i} x^{i-1}$ and $\rho_3(x)$ be the usual degree distribution functions from the edge perspective. We define a new degree distribution function for the combined LDPC code $\lambda(x,y) = \sum_{i,j} \lambda_i \lambda_{3,j} x^{i-1} y^j$. We have the following results for density evolution, where we use the same notations as the previous section.

Theorem 5. For our joint decoding of related content-replicated codes, the average erasure probabilities after l-round of belief-propagation decoding are given by

$$
\begin{aligned}
x_p^{(l)} &= \epsilon\lambda^{(p)}(1-\rho^{(p)}(1-x_i^{(l-1)}, 1-x_p^{(l-1)}))), \\
x_i^{(l)} &= \epsilon^2\lambda^{(i)}(1-\rho^{(i)}(1-x_p^{(l-1)}, 1-x_i^{(l-1)})) \\
&\quad \cdot\lambda_3(1-\rho_3(1-x_i^{(l-1)})).
\end{aligned}
$$

Proof. The proof is the similar to that of the previous theorem, and thus we present its sketch as follows.

Let $y_i^{(l)}$ (resp. $y_p^{(l)}$) denote the average probability of being an erasure after the lth round of belief propagation decoding for an output edge from a check node of $y_0^{N-1}(1)$ or $y_0^{N-1}(2)$ to an information bit (resp. parity check bit) of variable node of y_0^{2N-1}. Clearly, they follow the same formulas as Theorem 3. □

Similarly, we obtain the following convergence results:

Theorem 6. Based on Theorem 5, one sees density evolution updates are given by $f_i(\epsilon, x, y) = \epsilon^2\lambda^{(i)}(1-\rho^{(i)}(1-x, 1-y))\cdot\lambda_3(1-\rho_3(1-x))$ and $f_p(\epsilon, x, y) = \epsilon\lambda^{(p)}(1-\rho^{(p)}(1-x, 1-y))$. We observe the following:

1. $f_i(\epsilon, x, y)$ and $f_p(\epsilon, x, y)$ are non-decreasing in all arguments for $\epsilon, x, y \in [0, 1]$ and strictly increasing if $\epsilon, x, y \in (0, 1)$.

2. For any $x_0, y_0, \epsilon \in [0, 1]$, the sequence $x_{l+1} = f_i(\epsilon, x_l, y_l)$ and $y_{l+1} = f_p(\epsilon, x_l, y_l)$ are monotonic in l.

3. Let $x_{l+1}(\epsilon)$ and $y_{l+1}(\epsilon)$ be defined recursively by $x_{l+1}(\epsilon) = f_i(\epsilon, x_l(\epsilon), y_l(\epsilon))$, $y_{l+1}(\epsilon) = f_p(\epsilon, x_l(\epsilon), y_l(\epsilon))$, $x_0(\epsilon) = \epsilon^2$ and $y_0(\epsilon) = \epsilon$. Then, $x_{l+1}(\epsilon)$ and $y_{l+1}(\epsilon)$ are non-decreasing in ϵ.

4. The function $x_\infty(\epsilon) = \lim_{l\to\infty} x_l(\epsilon)$ and $y_\infty(\epsilon) = \lim_{l\to\infty}(y_l(\epsilon))$ exist and are non-decreasing for all $\epsilon \in [0,1]$.

Let $\epsilon_{re}^{BP}(\lambda^{(i)}, \rho^{(i)}) = \sup\{\epsilon \in [0,1] : x_\infty(\epsilon) = 0\}$ be the threshold defined by the density evolution. We calculate several ϵ_{re}^{BP} based on the recursive functions defined in Theorem 6 in Table 3.2, where the first row indicates the regular LDPC for $\mathbf{G}_3$, and the first column indicates the regular LDPC code for $\mathbf{G}_1$ and $\mathbf{G}_2$. For example, the result 0.8918 is the threshold when LDPC codes for $\mathbf{G}_1$ and $\mathbf{G}_2$ are $(3,4)$ regular codes, and the intermediate LDPC code is $(2,4)$ code in Table 3.2. From this table, we see that $\epsilon_{re}^{BP} > \epsilon_{iden}^{BP} > \epsilon_{dif}^{BP} > \epsilon^{BP}$ is possible with appropriate $\mathbf{G}_3$. That is the threshold can be improved by increasing the diversity of the underlying error-correcting codes.

Table 3.2: Calculation of ϵ_{re}^{BP}

(d_v, d_c)	(1,2)	(2,4)	(3,6)	(4,8)
(3,4)	0.8741	0.8918	0.8794	0.8754
(3,5)	0.7594	0.8169	0.7928	0.7771
(3,6)	0.6600	0.7569	0.7327	0.7085
(4,6)	0.7335	0.7976	0.772	0.7543
(4,8)	0.5814	0.7082	0.6917	0.662

3.4 Joint Decoders for AWGN Channels

In this section, we proceed the joint decoder designs for AWGN channel with the insight provided in previous sections. In the following, we assume that both $\mathbb{Q}$ and $\mathbb{Q}$ are AWGN channels with the same parameters, let the rates of two LDPC codes still be $\frac{K}{N}$, let $\mathbf{G}_1$, $\mathbf{G}_2$ be the encoding matrices, and let $\mathbf{H}_1$, $\mathbf{H}_2$ denote their parity check matrices. Let $x_0^{N-1}(1)$ and $x_0^{N-1}(2)$ be all zeros due to the channel symmetry, $y_0^{N-1}(1)$ and

$y_0^{N-1}(2)$ are noisy codewords through $\mathbb{P}$ and $\mathbb{Q}$, respectively, thus $y_i(1), y_i(2) \sim \mathcal{N}(0, \sigma^2)$ for $i = 0, \cdots, N-1$.

3.4.1 Joint decoder of identical content-replicated codes

We first present the joint decoder design and its theoretical performance for the case when encoders are identical, i.e., $\mathbf{G}_1 = \mathbf{G}_2$ and $\mathbf{H}_1 = \mathbf{H}_2$.

Given noisy codewords $y_0^{N-1}(1), y_0^{N-1}(2) \in \mathbb{R}^N$ of the same codeword x_0^{N-1}, the log-likely-ratio (LLR) message from channel $\mathbb{P}$, denoted as $u_{P_0}(i)$, is $\ln \frac{p(y_i(1)|x_i=1)}{p(y_i(1)|x_i=0)} = \frac{2y_i(1)}{\sigma^2}$, i.e., Gaussian with mean $\frac{2}{\sigma^2}$ and variance $\frac{4}{\sigma^2}$, and so is for the LLR message from channel $\mathbb{Q}$, denoted as $u_{Q_0}(i)$. Therefore, by averaging of LLR messages from $\mathbb{P}$ and $\mathbb{Q}$, we can obtain the *combined* LLR message as $u_0(i) = \frac{u_{P_0}(i)+u_{Q_0}(i)}{2}$, i.e., Gaussian with mean $\frac{2}{\sigma^2}$ and variance $\frac{2}{\sigma^2}$.

The decoding result is obtained by applying sum-product algorithm with $\mathbf{H}_1$ and initial LLR messages $u_0(i)$ for $i = 0, \cdots, N-1$. That is, let v be a LLR message from a variable node (with initial LLR $u_0(i)$) to a check node, then $v = u_0(i) + \sum_{i=1}^{d_v-1} u_i$, where u_i, $i = 1, \cdots, d_v - 1$, are the incoming LLRs from the neighbors of the variable node except the check node that gets the message v, and u is updated by $\tanh(\frac{u}{2}) = \prod_{i=1}^{d_c-1} \tanh(\frac{v_j}{2})$, where v_j, $j = 1, \cdots, d_c - 1$, are the incoming LLRs from $d_c - 1$ neighbors of a check node.

Let $u(l)$ be the average of LLRs from a check node to a variable node at the l-th round of sum-product decoding, let $\lambda(x)$ and $\rho(x)$ be degree distribution functions for the LDPC code used, define $\sigma_{iden}^{BP}(\lambda, \rho) = \sup\{\sigma : u(l) \to \infty \text{ as } l \to \infty\}$ be the threshold for our joint decoder. $\sigma_{iden}^{BP}(\lambda, \rho)$ can be obtained through the methods provided by Fu [28], compare it with $\sigma^{BP}(\lambda, \rho)$ in the Table 3.3, and we conclude $\sigma_{iden}^{BP}(\lambda, \rho) > \sigma^{BP}(\lambda, \rho)$.

3.4.2 Joint decoder for different content-replicated codes

In this part, we present the joint decoder design for *different* content-replicated codes. We use two (d_v, d_c) regular LDPC codes to simplify the analysis of decoding algorithm. The two content-replicated codes are different in this way, i.e, $\mathbf{G}_1 \neq \mathbf{G}_2$, $\mathbf{H}_1 \neq \mathbf{H}_2$.

3.4.2.1 Joint decoder design

Let $\mathcal{I}_1, \mathcal{I}_2, \mathcal{P}_1, \mathcal{P}_2$ and $g(\cdot)$ be the same notations as before, for the *combined* codeword obtained, $y_0^{2N-K-1} = (y_{\mathcal{I}_1}, y_0^{N-1}(1)_{\mathcal{P}_1}, y_0^{N-1}(2)_{\mathcal{P}_2})$, we have initial LLR from channel $u_i \sim \mathcal{N}(\frac{2}{\sigma^2}, \frac{2}{\sigma^2})$ for $i \in \mathcal{I}_1$ (that is by combining LLRs from $y_0^{N-1}(1)_{\mathcal{I}_1}$ and $y_0^{N-1}(2)_{\mathcal{I}_2}$), $u_j, u_k \sim \mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$ for $j \in \mathcal{P}_1$ and $k \in \mathcal{P}_2$. The decoding result is obtained by applying sum-product decoding algorithm on y_0^{2N-K-1} with $\mathbf{H}$ demonstrated in Figure 3.3 with $u_{\mathcal{I}_1}$, $u_{\mathcal{P}_1}$ and $u_{\mathcal{P}_2}$ specified as above.

3.4.2.2 Theoretical performance analysis by density evolution

In sum-product decoding algorithm, for variable nodes of y_0^{2N-K-1}, let $v_i^{(l)}$ be the average log-likely-ratio (LLR) from an information bit to parity nodes at the l-round, and similarly let $v_p^{(l)}$ be that from a parity check bit of y_0^{2N-K-1}.

For a parity node connecting to j information edges and k parity edges, let $\mu_i^{(l)}(j,k)$ and $\mu_p^{(l)}(j,k)$ be its LLR sent to an information and a parity bit at l-round, respectively. Thus, we have

$$\begin{aligned}
\mu_i^{(l)}(j,k) &= 2\tanh^{-1}\left((\tanh\frac{v_i^{(l)}}{2})^{j-1}(\tanh\frac{v_p^{(l)}}{2})^{k}\right) \\
\mu_p^{(l)}(j,k) &= 2\tanh^{-1}\left((\tanh\frac{v_i^{(l)}}{2})^{j}(\tanh\frac{v_p^{(l)}}{2})^{k-1}\right).
\end{aligned} \tag{3.1}$$

Let $u_i^{(l)}$ be the average LLR from a parity nodes to an information bit at the l-round, and similarly let $u_p^{(l)}$ be that from a parity check node to a parity bit. Then, by averaging

LLR from a check node to a information and parity bit, we have

$$\begin{aligned} u_i^{(l)} &= \sum_{j=1}^{d_c} \rho_{j,k}^{(i)} \mu_i^{(l)}(j,k), \\ u_p^{(l)} &= \sum_{j=1}^{d_c} \rho_{j,k}^{(p)} \mu_p^{(l)}(j,k), \end{aligned} \tag{3.2}$$

where $\rho_{j,k}^{(i)}$ and $\rho_{j,k}^{(p)}$ are the same as previous sections.

Thus we obtain our main result of this subsection as below:

Theorem 7. For our joint decoding of different content-replicated codes (i.e., the two LDPC codes are both (d_v, d_c) LDPC codes), the LLRs after l-round of sum-product decoding at the variable node are given by

$$\begin{aligned} v_i^{(l)} &= \mu_i^{(0)} + (2d_v - 1) \cdot u_i^{(l-1)}, \\ v_p^{(l)} &= \mu_p^{(0)} + (d_v - 1) \cdot u_p^{(l-1)}, \end{aligned}$$

where $\mu_i^{(0)}$ is the initial LLR for information bits of y_0^{2N-K-1}, and $\mu_p^{(0)}$ is that for parity bits.

Check nodes are updated as equation (3.2).

In this part, we present an approximate algorithm to obtain the density evolution based on Theorem 7.

For calculation of density evolution of LDPC codes, there is several work so far, such as [59], [63] and [28]. The method presented in [59] obtains thresholds with the Fourier transform, which is computationally intensive and thus not very practical. The method presented in [63] obtains approximate thresholds for AWGN channels with sum-product decoding based on two assumptions of the LLR passed: one is their densities are approximately Gaussian when the channel is AWGN, and the other one is the so-called *symmetry*

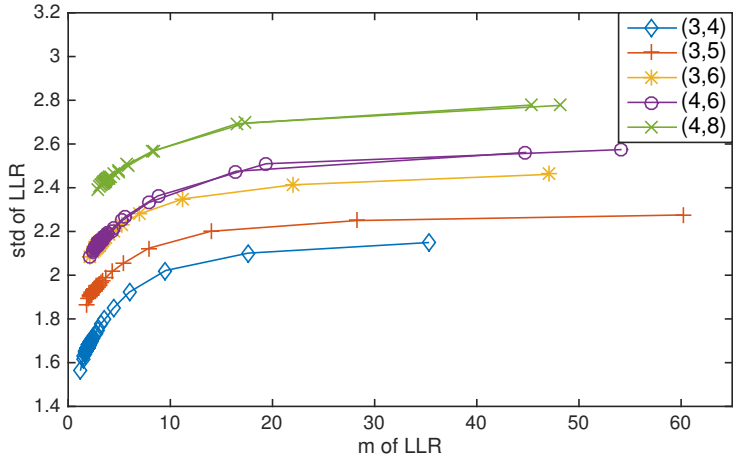


Figure 3.7: m and σ^2 of LLR for joint decoding of different content-replicated codes.

condition which requires a density function $f(x)$ to satisfy $f(x) = f(-x)e^x$ (By enforcing this condition for Gaussian with mean m and variance σ^2, this condition reduces to $\sigma^2 = 2m$). As pointed by Fu in [28] this method is not very accurate as the Gaussian assumption does not always hold especially for LLR from check nodes.

For our analysis of density evolution, we turn to the method presented in [28] to obtain the approximate threshold, and this is not only as Gaussian assumption is invalid but also as the *symmetry condition* property does not hold for our case, which is verified by intensive numerical calculations as shown in Figure 3.7 (i.e., from this figure clearly the assumption that $\sigma^2 = 2m$ does not hold). Also the update rules stated by Theorem 7 and the initial samples of $\mu_i^{(0)}$ and $\mu_p^{(0)}$ are stationary (i.e., invariant with respect to the iteration number), thus we know that those update rule preserves ergodicity. Therefore, based on the well-known property of ergodicity, i.e, any statistical parameter of the random process can be arbitrarily closely approximated by averaging over a sufficient number of samples, we have the following approximate algorithm for density evolution.

1. Step 0: choose a large number n, generate an initial n samples of $\mu_i^{(0)}$ accord-

ing to $\mathcal{N}(2/\sigma^2, 2/\sigma^2)$, and similarly generate a n samples of $\mu_p^{(0)}$ according to $\mathcal{N}(2/\sigma^2, 4/\sigma^2)$.

2. Step 1 (for variable nodes): For iteration 0, copy $\mu_i^{(0)}$ to $v_i^{(l)}$ and copy $\mu_p^{(0)}$ to $v_p^{(l)}$ as shown by variable update formula of Theorem 7. For other iterations, take the n samples of $u_p^{(l-1)}$ and $u_i^{(l-1)}$ from the previous iteration, randomly interleave $(d_v - 1)$ samples $u_p^{(l)}$ and $(2d_v - 1)$ samples $u_i^{(l)}$, respectively. Then, update $v_i^{(l)}$ and $v_p^{(l)}$ by variable update formula Theorem 7.

3. Step 2 (for check nodes): For each iteration, take the n samples of $v_i^{(l)}$ and $v_p^{(l)}$ as calculated above. Randomly interleave $(d_c - 1)$ samples of them, and then compute the n samples of $u_i^{(l)}$ and $u_p^{(l)}$ based on equation (3.1) and check update formula Theorem 7.

Let $\sigma_{diff}^{BP}(\lambda, \rho)$ be the threshold for our joint decoder with (λ, ρ) being degree distribution functions for our different LDPC codes. We calculate $\sigma_{diff}^{BP}(\lambda, \rho)$ based on the method presented above and compare it with $\sigma^{BP}(\lambda, \rho)$ and $\sigma_{iden}^{BP}(\lambda, \rho)$ in the Table 3.3. From the table we can see that unlike the BEC case, here it is possible that $\sigma_{diff}^{BP}(\lambda, \rho) > \sigma_{iden}^{BP}(\lambda, \rho)$.

We also observed that ν_i^l is slightly larger than ν_p^l for each round of sum product decoding, and we show this by the results of the $(3, 5)$ different content-replication codes in Figure 3.8.

3.4.3 Joint decoder for related content-replicated codes

In this subsection, we present the joint decoder design for *related* content-replicated codes under AWGN channel and present its theoretical analysis.

3.4.3.1 Joint decoder design

Similar as the BEC case, an intermediate generator matrix $\mathbf{G}_3$ with rate 1/2 is used to connect two LDPC generator matrices $\mathbf{G}_1$ and $\mathbf{G}_2$, and the encoding process is exactly the

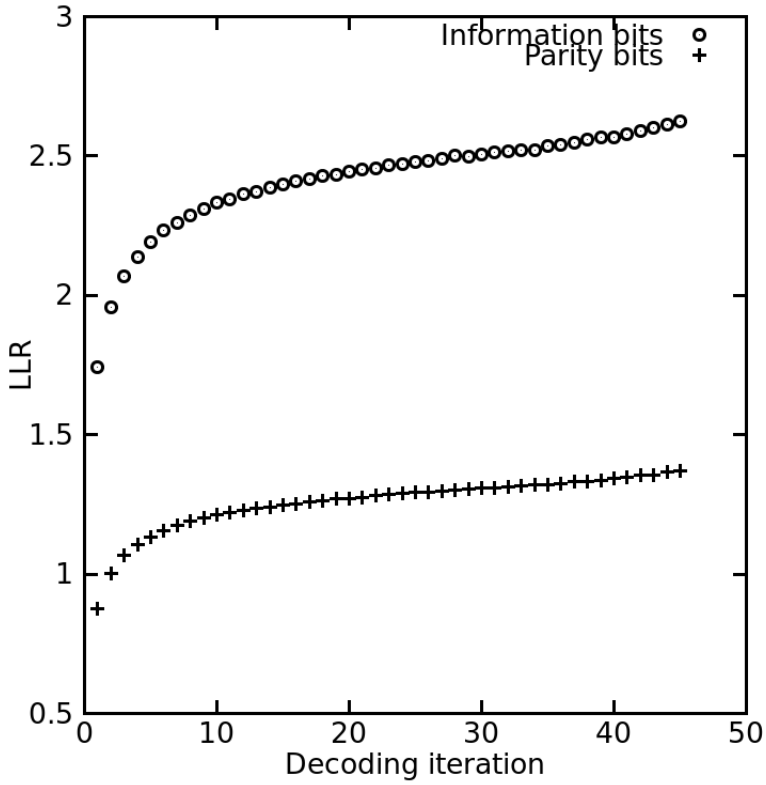


Figure 3.8: Comparison of LLR of information bits and parity bits for the joint decoder of different content-replication codes.

same as the BEC counterpart.

The decoding process is presented here: given $y_0^{N-1}(1)$ and $y_0^{N-1}(2)$, a *combined* codeword y_0^{2N-1} is constructed the same as before, i.e., $y_0^{2N-1} = (y_0^{N-1}(1)_{\mathcal{P}_1}, y_0^{N-1}(1)_{\mathcal{I}_1}, y_0^{N-1}(2)_{\mathcal{P}_2}, y_0^{N-1}(2)_{\mathcal{I}_2})$. The decoding result is obtained by applying sum-product decoding algorithm to y_0^{2N-1} with the parity check matrix $\mathbf{H}$ (constructed the same as Figure 3.6) and the initial LLR message $u_0 \sim \mathcal{N}(\frac{2}{\sigma^2}, \frac{4}{\sigma^2})$.

3.4.3.2 Theoretical performance analysis by density evolution

For density evolution, we assume that (d'_v, d'_c) regular LDPC code is used to connect two (d_v, d_c) regular LDPC codes.

For one (d_v, d_c) LDPC code, let $v_i^{(l)}$ be the average LLR from an information bit to its parity nodes (not the intermediate ones) at the l-round, and similarly let $v_p^{(l)}$ be that from a parity check bit. For a parity node connecting to j information edges and k parity edges, let $\mu_i^{(l)}(j,k)$ and $\mu_p^{(l)}(j,k)$ be its LLR sent to an information and a parity bit at l-round, respectively. Similarly, their values can be expressed the same as equation (3.1).

For the intermediate (d'_v, d'_c) LDPC code, let $x^{(l)}$ be the average LLR sent to its parity nodes and let $y^{(l)}$ be the average LLR sent to its variable nodes at the l-round of sum-product decoding. Thus we have

$$\begin{aligned} x^{(l)} &= \mu^{(0)} + d_v \cdot u_i^{(l-1)} + (d'_v - 1) \cdot y^{(l-1)}, \\ y^{(l)} &= 2\tanh^{-1}(\tanh \frac{x^{(l)}}{2})^{d'_c - 1}, \end{aligned}$$

where $\mu^{(0)}$ is the initial LLR for bits of y_0^{2N-1}.

We have the following result for the density evolution of our joint decoder:

Theorem 8. For our joint decoding of related content-replicated codes (i.e., the two LDPC codes are both (d_v, d_c) LDPC codes and the intermediate LDPC code is (d'_v, d'_c) LDPC code), the LLRs after l-round of sum-product decoding at the variable node are given by

$$\begin{aligned} v_i^{(l)} &= \mu^{(0)} + (d_v - 1) \cdot u_i^{(l-1)} + d'_v \cdot y_i^{(l-1)}, \\ v_p^{(l)} &= \mu^{(0)} + (d_v - 1) \cdot u_p^{(l-1)}, \end{aligned}$$

where $u_i^{(l)}$ and $u_p^{(l)}$ are updated as equation (3.2).

We present the approximated algorithm for density evolution based on Theorem 8 below.

1. Step 0: choose a large number n, generate an initial n samples of $\mu^{(0)}$ according to $\mathcal{N}(2/\sigma^2, 4/\sigma^2)$.

2. Step 1 (for variable nodes): For iteration 0, copy $\mu^{(0)}$ to $\nu_i^{(l)}$, $x^{(0)}$ and $\nu_p^{(0)}$ as shown by variable update formula of Theorem 8 and equation (3.2). For other iterations, take the n samples of $u_p^{(l-1)}$, $u_i^{(l-1)}$ and $y^{(l-1)}$ from the previous iteration, randomly interleave $(d_v - 1)$ samples $u_p^{(l-1)}$, $u_i^{(l-1)}$ and $y^{(l-1)}$, respectively. Then, update $v_i^{(l)}$, $v_p^{(l)}$ and $x^{(l)}$ by variable update formula of Theorem 8 and equation (3.2).

3. Step 2 (for check nodes): For each iteration, take the n samples of $v_i^{(l)}$, $x^{(l)}$ and $v_p^{(l)}$ as calculated above. Randomly interleave the samples of them, and then compute the n samples of $u_i^{(l)}$, $x^{(l)}$ and $u_p^{(l)}$ based on equation (3.2) and check update formula of Theorem 8 and equation (3.2).

Let $\sigma^{BP}_{d'_v,d'_c}(\lambda,\rho)$ be the threshold for our joint decoder with (λ,ρ) being degree distribution functions for our LDPC codes and (d'_v, d'_c) as the intermediate LDPC code. We calculate $\sigma^{BP}_{d'_v,d'_c}(\lambda,\rho)$ based on the method presented above and compare it with $\sigma^{BP}(\lambda,\rho)$, $\sigma^{BP}_{iden}(\lambda,\rho)$ and $\sigma^{BP}_{diff}(\lambda,\rho)$ in the Table 3.3. From the results, we can see that it is possible that $\sigma^{BP}_{d'_v,d'_c}(\lambda,\rho) > \sigma^{BP}_{iden}(\lambda,\rho)$ with appropriate (d'_v, d'_c).

3.5 Conclusion and Future Work

In this paper, we study two codewords carrying the same message problem, and present various joint decoding schemes and their performances. We propose the following future work:

- For our joint decoder designs, we assume that the two channels are the same for simplification, which is not true in practice. Therefore, it is interesting to explore

Table 3.3: Thresholds σ^* of AWGN channels for joint decoders

(d_v, d_c)	σ^{BP}	σ^{BP}_{iden}	σ^{BP}_{diff}	$\sigma^{BP}_{2,4}$	$\sigma^{BP}_{3,6}$	$\sigma^{BP}_{4,8}$
(3,4)	1.261	1.555	1.69	1.655	1.5	1.45
(3,5)	1.004	1.264	1.379	1.462	1.267	1.201
(3,6)	0.880	1.116	1.19	1.358	1.161	1.085
(4,6)	1.002	1.242	1.3	1.382	1.207	1.145
(4,8)	0.838	1.044	1.065	1.3	1.091	1.007

joint decoder designs with channels of the same type but with different parameters or even different channels.

- For the joint decoder design over AWGN channel case, we just focus on the regular LDPC codes for simplicity and thus to explore the joint decoder performance of content-replication codes consisting of irregular LDPC codes is another future work.
- It is interesting to explore other joint decoder design schemes with better performances.
- The current joint decoders are for LDPC codes, and we are curious whether similar results can be obtained for other codes (e.g., BCH and Polar codes).

4. COMPRESSED RANK MODULATION

4.1 Introduction

Flash memories are nonvolatile memories both electrically programmable (charge placement) and electrically erasable (charge removal). Elements of flash memories are called cells, and data is represented by the charge level of a cell: let n flash memory cells denoted by $1, 2, ..., n$. $c_i \in \mathbb{R}$ denotes the charge level of cell i, Th_1 denotes a threshold charge level, cell i represents information '0' if $c_i > Th_1$, and '1' otherwise. Such scheme is called Single-Level Cell, or *SLC*; generally, by setting $q - 1$ ($q > 2$) increasing threshold charge levels, $Th_1 < Th_2 < ... < Th_{q-1}$, cell i represents information '0' if $c_i > Th_{q-1}$, 'q-1' if $c_i \leq Th_1$, and 'j' $\in$ { '1','2', ..., 'q-2' } if $c_i \in (Th_{q-j-1}, Th_{q-j}]$. Such scheme is called Multi-Level Cell (MLC).

Flash memories have the conspicuous property that the programming and the erasing are asymmetric: while adding charge to a cell is easy, removing charge has to be done with a large number (between 2^{15} and 2^{18}) of cells.

One problem of the MLC scheme incurred by the asymmetry is the *overshooting*: since flash memory technologies usually do not support charge removals from individual cells, the charge placement has to be done in a cautious approach to avoid the charge level exceeding the threshold charge level and the global erases.

Another problem caused by the asymmetry is the limited lifetime of flash memories: it is found that after a certain number of erasing, typical quoted at 10^4 to 10^5 depending on the specific device, the performance of flash memories becomes unreliable [13] e.g., bits in a flash chip will fail. Therefore, one research area of flash memories (e.g., [35, 72, 71]) is to treat flash memories as the memories with the charge level can only be increased, which is the well-known *Write-Once Memories*, or WOM [61]. Theoretical work on WOM can

be found in [26, 25, 33].

Rank Modulation (RM) codes [36] are proposed in the context of flash memories to eliminate mainly the overshooting problem, and they can be applied to other nonvolatile memories, such as Phase Change Memories. Besides eliminating the overshooting problem, RM has other advantages such as a faster programming speed and a better reliability.

Contrary to the MLC scheme, data of RM is represented not by the charge level but by the permutation induced by n cells' charge levels. Let $\mathcal{S}_n$ denote the set of $n!$ permutations over $\{1, 2, ..., n\}$. n cells' charge levels, $c^n \stackrel{def}{=} (c_1, ..., c_n) \in \mathbb{R}^n$, induce a permutation $\mathbf{a} \in \mathcal{S}_n$ in the following way: the induced permutation is $\mathbf{a} \stackrel{def}{=} (a_1, a_2, ..., a_n) \in \mathcal{S}_n$ if and only if $c_{a_1} > c_{a_2} > ... > c_{a_n}$, i.e., cell a_1 has the highest charge level, and cell a_n has the lowest charge level. In this way, no discrete levels are needed (i.e., no need for threshold levels), thus it eliminates the overshooting problem.

Supposing a RM scheme represents any data of $\mathcal{D} = \{1, 2, ..., D\}$, the decoding function, $d : \mathcal{S}_n \to \mathcal{D}$, is to map the given permutation $\mathbf{a} \in \mathcal{S}_n$ to data $x \in \mathcal{D}$. The rewriting function, $r : \mathcal{S}_n \times \mathcal{D} \to \mathcal{S}_n$, is defined as given the current permutation $\mathbf{a} \in \mathcal{S}_n$ and data to rewrite $x \in \mathcal{D}$, we are seeking $\mathbf{b} = r(\mathbf{a}, x) \in \mathcal{S}_n$ such that $d(\mathbf{b}) = x$. The decoding performance is done by a charge-comparing operation to obtain the permutation and by the mapping function. The rewriting performance can be done by various ways, e.g., by a series of *push-to-the-top* [36] operations (raising the charge level of one cell above the current highest one).

In this work, we propose a generalization of RM, Compressed Rank Modulation (CRM) scheme, where the similar idea was independently presented by E. En Gad et al. [29]. For RM, in order to tolerate noise, there is a sufficiently large gap between every two analog charge levels, and thus *it constraints its capacity since every rank has one cell*. This motivates us to study CRM, where we let multiple cells share the same rank to achieve a higher capacity, and meanwhile it maintains the advantages of RM.

We first define the following terms. For $m, n \in \mathbb{N}$, $[n]$ is defined as the set $\{1, 2, ..., n\}$, and if $m > n$, $[n, m]$ is defined as the set $\{n, n+1, ..., m\}$. $\mathcal{S}_{mn}$ is the $(mn)!$ permutations over $[mn]$. Given a vector $m^n \stackrel{def}{=} (m_1, m_2, ..., m_n) \in \mathbb{N}^n$ and $N = \sum_{i=1}^{n} m_i$, we define $\mathcal{S}_{m^n}$ as the set $\{(\mathbf{s_1}, \mathbf{s_2}, ..., \mathbf{s_n}) | \mathbf{s_i} = \{s_{i,1}, ..., s_{i,m_i}\} \subset \{1, 2, ..., N\}, \mathbf{s_i} \bigcap \mathbf{s_j} = \emptyset$ for $i \neq j$, and $\bigcup_{i=1}^{n} \mathbf{s_i} = \{1, 2, ..., N\}\}$. When $m_1 = m_2 = ... = m_n = m$, $\mathcal{S}_{m^n}$ is denoted as $\mathcal{S}_{n,m}$. For example, $\mathcal{S}_{2,2} = \{(\{1, 2\}, \{3, 4\}), (\{1, 3\}, \{2, 4\}), (\{1, 4\}, \{2, 3\}), (\{2, 3\}, \{1, 4\}), (\{2, 4\}, \{1, 2\}), (\{3, 4\}, \{1, 2\})\}$.

We define CRM as follows: given $N = \sum_{i=1}^{n} m_i$ cells, denoted as $1, 2, ..., N$, their charge levels, $c^N \stackrel{def}{=} (c_1, c_2, ..., c_N) \in \mathbb{R}^N$, and a permutation $\mathbf{a} = (\mathbf{a_1}, ..., \mathbf{a_n}) \in \mathcal{S}_{m^n}$, c^N induces $\mathbf{a}$ if and only if $c_{a_{1,j_1}} > c_{a_{2,j_2}} > ... > c_{a_{n,j_n}}$ for $j_1 \in [m_1], j_2 \in [m_2], ..., j_n \in [m_n]$, i.e., the first m_1 highest charge level cells are in $\mathbf{a}_1$, the next m_2 highest charge level cells are in $\mathbf{a}_2$, and the last m_n highest charge level cells are in $\mathbf{a_n}$. For example, if let the charge levels of 4 cells be $(3.04, 2.56, 0.98, 2.96)$ and $m^n = (2, 2)$, then the induced permutation is $(\{1, 4\}, \{2, 3\}) \in \mathcal{S}_{2,2}$.

The decoding performance of CRM is similar to that of RM, that is though charge-comparing operations. The rewriting performance is not by the push-to-the-top operations but by the *minimal-push-up* operations, which is first presented in [46], to obtain a longer lifetime. In order to better understand it, we define *virtual level* and *rewriting cost* as follows.

As mentioned, there is no need to quantize continous cell levels for CRM, which makes it safe for overshooting, however, also makes it hard for theoretical analysis. In order to allow easy and fair analysis, we use the virtual level concept similar to [38], which is formally defined as follows.

Given a permutation $\mathbf{u} = (\mathbf{u}_1, ..., \mathbf{u}_n) \in \mathcal{S}_{m^n}$ with $N = \sum_{i=1}^{n} m_i$, the virtual level of $\mathbf{u}$, l^N, is a vector $(l_1, ..., l_N) \in \mathbb{N}^N$ that satisfies the following conditions:

- $\forall 1 \leq i \leq n$ and $j_1, j_2 \in \mathbf{u}_i$ we have $l_{j_1} = l_{j_2}$;

- $\forall 1 \leq i_1 < i_2 \leq n, j_1 \in \mathbf{u}_{i_1}$ and $j_2 \in \mathbf{u}_{i_2}$, we have $l_{j_1} > l_{j_2}$.

Besides the two conditions, the following condition is required rewriting $\mathbf{u}$ to $\mathbf{v}$:

- Given a permutation $\mathbf{v} \in \mathcal{S}_{m^n}$ representing data to rewrite, its virtual level, $l'^N = (l'_1, ..., l'_N)$, should not decrease, i.e., $l'_i \geq l_i$ for $i \in [N]$.

The basic operation of changing $\mathbf{u}$ to $\mathbf{v}$ is a *push-up*: for cell i with rank k, $i \in \mathbf{u}_k$, the push-up operation over i is to make its charge level higher than that of all other $m_k - 1$ cells in $\mathbf{u}_k$. Our objective is *to increase the highest charge level or virtual level as few as possible to obtain a longer lifetime*.

With this purpose, we define rewriting cost from $\mathbf{u}$ to $\mathbf{v}$, $\mathcal{C}(\mathbf{u} \to \mathbf{v})$, as $\min\limits_{l'^N, l^N} \{\max\limits_{i \in [N]} l'_i - \max\limits_{i \in [N]} l_i\}$, where l'^N and l^N are virtual levels of $\mathbf{v}$ and $\mathbf{u}$, respectively. For easy and fair analysis, we assign the virtual level of $\mathbf{u}$ by letting $n, n-1, ..., 1$ to $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n$. The virtual level of $\mathbf{v}$ is determined as $\arg\min\limits_{l'^N} \{\max\limits_{i \in [N]} l'_i - n\}$. Minimal-push-up operations are the push-up operations that achieve rewriting cost.

The following example makes the above notions more concrete:

Example 9. Let $m^n = (3, 2, 1)$, two permutations are $\mathbf{u} = (\{1, 2, 3\}, \{4, 5\}, \{6\})$ and $\mathbf{v} = (\{1, 3, 6\}, \{2, 5\}, \{4\})$. Let us consider the virtual levels of $\mathbf{u}$ and $\mathbf{v}$ that achieve $\mathcal{C}(\mathbf{u} \to \mathbf{v})$. We assign the virtual level of $\mathbf{u}$ as $(3, 3, 3, 2, 2, 1)$, and the virtual level of $\mathbf{v}$ can be assigned as $(4, 3, 4, 2, 3, 4), (5, 3, 5, 2, 3, 5)$ or others satisfying its definition. It is easy to verify that $(4, 3, 4, 2, 3, 4)$ is the one making $\mathcal{C}(\mathbf{u} \to \mathbf{v}) = 1$.

The remaining of this work is structed as follows: in Section 4.2, the programming method with the minimal cost and the closed-form formula for rewriting cost are presented; in Section 4.3, the incoming ball size and the outgoing ball size over $\mathcal{S}_{n,m}$ are presented; in

Section 4.4, asymptotical rate analysis of rewriting codes and an optimal code construction are presented; the conclusion is obtained in Section 4.5.

4.2 Rewriting Data with Minimal Cost

In this section, we present the way to program from $\mathbf{u}$ to $\mathbf{v}$ such that push-up operations are minimal-push-up operations, and the closed-form formula to compute rewriting cost from the given permutations directly.

4.2.1 Minimal-push-up operations

Given two permutations $\mathbf{u}, \mathbf{v} \in \mathcal{S}_{m^n}$ with $m^n = (m_1, m_2, ..., m_n)$ and $N = \sum_{i=1}^{n} m_i$, we first present the way to program from $\mathbf{u}$ to $\mathbf{v}$ such that push-up operations are minimal-push-up operations.

Let $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n)$, $i \in [n]$ and $j \in [m_i]$, $u(i,j) \stackrel{def}{=} \arg\min_k\{|\{v|v \in \mathbf{u}_i, v = 1, 2, ..., k\}| = j\}$. That is, $u(i,j)$ is the function to obtain the index of the cell, which is the jth (in lexical order) among the ith rank cells of $\mathbf{u}$. For example, let $u = (\{1,2,3\}, \{4,5\}, \{6\})$, then $u(2,2) = \arg\min_k\{|\{v|v \in \mathbf{u}_2, v = 1, 2, ..., k\}| = 2\}$, which is 5.

Reversely, let $i \in [N]$, and $u^{-1}(i)$ be the function to obtain the rank of cell i in $\mathbf{u}$ such that $i \in \mathbf{u}_{u^{-1}(i)}$.

According to the definition, the virtual level of cell $u(i,j)$, $l_{u(i,j)}$ for $i \in [n]$ and $j \in [m_i]$, of $\mathbf{u}$ is determined as follows:

for $i = 1, 2, ..., n$ do:

  for $j = 1, 2, ..., m_i$ do:

    $l_{u(i,j)} \leftarrow n + 1 - i$.

Then, we program $\mathbf{u}$ to $\mathbf{v}$ rank by rank, from rank n to rank 1, and an example is shown in Figure 4.1 to illustrate the programming process.

We first identify the virtual level of the nth rank cells in $\mathbf{v}$, $l_{v(n,i)}$ for $i \in [m_n]$:

for $i = 1, 2, ..., m_n$ do:

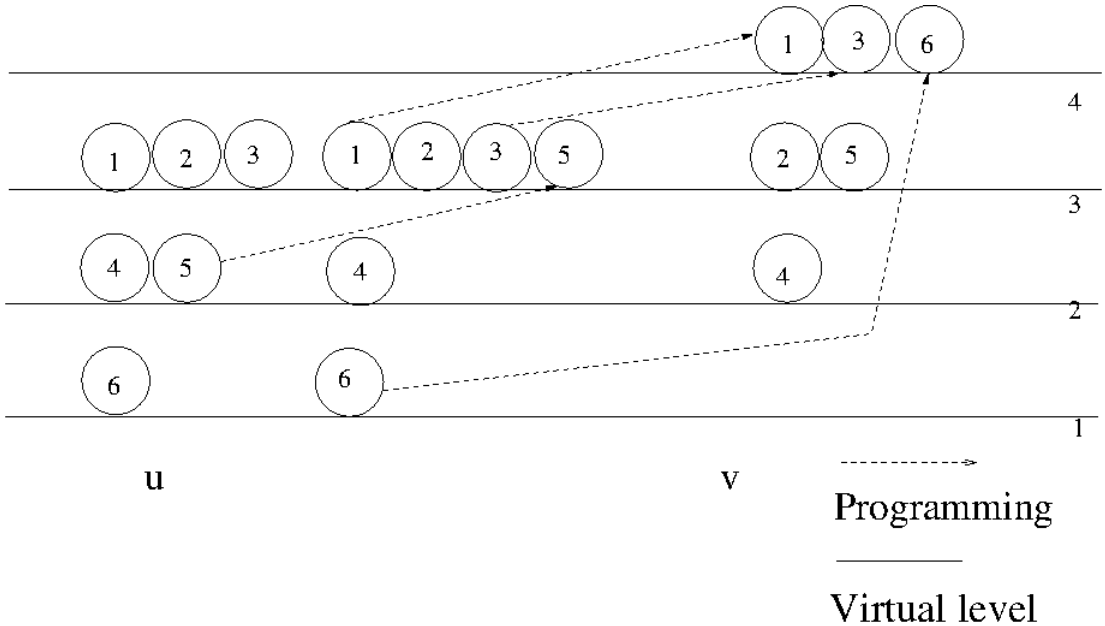


Figure 4.1: An example of programming from $\mathbf{u} = (\{1,2,3\},\{4,5\},\{6\})$ to $\mathbf{v} = (\{1,3,6\},\{2,5\},\{4\})$ via minimal-push-up operations.

$l_{v(n,i)} \leftarrow \max_{j\in[m_n]} l_{v(n,j)}$.

That is, the virtual level of the rank n cells in $\mathbf{v}$ is the highest virtual level of those cells of $\mathbf{u}$, which rank n in $\mathbf{v}$. For example, in Figure 4.1, $l_{v(3,1)} = l_4 = 2$.

We next identify the rest cell virtual levels:

for $i = n-1, n-2, ..., 1$ do:

for $j = 1, 2, ..., m_i$ do:

$l_{v(i,j)} \leftarrow \max\{l_{v(i+1,j)} + 1, l_{v(i,j)}\}$.

That is, if the virtual level of one cell is already higher than that of the next rank cells, it should stay at its original virtual level (e.g., cell 2 in Figure 4.1), otherwise the virtual level has to be higher than that of the next rank cells by 1 (e.g., cell 1, 3, 5, and 6 in Figure 4.1).

The fact that the above programming process is minimal-push-up operations can be proved briefly as follows. It is easy to obtain that to minimize the increase of $l_{v(1,j_1)}$, we

have to minimize that of $l_{v(2,j_2)}$ and $l_{v(3,j_3)}, ..., l_{v(n,j_n)}$ for $j_1 \in [m_1], j_2 \in [m_2], ..., j_n \in [m_n]$ since $l_{v(1,j_1)} > l_{v(2,j_2)} > ... > l_{v(n,j_n)}$. Thus the push-up operations minimizing the increase of $l_{v(i,j)}$ for $i \in [n]$ and $j \in [m_i]$, which is a greedy approach, are minimal-push-up operations.

4.2.2 The closed-form formula for rewriting cost

Next, we present the closed-form formula to compute rewriting cost directly from the given permutations.

According to the presented programming process, we have $\mathcal{C}(\mathbf{u} \to \mathbf{v}) = l_{v(1,i)} - n$. The next theorem presents that $\mathcal{C}(\mathbf{u} \to \mathbf{v})$ can be obtained directly from $\mathbf{u}$ and $\mathbf{v}$, instead of their virtual levels. It is actually an extention of Theorem 1 in [46], but for completeness we present its proof here:

Theorem 10. $\mathcal{C}(\mathbf{u} \to \mathbf{v}) = \max\limits_{k \in [1,n], j \in [m_k]} (k - u^{-1}(v(k,j)))$.

Proof. First, we prove by induction on i that $l_{v(i,j)}$ is

$$n + 1 - i + \max_{k \in [i,n], l \in [m_k]} (k - u^{-1}(v(k,l))).$$

The base case is $i = n$. Based on the programming process, we know $l_{v(n,j)} = \max\limits_{l \in [m_n]} l_{v(n,l)}$, which is $n + 1 - n + \max\limits_{l \in [m_n]} (n - u^{-1}(v(n,l)))$ according to the fact that $l_{u(i,j)} = n + 1 - i$. Thus, the assumption for the base case is correct.

Now, we assume it is correct for $i = h$, and we are proving the case for $i = h - 1$.

$$\begin{aligned} l_{v(h-1,j)} &= \max\{l_{v(h,j)} + 1, l_{v(h-1,j)}\} \\ &= \max\{n + 1 - (h-1) + \max_{k \in [h,n], l \in [m_k]} \end{aligned}$$

$$(k - u^{-1}(v(k,l))), n + 1 - u^{-1}(v(h-1,j))\},$$

where the first equation is based on minimal-push-up operations; the second equation is by the assumption when $i = h$ and substituting $l_{v(h-1,j)}$ by its original virtual level in $\mathbf{u}$, $n + 1 - u^{-1}(v(h-1,j))$.

Then, we proceed as:

$$\begin{aligned} &= n + 1 - (h-1) + \max\{\max_{k\in[h,n],l\in[m_k]} \\ &\quad (k - u^{-1}(v(k,l)), h - 1 - u^{-1}(v(h-1,j))\} \end{aligned}$$

$$= n + 1 - (h-1) + \max_{k\in[h-1,n],l\in[m_k]}(k - u^{-1}(v(k,l))),$$

thus the conclusion about $l_{v(i,j)}$ is correct.

Therefore,

$$\mathcal{C}(\mathbf{u} \to \mathbf{v}) = l_{v(1,i)} - n = \max_{k\in[n],l\in[m_k]}(k - u^{-1}(v(k,l))).$$

□

From the above theorem, we know that $\mathcal{C}(\mathbf{u} \to \mathbf{v})$ is equal to the maximal rank increase of cells from $\mathbf{u}$ to $\mathbf{v}$, and that $\mathcal{C}(\mathbf{u} \to \mathbf{v}) \in [n-1]$.

Given $\mathbf{u} \in \mathcal{S}_{m^n}$, $\mathbf{v} \in \mathcal{S}_{m'^n}$ with $m^n \neq m'^n$ and $N = \sum_{i=1}^{n} m_i = \sum_{i=1}^{n} m'_i$, we now generalize the above results to the rewriting case from $\mathbf{u}$ to $\mathbf{v}$, which will be used in the next section.

After applying virtual levels to $\mathbf{u}$ and $\mathbf{v}$, rewriting cost $\mathcal{C}(\mathbf{u} \to \mathbf{v})$ is still defined as $\arg\min_{l'^N}\{\max_{i\in[N]} l'_i - n\}$ with the objective to increase the highest virtual level as few as possible, where l'^N is the virtual level for $\mathbf{v}$. Extending notations of index function as well as its inverse function to $\mathbf{u}$ and $\mathbf{v}$, we obtain that minimal-push-up operations are $l_{v(i,j)} = \max\{l_{v(i+1,j_1)} + 1, l_{v(i,j)}\}$ for $i \in [n]$, $j_1 \in [m'_{i+1}]$ and $j \in [m'_i]$. Using the skill

similar to Theorem 10, we obtain that $\mathcal{C}(\mathbf{u} \to \mathbf{v}) = \max_{k\in[n],l\in[m'_k]}(k - u^{-1}(v(k,l)))$.

4.3 Analyzing Ball Sizes for Rewriting Codes

In this section, we present the exact number of permutations that $\forall \mathbf{u} \in \mathcal{S}_{n,m}$ can be programmed to (reps. from) $\mathbf{v} \in \mathcal{S}_{n,m}$ with rewriting cost constraint r.

Given $\mathbf{u} \in \mathcal{S}_{n,m}$ and $r \in [n-1]$, we define the outgoing ball (reps. the incoming ball) centered at $\mathbf{u}$ with radius r as $\mathcal{B}^{(out)}_{m,n,r}(\mathbf{u}) = \{\mathbf{v} \in \mathcal{S}_{n,n} | \mathcal{C}(\mathbf{u} \to \mathbf{v}) \leq r\}$ (reps. $\mathcal{B}^{(in)}_{n,m,r}(\mathbf{u}) = \{\mathbf{v} \in \mathcal{S}_{n,m} | \mathcal{C}(\mathbf{v} \to \mathbf{u}) \leq r\}$).

The following theorem presents us the size of an outgoing ball over $\mathcal{S}_{n,m}$ using combinatorial skills, and it is worthy to note that E. En Gad et al. [29] derived the same result independently using the induction method.

Theorem 11. $|\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})| = \binom{(r+1)m}{m}^{n-r} \frac{(rm)!}{m!^r}$.

Proof. We list $\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$ by the following steps, and we use the example of Figure 4.2 to illustrate it.

(a) Given $\forall (l_1, l_2, ..., l_{r+1}) \in \mathbb{N}^{r+1}$ such that $\sum_{i=1}^{r+1} l_i = m$, program l_i cells of $\mathbf{u}_1$ to rank i ($i \in [2, r+1]$) obtaining a cell state $\mathbf{a} \in \mathcal{S}_{m^n}$, where $m^n = (l_1, m + l_2, ..., m + l_{r+1}, m, ..., m)$. $\mathbf{A} \stackrel{def}{=} \{\mathbf{a}\} \subset \mathcal{S}_{m^n}$.

(b) $\forall \mathbf{a} \in \mathbf{A}$ and $\mathbf{u}_2, ..., \mathbf{u}_n$, define $\mathbf{u}' = (\mathbf{u}'_1, ..., \mathbf{u}'_{n-1}) \in \mathcal{S}_{n-1,m}$ with $\mathbf{u}'_i = \mathbf{u}_{i+1}$ ($i \in [n-1]$) as the set of permutations over $\bigcup_{i=2}^{n} \mathbf{u}_i$. Recursively, program $\mathbf{u}'$ to $\mathbf{v}' = (\mathbf{v}'_1, \mathbf{v}'_2, ..., \mathbf{v}'_{n-1}) \in \mathcal{S}_{n-1,m}$ such that $\mathcal{C}(\mathbf{u}' \to \mathbf{v}') \leq r$.

With $\mathbf{v}' \in \mathcal{S}_{n-1,m}$, the obtained cell state, $\mathbf{b} \in \mathcal{S}_{m^n}$, consists of $\mathbf{v}'$, with $b^{-1}(v'(i,j)) = i + 1$ for $i \in [n-1]$ and $j \in [m]$, and $\mathbf{u}_1$ with $b^{-1}(i) = a^{-1}(i)$ for $i \in \mathbf{u}_1$, where $m^n = (l_1, m + l_2, ..., m + l_{r+1}, m, ..., m)$. $\mathbf{B} \stackrel{def}{=} \{\mathbf{b}\} \subset \mathcal{S}_{m^n}$.

(c) $\forall \mathbf{b} \in \mathbf{B}$ with its corresponding $\mathbf{v}'$ specified in (b), program l_i cells of $\mathbf{v}'_{i-1}$ for $i \in [2, r+1]$ to rank 1 obtaining a cell state $\mathbf{c} \in \mathcal{S}_{n,m}$. $\mathbf{C} \stackrel{def}{=} \{\mathbf{c}\} \subset \mathcal{S}_{n,m}$.

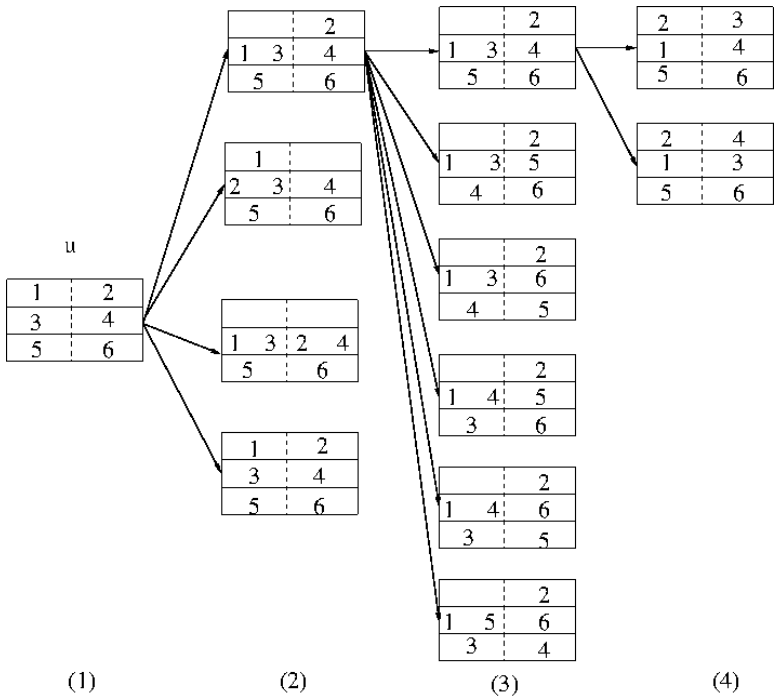


Figure 4.2: An example for the outline of Theorem 11: Given $\mathbf{u} = (\{1,2\},\{3,4\},\{5,6\}) \in \mathcal{S}_{3,2}$ where cells in the same row are in the same rank, we list all $\mathbf{v} \in \mathcal{S}_{3,2}$ satisfying $\mathcal{C}(\mathbf{u} \to \mathbf{v}) \leq 1$. (a) ((1) $\to$ (2)): program some cells of rank 1 to rank 2, and there are four possible ways; (b) ((2) $\to$ (3)): cell 3, 4, 5, and 6 in $\mathbf{u}$ can be regarded as $\mathbf{u}' = (\{3,4\},\{5,6\}) \in \mathcal{S}_{2,2}$. For each cell state in (2), recursively, we program $\mathbf{u}'$ to $\mathbf{v}' \in \mathcal{S}_{2,2}$ such that $\mathcal{C}(\mathbf{u}' \to \mathbf{v}') \leq 1$ and keep the remaining cell ranks still; (c) ((3) $\to$ (4)): after (b) the cell states are not valid permutations of $\mathcal{S}_{3,2}$, e.g., $(\{2\},\{1,3,4\},\{5,6\})$, thus cells of rank 2 are programmed to the rank 1 to make it a valid permutations of $\mathcal{S}_{3,2}$, e.g., cell 3 and cell 4 in $(\{2\},\{1,3,4\},\{5,6\})$ are programmed to rank 1 forming $(\{2,3\},\{1,4\},\{5,6\})$ and $(\{2,4\},\{1,3\},\{5,6\})$, respectively.

We are proving $\mathbf{C} = \mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$, compute $|\mathbf{C}|$, thus $|\mathbf{C}| = |\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|$.

Fist, based on the outline, $\forall \mathbf{c} \in \mathbf{C}$, $\mathcal{C}(\mathbf{u} \to \mathbf{c}) \leq r$ since any cell rank of $\mathbf{c}$ is increased by r at most. Therefore $\mathbf{c} \in \mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$, $\mathbf{C} \subseteq \mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$ and $|\mathbf{C}| \leq |\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|$.

Second, we will prove $\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u}) \subseteq \mathbf{C}$.

Define the following two terms:

- The set of hybrid states

The set of hybrid states of $\mathbf{u}, \mathbf{v} \in \mathcal{S}_{n,m}$ is denoted as $\mathcal{S}_h(\mathbf{u}, \mathbf{v}) \subseteq \mathcal{S}_{m^n}$, where $m^n = (m - \sum_{i=2}^{n} n_i, m + n_2, ..., m + n_n)$ and $n_i = |\{v(i,g) | v(i,g) = u(1,j) \text{ for } g, j \in [m]\}|$ for $i \in [2, n]$.

Given $\mathbf{u}$ and $\mathbf{v}$, $\forall \mathbf{m} \in \mathcal{S}_h(\mathbf{u}, \mathbf{v})$ can be specified via its ranks, $m^{-1}(i)$ for $i \in [mn]$, as follows:

for $i = 1, 2, ..., mn$ do:

if $i = u(1,j) = v(h,g)$ for $j, g \in [m]$, and $h \in [n]$, $m^{-1}(i) \leftarrow h$.

else if $i = u(h,g)$ for $h \in [n]$ and $g \in [m]$, $m^{-1}(i) \leftarrow h$.

That is, $\mathbf{m}$ is obtained by programming cells of $\mathbf{u}_1$ to their ranks in $\mathbf{v}$, and keeping the remaining cell ranks still. For example, the state of (2) in Figure 4.3 is the hybrid state of $\mathbf{u}$ and $\mathbf{v}$.

Clearly, $\mathbf{m}$ is *uniquely* determined by $\mathbf{v}$ and $\mathbf{u}$, thus $|\mathcal{S}_h(\mathbf{u}, \mathbf{v})| = 1$.

Furthermore, given $\forall \mathbf{v} \in \mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$, $\mathcal{S}_h(\mathbf{u}, \mathbf{v}) \subseteq \mathbf{A}$ since the cell ranks of $\mathbf{u_1}$ are increased by r at most.

- The set of near destination states

The set of near destination states from $\mathbf{u}' \in \mathcal{S}_{m^n}$ to $\mathbf{v} \in \mathcal{S}_{n,m}$ with $r \in [n-1]$ is denoted as $\mathcal{S}_n(\mathbf{u}', \mathbf{v}, r)$, where $m^n = (m - \sum_{i=2}^{n} n_i, m + n_2, ..., m + n_n)$ and $n_i = |\{v(i,g) | v(i,g) = u(1,j) \text{ for } g, j \in [m]\}|$ for $i \in [2, n]$.

Given $\mathbf{u}', \mathbf{v}$ and r, $\forall \mathbf{m} \in \mathcal{S}_n(\mathbf{u}', \mathbf{v}, r)$ can be specified via its ranks, $m^{-1}(i)$ for $i \in [mn]$, as follows:

for $i = 1, 2, ..., mn$ do:

if $i = u'(1, h) = v(1, g)$ for $h \in [m - \sum_{i=2}^{n} n_i]$ and $g \in [m]$, $m^{-1}(i) \leftarrow 1$.

if $i = v(h, g)$ for $h \in [2, n]$ and $g \in [m]$, $m^{-1}(i) \leftarrow h$.

else if $i = u'(h, g)$ for $h \in [n]$ and $g \in [m + n_h]$, $m^{-1}(i) - h \leq r$.

That is, $\mathbf{m}$ is obtained by programming cells, which are in $\mathbf{u}'_1$ as well as in $\mathbf{v}_1$ (e.g. cell 12 in Figure 4.3 if $\mathbf{u}'$ is the permutation of (2)), and cells of $\mathbf{v}_i$ for $i \in [2, n]$ (e.g. cell 10, 1, 2, 5, 11, 6, 3, 4, 7 in Figure 4.3), to their ranks in $\mathbf{v}$, and increasing the remaining cells (which are not in $\mathbf{u}'_1$ but in $\mathbf{v}_1$, e.g. cell 8, 9 in Figure 4.3) by r at most. For example, both states of (3) in Figure 4.3 are near destination states from $\mathbf{u}'$ to $\mathbf{v}$ with parameter 2.

Let $k_i = |\{u'(i, j) | v(1, l) = u'(i, j) \text{ for } j \in [m + n_i], l \in [m]\}|$ for $i \in [r + 1]$, that is among the m cells in $\mathbf{v_1}$, the numbers of cells from $\mathbf{u}'_i$ for $i \in [r + 1]$ are $k_1, k_2, ..., k_{r+1}$, respectively. Thus $|\mathcal{S}_n(\mathbf{u}', \mathbf{v}, r)| = \binom{m-k_1}{k_2, k_3, ..., k_{r+1}}$. For example, in Fig. 4.3, let $\mathbf{u}'$ be the state of (2), $|\mathcal{S}_n(\mathbf{u}', \mathbf{v}, 2)| = 2$.

For $\forall \mathbf{w} \in \mathcal{S}_n(\mathcal{S}_h(\mathbf{u}, \mathbf{v}), \mathbf{v}, r)$, it is easy to obtain that for $i \in \mathbf{u_1}$, $w^{-1}(i) = v^{-1}(i)$, that is the cell ranks of $\mathbf{u_1}$ are the same in $\mathbf{w}$ and $\mathbf{v}$. Therefore, the permutations obtained by applying (b) on $\mathcal{S}_h(\mathbf{u}, \mathbf{v})$ must contain $\mathbf{w}$. Thus $\mathcal{S}_n(\mathcal{S}_h(\mathbf{u}, \mathbf{v}), \mathbf{v}, r) \subseteq \mathbf{B}$.

Given $\forall \mathbf{v} \in \mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$, $\mathbf{v}$ can be obtained by first programming $\mathbf{u}$ to $\mathcal{S}_h(\mathbf{u}, \mathbf{v})$, $\mathcal{S}_h(\mathbf{u}, \mathbf{v})$ to $\forall \mathbf{w} \in \mathcal{S}_n(\mathcal{S}_h(\mathbf{u}, \mathbf{v}), \mathbf{v}, r)$, and $\mathbf{w}$ to $\mathbf{v}$. The example of Fig. 4.3 makes this process more concrete. Since $\mathcal{S}_h(\mathbf{u}, \mathbf{v}) \subseteq \mathbf{A}$, and $\mathcal{S}_n(\mathcal{S}_h(\mathbf{u}, \mathbf{v}), \mathbf{v}, r) \subseteq \mathbf{B}$, obtain that $\mathbf{v} \in \mathbf{C}$, $\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u}) \subseteq \mathbf{C}$, and $|\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})| \leq |\mathbf{C}|$.

Finally, we compute $|\mathbf{C}|$.

First, consider permutations without duplications.

For (a), let the numbers of cells in $\mathbf{u_1}$ programmed to the 1st, the 2nd,..., the $(r + 1)^{th}$

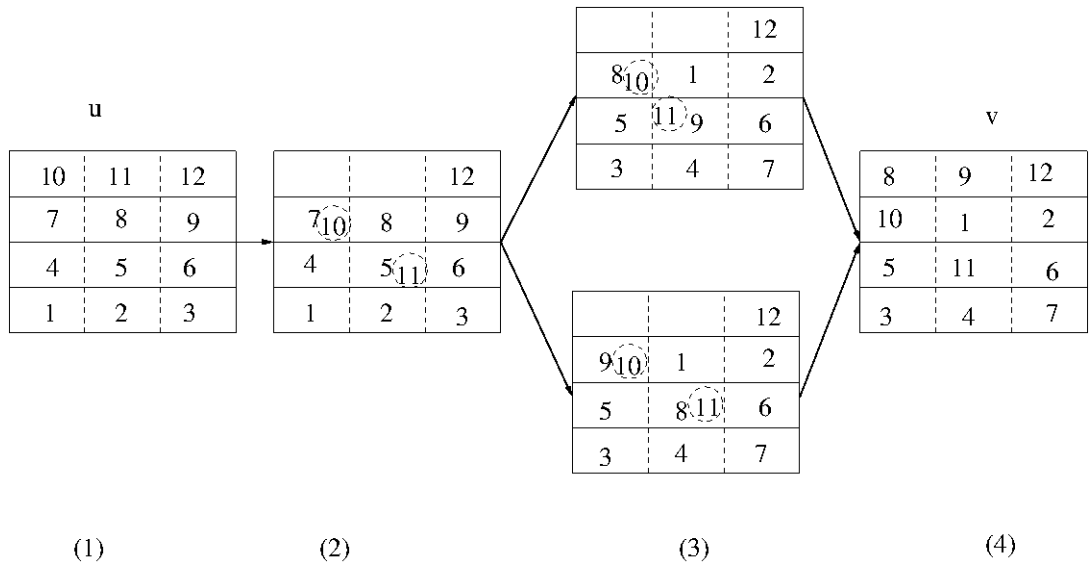


Figure 4.3: An example of changing $\mathbf{u}$ to $\mathbf{v}$ in three steps, where $r = \mathcal{C}(\mathbf{u} \to \mathbf{v}) = 2$. The state of (2) is the hybrid state of $\mathbf{u}$ and $\mathbf{v}$, and two states of (3) are near destination states from the state of (2) to $\mathbf{v}$ with parameter 2.

ranks be $l_1, l_2, ..., l_{r+1}$, respectively, thus $|\mathbf{A}| = \sum_{l_1+...+l_{r+1}=m} \binom{m}{l_1, l_2, ..., l_{r+1}}$.

For (b), $|\mathbf{B}| = \sum_{l_1+...+l_{r+1}=m} \binom{m}{l_1, l_2, ..., l_{r+1}} |\mathcal{B}^{(out)}_{n-1,m,r}|$.

For (c), rank i has $\binom{m}{l_i}$ ways to select and program cells to the 1st rank, thus the result is

$$\sum_{l_1+...+l_{r+1}=m} \binom{m}{l_1, ..., l_{r+1}} \binom{m}{l_2} ... \binom{m}{l_{r+1}} |\mathcal{B}^{(out)}_{n-1,m,r}|.$$

Next, we consider duplicated permutations in our scheme: duplications come from multiple near destination states, the number of which equals to $\binom{m-l_1}{l_2, l_3, ..., l_{r+1}}$ for each combination of $l_1, l_2, ..., l_n$.

Thus, $|\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|$ can be written as

$$\sum_{l_1+...+l_{r+1}=m} \frac{\binom{m}{l_1, ..., l_{r+1}} \binom{m}{l_2} ... \binom{m}{l_{r+1}}}{\binom{m-l_1}{l_2, ..., l_{r+1}}} |\mathcal{B}^{(out)}_{n-1,m,r}(\mathbf{u}_2..\mathbf{u}_n)|$$

$$
\begin{aligned}
&= \sum_{l_1+\ldots+l_{r+1}=m} \binom{m}{l_1}\ldots\binom{m}{l_{r+1}} |\mathcal{B}^{(out)}_{n-1,m,r}| \\
&= \binom{(r+1)m}{m} |\mathcal{B}^{(out)}_{n-1,m,r}| \\
&= \binom{(r+1)m}{m}^{n-r} \frac{(rm)!}{m!^r},
\end{aligned}
$$

where the first equation holds by simple recursive calculations, the second equation holds by the fact that $\sum_{l_1+\ldots+l_{r+1}=m} \binom{m}{l_1}\ldots\binom{m}{l_{r+1}}$ equals to the coefficient of x^m in $((1+x)^m)^{r+1}$, which is $\binom{(r+1)m}{m}$, and the last one is because $|\mathcal{B}^{(out)}_{r,m,r}(\ldots)| = |\mathcal{B}^{(out)}_{r,m,r-1}(\ldots)| = \frac{(rm)!}{m!^r}$. $\square$

Similarly, we have:

Lemma 12. $|\mathcal{B}^{(in)}_{n,m,r}(\mathbf{u})| = \binom{(r+1)m}{m}^{n-r} \frac{(rm)!}{m!^r}$.

4.4 Rewriting Codes with Bounded Cost

In this section, we study codes where the cost of the rewriting operation is limited by r. We present three rewriting codes, we study one rewriting code in detail including its asymptotical rate analysis and one code construction, and the skills can be extent to two other variants easily. For simplicity, the analysis is mainly focus on $\mathcal{S}_{n,m}$.

4.4.1 (n, m, M, r) rewriting codes

Denote a code in $\mathcal{S}_{n,m}$ with rewriting cost r and cardinality M as an (n, m, M, r) code, and we formally define it as follows:

Definition 13. An (n, m, M, r) rewriting code for CRM, where $r \in [n-1]$, is a collection of subsets $\mathcal{B} = \{\mathcal{B}_i | i \in [M]\}$ where $\mathcal{B}_i \subseteq \mathcal{S}_{n,m}$, and it represents data i, such that

- $\mathcal{B}_i \bigcap \mathcal{B}_j = \emptyset$ for $i \neq j$, and $\bigcup_{i=1}^{M} \mathcal{B}_i = \mathcal{S}_{n,m}$.
- $\forall \mathbf{u} \in \bigcup_{i=1}^{M} \mathcal{B}_i$, and $\forall j \in [M], \exists \mathbf{v} \in \mathcal{B}_j$ such that $\mathcal{C}(\mathbf{u} \to \mathbf{v}) \leq r$.

- $\forall \mathbf{u} \in \bigcup_{i=1}^{M} \mathcal{B}_i$, and $\forall j \in [M], \exists \mathbf{v} \in \mathcal{B}_j$ such that $\mathcal{C}(\mathbf{v} \to \mathbf{u}) \leq r$.

That is, we partition $\mathcal{S}_{n,m}$ into M disjoint sets, each set represents data, and every permutation can be programmed *to and from* some permutation of any set $\mathcal{B}_i$ for $i \in [M]$ with the cost constraint r. Construction 1 of this subsection presents us an example of $(3, 2, 30, 1)$ code.

The reason that we use set $\mathcal{B}_i$ to represent data i is to increase the possibility that such permutation with the cost constraint r can be found during rewriting.

According to the definition, we know that for $\forall \mathbf{u} \in \mathcal{S}_{n,m}$ every $\mathcal{B}_i$ contains at least one element of $\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$ as well as that of $\mathcal{B}^{(in)}_{n,m,r}(\mathbf{u})$, thus we have the following corollary immediately:

Corollary 14. For (n, m, M, r) code, $M \leq |\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|$ and $M \leq |\mathcal{B}^{(in)}_{n,m,r}(\mathbf{u})|$.

4.4.1.1 Asymptotical rate analysis

The rate of the (n, m, M, r) code is defined as $R = \frac{\log_2 M}{nm}$, and its asymptotical rate, denoted as *storage capacity*, $\mathcal{R}(n, r)$, is defined as $\mathcal{R}(n, r) = \lim_{m \to \infty} \frac{\log_2 M}{nm}$.

Let two random variables $X, Y \in [n]$, P_{XY}, P_X and $P_{Y|X}$ denote the joint, marginal and conditional distribution, respectively. If X is uniformly distributed in the set $[n]$, we denote it as $X \sim U(1, n)$. We define a joint probability distribution set with parameters r and n, $\mathcal{P}(n, r) = \{P_{XY} | P_X = P_Y, X \sim U(1, n), P(Y|X) = 0 \text{ if } |Y - X| \geq r + 1\}$. For example, the following probability transition matrix

$$P = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 2/9 & 1/9 & 0 \\ 1/9 & 1/9 & 1/9 \\ 0 & 1/9 & 2/9 \end{pmatrix} \end{array}$$

gives us an element of $\mathcal{P}_{1,3}$, where every entry p_{ij} stands for $P(X = i, Y = j)$.

The next lemma derives the characterization for $\mathcal{R}(n, r)$, and its proof uses the skill similar to [2, 3].

Lemma 15. $\mathcal{R}(n, r) = \max_{P_{XY} \in \mathcal{P}(n,r)} H(Y|X)$.

Proof. First, we list some notations and one property of typical sequences, and for more details please refer to [16].

Let x^n be a sequence with n elements drawn from the alphabet $\mathcal{X}$. Define the type of x^n by $\pi(x|x^n) = \frac{|\{i|x_i=x\}|}{n}$. Suppose the distribution of elements in $\mathcal{X}$ is $P(\mathcal{X})$, denote it as $X \sim P(\mathcal{X})$, and the set $\mathcal{T}^n_{P_X}$ of n-sequence with type $X \sim P(\mathcal{X})$ is defined as,

$$\mathcal{T}^n_{P_X} = \{x^n | \pi(x|x^n) = P(x), \forall x\}.$$

Let (x^n, y^n) be a pair of sequences with elements drawn from alphabets $(\mathcal{X}, \mathcal{Y})$. Define their joint type: $\pi(x, y|x^n, y^n) = \frac{|\{i|(x_i,y_i)=(x,y)\}|}{n}$ for $(x, y) \in \mathcal{X} \times \mathcal{Y}$. We denote

$$\mathcal{T}^n_{P_{XY}}(x^n) = \{y^n | \pi(x, y|x^n, y^n) = P(x, y), \forall (x, y)\}.$$

The following property is useful:

$$(n+1)^{-|\mathcal{X}||\mathcal{Y}|} 2^{nH(Y|X)} \leq |\mathcal{T}^n_{P_{XY}}(x^n)| \leq 2^{nH(Y|X)}. \tag{4.1}$$

For convenience, given $\forall \mathbf{u} \in \mathcal{S}_{n,m}$, write it as a mn sequence, that is $\mathbf{u} = (u^{-1}(1), \cdots, u^{-1}(mn))$.

Now, we formally prove Lemma 15 as follows:

Proof of the direct part

$\forall \mathbf{u} \in \bigcup_{i=1}^{M} \mathcal{B}_i$, we have $M \leq |\mathcal{B}_{n,m,r}^{(out)}(\mathbf{u})|$ according to corollary 14, and thus $R \leq \frac{\log_2 |\mathcal{B}_{n,m,r}^{(out)}(\mathbf{u})|}{mn}$.

Let Q be some joint probability distribution of $X, Y \in [n]$, and it has a marginal distribution P_X. Define $\mathcal{T}_Q^{mn}(\mathbf{u}) = \{\mathbf{v} \in \mathcal{S}_{n,m} | P_{\mathbf{u},\mathbf{v}} = Q\}$ for $\mathbf{u} \in \mathcal{S}_{n,m}$, where $P_{\mathbf{u},\mathbf{v}}$ denotes the joint type of $\mathbf{u}$ and $\mathbf{v}$.

Consider the following set of joint types $\mathcal{P}^{(out)}(\mathbf{u}, \mathbf{v}) = \{P_{\mathbf{u},\mathbf{v}} | \mathbf{v} \in \mathcal{B}_{n,m,r}^{(out)}(\mathbf{u})\}$. Define the following set of joint types $\mathcal{P}^{(out)}(n, r) = \{P_{XY} | P_X = P_Y, X \sim U(1, n), P(Y|X) = 0 \text{ if } Y - X \geq r + 1\}$. Clearly, $\mathcal{P}(n, r) \subset \mathcal{P}^{(out)}(n, r)$.

- Now, we prove that $\mathcal{P}^{(out)}(\mathbf{u}, \mathbf{v}) = \mathcal{P}^{(out)}(n, r)$.

First, given $\mathbf{u} \in \mathcal{S}_{n,m}$ and $\forall \mathbf{v} \in \mathcal{B}_{n,m,r}^{(out)}(\mathbf{u})$, consider $P_{\mathbf{u},\mathbf{v}}$. According to the definition of $\mathcal{S}_{n,m}$, the marginal distributions of $P_{\mathbf{u},\mathbf{v}}$, P_X, P_Y, satisfy $P_X = P_Y$ and $X \sim U(1, n)$. Based on $\mathbf{v} \in \mathcal{B}_{n,m,r}^{(out)}(\mathbf{u})$ and Theorem 11, we conclude that $P(Y|X) = 0$ if $Y - X \geq r + 1$. Therefore, $P_{\mathbf{u},\mathbf{v}} \in \mathcal{P}^{(out)}(n, r)$ and $\mathcal{P}^{(out)}(\mathbf{u}, \mathbf{v}) \subseteq \mathcal{P}^{(out)}(n, r)$.

Next, given $\forall Q \in \mathcal{P}^{(out)}(n, r)$, and $\mathbf{u} \in \mathcal{S}_{n,m}$, consider the typical sequence with joint type Q, $\mathcal{T}_Q^{mn}(\mathbf{u})$. Based on $P_X = P_Y, X \sim U(1, n)$, we obtain that $\mathcal{T}_Q^{mn}(\mathbf{u}) \in \mathcal{S}_{n,m}$ according to the definition of $\mathcal{S}_{n,m}$. According to $P(Y|X) = 0$ if $Y - X \geq r + 1$, $\mathcal{T}_Q^{mn}(\mathbf{u}) \in \mathcal{B}_{n,m,r}^{(out)}(\mathbf{u})$ based on Theorem 11. Thus $Q \in \mathcal{P}^{(out)}(\mathbf{u}, \mathbf{v})$ and $\mathcal{P}^{(out)}(n, r) \subseteq \mathcal{P}^{(out)}(\mathbf{u}, \mathbf{v})$. $\square$

Now, we partition $\mathcal{B}_{n,m,r}^{(out)}(\mathbf{u})$ as follows:

$$\mathcal{B}_{n,m,r}^{(out)}(\mathbf{u}) = \bigcup_{Q \in \mathcal{P}^{(out)}(n,r)} (\mathcal{B}_{n,m,r}^{(out)}(\mathbf{u}) \cap \mathcal{T}_Q^{mn}(\mathbf{u})).$$

Since the total number of the joint types among $\mathcal{T}_Q^{mn}(\mathbf{u})$ is up bounded by $(mn + 1)^{n^2}$ [16], and the size of each joint type is up bounded by $2^{mn \max_{P' \in \mathcal{P}^{(out)}(n,r)} H(Y|X)}$ according

to (4.1), we obtain that

$$|\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})| \leq (mn+1)^{n^2} 2^{mn \max_{P' \in \mathcal{P}^{(out)}(n,r)} H(Y|X)}.$$

That is,

$$\begin{aligned} R &\leq \frac{\log_2 |\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|}{nm} \\ &\leq n^2 \frac{\log_2 mn+1}{mn} + \max_{P' \in \mathcal{P}^{(out)}(n,r)} H(Y|X). \end{aligned} \tag{4.2}$$

On the other hand, $\forall \mathbf{u} \in \bigcup_{i=1}^{M} \mathcal{B}_i$, we have $M \leq |\mathcal{B}^{(in)}_{n,m,r}(\mathbf{u})|$ based on corollary 14, thus $R \leq \frac{\log_2 |\mathcal{B}^{(in)}_{n,m,r}(\mathbf{u})|}{mn}$. Similarly, we consider the following set of joint types $\mathcal{P}^{(in)}(\mathbf{u},\mathbf{v}) = \{P_{\mathbf{u},\mathbf{v}} | \mathbf{v} \in \mathcal{B}^{(in)}_{n,m,r}(\mathbf{u})\} = \mathcal{P}^{(in)}(n,r) \stackrel{def}{=} \{P_{XY} | P_X = P_Y, X \sim U(1,n), P(Y|X) = 0 \text{ if } X - Y \geq r+1\}$. Clearly, $\mathcal{P}(n,r) \subset \mathcal{P}^{(in)}(n,r)$.

Using the same technique as above, we can obtain that

$$\begin{aligned} R &\leq \frac{\log_2 |\mathcal{B}^{(in)}_{n,m,r}(\mathbf{u})|}{nm} \\ &\leq n^2 \frac{\log_2 mn+1}{mn} + \max_{P' \in \mathcal{P}^{(in)}(n,r)} H(Y|X). \end{aligned} \tag{4.3}$$

By (4.2) and (4.3), we know that

$$R \leq n^2 \frac{\log_2 mn+1}{mn} + \max_{P \in \mathcal{P}(n,r)} H(Y|X).$$

Proof of the converse part

We prove this part by a random code construction.

- We first prove that random codes satisfying the first two conditions of (n, m, M, r)

codes exist.

Now label the elements of $\mathcal{S}_{n,m}$ independently and uniformly with probability $1/M$ using one of the numbers $1, 2, ..., M$. The elements labeled with i form the set $\mathcal{B}_i$. M will be determined below.

We first calculate the probability $Pr(\mathbf{u}, i)$ that for a fixed $\mathbf{u}$ there does not exist a $\mathbf{v} \in \mathcal{B}_i \cap \mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$.

Clearly,

$$Pr(\mathbf{u}, i) = (1 - \frac{1}{M})^{|\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|}.$$

Then, the probability that we do not have such random code satisfying the first two conditions of (n, m, M, r) code is $\sum\limits_{\mathbf{u}\in\mathcal{S}_{n,m}} \sum\limits_{i=1}^{M} Pr(\mathbf{u}, i)$, which is

$$\sum_{\mathbf{u}\in\mathcal{S}_{n,m}} \sum_{i=1}^{M} Pr(\mathbf{u}, i) = |\mathcal{S}_{n,m}| M (1 - \frac{1}{M})^{|\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|} \leq 1,$$

if $M \leq (2 \log_2 |\mathcal{S}_{n,m}|)^{-1} |\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|$.

With the same skill of partitioning $\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})$, we obtain that $|\mathcal{B}^{(out)}_{n,m,r}(\mathbf{u})|$

$$\begin{aligned} &= \Big| \bigcup_{Q\in\mathcal{P}^{(out)}(n,r)} \mathcal{B}^{(out)}_{n,m,r}(\mathbf{u}) \bigcap \mathcal{T}^{mn}_{Q}(\mathbf{u}) \Big| \\ &\geq \max_{Q\in\mathcal{P}^{(out)}(n,r)} 2^{mnH(Y|X)} (mn+1)^{-n^2}, \end{aligned}$$

based on (4.1), thus we can choose M such that

$$\frac{1}{nm} \log_2 M \leq \max_{Q\in\mathcal{P}^{(out)}(n,r)} H(Y|X) - n \frac{\log_2 mn + 1}{m}, \tag{4.4}$$

and in this case we make sure that there exists random code satisfying the first two conditions of (n, m, M, r) codes. $\square$

By using similar techniques, we can choose M such that

$$\frac{1}{nm}\log_2 M \leq \max_{Q\in\mathcal{P}^{(in)}(n,r)} H(Y|X) - n\frac{\log_2 mn+1}{m}, \tag{4.5}$$

in which case we assure that there are random codes satisfying the first and the third condition of (n, m, M, r) codes.

Thus, according to (4.4) and (4.5) we can choose M such that

$$\frac{1}{nm}\log_2 M \leq \max_{Q\in\mathcal{P}(n,r)} H(Y|X) - n\frac{\log_2 mn+1}{m}.$$

to make sure there are random codes satisfying all three conditions of (n, m, M, r) codes.

$\square$

Note that the same skill to obtain (4.2) and (4.4) can be used to prove part of Lemma 22, and the skill to obtain (4.3) and (4.5) can be applied to part of Lemma 20.

The next lemma presents us the exact value of $\mathcal{R}(n, r)$, where the matrix $A \stackrel{def}{=} [a_{i,j}]_{n\times n}$, $a_{i,j} = 1$ if $|i - j| \leq r$, and 0 otherwise, and λ_A is the largest eigenvalue of the matrix A. The proof skills are similar to those of [2, 40], which can be applied to remaining parts of Lemma 22 and Lemma 20. Figure. 4.4 presents us a surface plot of $\mathcal{R}(n, r)$ for $n, r \in [50]$.

Lemma 16.

$$\mathcal{R}(n, r) = \log_2 \lambda_A.$$

Proof. Let (X, Y) be a pair of random variables with $P_{XY} \in \mathcal{P}(n, r)$. For $i, j \in [n]$, denote $Pr(X = i, Y = j) = q_{ij}$, $Pr(X = i) = p_i$, and $Pr(Y = j|X = i) = w_{ij}$. Then $q_{ij} = p_i \cdot w_{ij}$, and $W = [w_{ij}]_{n\times n}$ is a stochastic matrix.

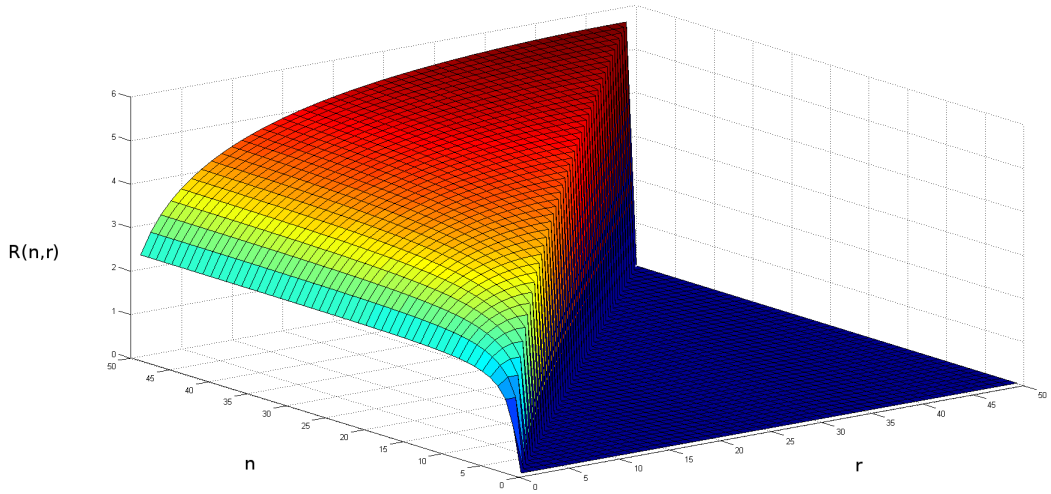


Figure 4.4: Surface plot of $\mathcal{R}(n,r)$ for $n, r \in \{1, 2, ..., 50\}$.

According to lemma 15, $\mathcal{R}(n,r)$ can be written down as the following form:

$$
\begin{aligned}
\max \quad &: \quad -\sum_{i=1}^{n} \sum_{j:|j-i|\leq r} q_{ij} \log_2(q_{ij} / \sum_{j:|j-i|\leq r} q_{ij}), \\
\text{s.t.} \quad &: \quad q_{ij} \geq 0, \text{ for } j : |j-i| \leq r \text{ and } i \in [n], \\
& \quad\;\; q_{ij} = 0, \text{ for } j : |j-i| \geq r+1 \text{ and } i \in [n], \\
& \quad\;\; \sum_{i=1}^{n} \sum_{j:|j-i|\leq r} q_{ij} = 1, \\
& \quad\;\; \sum_{i=1}^{n} q_{ij} = \sum_{i:|j-i|\leq r} q_{ji} = \frac{1}{n}, j \in [n].
\end{aligned} \tag{4.6}
$$

We solve this via Lagrange multiplier function:

$$L(q_{ij}, j : |j-i| \leq r, i \in [n]; s; t_1, ..., t_n)$$

$$
\begin{aligned}
&= -\sum_{i=1}^{n}\sum_{j:|j-i|\le r} q_{ij}\log_2(q_{ij}/\sum_{j:|j-i|\le r} q_{ij}) \\
&+ s\sum_{i=1}^{n}\sum_{j:|j-i|\le r} q_{ij} + \sum_{j=1}^{n} t_j(\sum_{i=1}^{n} q_{ij} - \sum_{i:|j-i|\le r} q_{ji}).
\end{aligned}
$$

Thus, we get

$$
\frac{\partial L}{\partial q_{ij}} = -\log_2 w_{ij} - \mu + t_j - t_i = 0,
$$

where $\mu = (\ln 2)^{-1}[\sum_{i=1}^{n}(|\{j : |i-j| \le r\}| - n)] - s$.

Then,

$$
w_{ij} = 2^{t_j - t_i - \mu}, \text{ for } j\text{: } |j-i| \le r, \text{ and } i \in [n]. \tag{4.7}
$$

Since $\sum_{j:|j-i|\le r} w_{ij} = 1$ for $i \in [n]$, we have

$$
\sum_{j:|j-i|\le r} 2^{t_j} = 2^{\mu}2^{t_i}, \text{ for } i \in [n], \tag{4.8}
$$

and

$$
w_{ij} = \frac{2^{t_j}}{\sum_{l:|l-i|\le r} 2^{t_l}} \text{ for } j : |j-i| \le r \text{ and } i \in [n].
$$

We interpret eq. (4.8) as indicating that $\{2^{t_i}\}$ is an eigenvector of the matrix $A = [a_{i,j}]_{n\times n}$, where $a_{i,j} = 1$ if$|j-i| \le r$ and 0 otherwise, thus 2^{μ} is the eigenvalue corresponding to $\{2^{t_i}\}$.

The fact that $W = [w_{ij}]_{n\times n}$ is a stochastic matrix implies that $(\frac{1}{n}, ..., \frac{1}{n})$ is its stationary distribution, which is actually p_i. This ensures that eq.(4.6) can be satisfied.

Substituting eq. (4.7) into the objective function and by simple arithmetic calculations, we obtain that

$$
-\sum_{i=1}^{n}\sum_{j:|j-i|\le r} q_{ij}\log_2(q_{ij}/\sum_{j:|j-i|\le r} q_{ij}) = \log_2 2^{\mu}.
$$

□

4.4.1.2 Code construction for $n = 3, m = 2$, and $r = 1$

Given $\mathbf{u}, \mathbf{v} \in \mathcal{S}_{n,m}$ and the cost constraint r, $\mathbf{u}$ *dominates* $\mathbf{v}$ if $\mathcal{C}(\mathbf{v} \to \mathbf{u}) \leq r$. Given a set $\mathcal{C} \subseteq \mathcal{S}_{n,m}$ and a $\mathbf{u} \in \mathcal{S}_{n,m}$ if $\exists \mathbf{v} \in \mathcal{C}$ such that $\mathbf{u}$ dominates $\mathbf{v}$, then $\mathbf{u}$ dominates $\mathcal{C}$. $\mathcal{C} \subseteq \mathcal{S}_{n,m}$ is a *dominating set* if $\forall \mathbf{u} \in \mathcal{S}_{n,m}, \exists \mathbf{v}, \mathbf{w} \in \mathcal{C}$ such that $\mathbf{u}$ dominates $\mathbf{v}$ and $\mathbf{w}$ dominates $\mathbf{u}$. Our goal is to divide $\mathcal{S}_{n,m}$ into the maximal number of disjoint dominating sets. In the following, we write a permutation of $\mathcal{S}_{n,m}$ as that of $\mathcal{S}_{nm}$, e.g., we write $(\{1, 2\}, \{3, 4\})$ as (1234).

Construction 17. Divide the 90 codes of $\mathcal{S}_{3,2}$ into 30 sets of 3 codes each, where each set is a *coset* of $\langle(135)(246)\rangle$, the *cyclic group generated by (135)(246)*, e.g., $(123456), (345612)$ and (561234) is a *cyclic group*. Map each set to a different symbol.

Theorem 18. Code construction 17 is optimal.

Proof. Let $g = (135)(246)$ and $G = \langle g \rangle = \{g^0, g^1, g^2\}$, where $g^0 = I$ is the identical permutation. $\forall \mathbf{u} \in \mathcal{S}_{3,2}$, $\mathbf{u}G \stackrel{def}{=} \{\mathbf{u}g^0, \mathbf{u}g^1, \mathbf{u}g^2\}$ is the set of codes.

First, we prove that $\mathbf{u}G$ is a dominating set. Denote $\mathbf{u}G = \{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$, then $\mathbf{x}_3, \mathbf{y}_3, \mathbf{z}_3$ forms a partition of $\{1, 2, ..., 6\}$. Thus $\forall \mathbf{v} \in \mathcal{S}_{3,2}$, $\exists \mathbf{w} \in \mathbf{u}G$ such that $\mathbf{v}_3 \bigcap \mathbf{w}_3 \neq \emptyset$, then $\mathbf{v}$ dominates $\mathbf{w}g$ or $\mathbf{w}$.

Similarly, $\forall \mathbf{v} \in \mathcal{S}_{3,2}$, $\exists \mathbf{w} \in \mathbf{u}G$ such that $\mathbf{v}_1 \bigcap \mathbf{w}_1 \neq \emptyset$, then $\mathbf{v}$ is dominated by $\mathbf{w}g$ or $\mathbf{w}$. This finishes the dominating set proof.

$|\mathcal{B}^{(in)}_{3,2,1}(\mathbf{u})| = 36$, thus the dominating set size should be at least $\lfloor \frac{90}{36} \rfloor$ = 3. Therefore the code is optimal. □

4.4.2 Two variants of (n, m, M, r) codes

In this subsection, we present two variants of worst case rewriting codes including their definitions, storage capacity characterizations, and their exact values. Since the cor-

responding proofs are exactly the same as subsection A of section 4.4, we omit them.

4.4.2.1 Variant one

Denote a code in $\mathcal{S}_{n,m}$ with rewriting cost r and cardinality M as an $(n, m, M, r)^{(in)}$ code, and we formally define it as follows:

Definition 19. An $(n, m, M, r)^{(in)}$ rewriting code for CRM, where $r \in [n-1]$, is a collection of subsets $\mathcal{B} = \{\mathcal{B}_i | i \in [M]\}$ where $\mathcal{B}_i \subseteq \mathcal{S}_{n,m}$, and it represents data i, such that

- $\mathcal{B}_i \bigcap \mathcal{B}_j = \emptyset$ for $i \neq j$, and $\bigcup_{i=1}^{M} \mathcal{B}_i = \mathcal{S}_{n,m}$.
- $\forall \mathbf{u} \in \bigcup_{i=1}^{M} \mathcal{B}_i$, and $\forall j \in [M], \exists \mathbf{v} \in \mathcal{B}_j$ such that $\mathcal{C}(\mathbf{v} \to \mathbf{u}) \leq r$.

That is, we partition $\mathcal{S}_{n,m}$ into M disjoint sets, each set represents data, and every permutation can be programmed *from* some permutation of any set $\mathcal{B}_i$ for $i \in [M]$ with the cost constraint r.

The rate of the $(n, m, M, r)^{(in)}$ code is defined as $R^{(in)} = \frac{\log_2 M}{nm}$, and its storage capacity, $\mathcal{R}^{in}(n, r)$, is defined as $\mathcal{R}^{(in)}(n, r) = \lim_{m \to \infty} \frac{\log_2 M}{nm}$.

$A^{(in)} \stackrel{def}{=} [a_{i,j}^{(in)}]_{n \times n}$, $a_{i,j}^{(in)} = 1$ if $i - j \leq r$, and 0 otherwise, and $\lambda_{A^{(in)}}$ is the largest eigenvalue of the matrix $A^{(in)}$. The following lemma presents us the characterization and the exact value of $\mathcal{R}^{in}(n, r)$.

Lemma 20. $\mathcal{R}^{(in)}(n, r) = \max_{P_{XY} \in \mathcal{P}^{(in)}(n,r)} H(Y|X) = \log_2 \lambda_{A^{(in)}}$.

4.4.2.2 Variant two

Denote a code in $\mathcal{S}_{n,m}$ with rewriting cost r and cardinality M as an $(n, m, M, r)^{(out)}$ code, and we formally define it as follows:

Definition 21. An $(n, m, M, r)^{(out)}$ rewriting code for CRM, where $r \in [n-1]$, is a collection of subsets $\mathcal{B} = \{\mathcal{B}_i | i \in [M]\}$ where $\mathcal{B}_i \subseteq \mathcal{S}_{n,m}$, and it represents data i, such that

- $\mathcal{B}_i \bigcap \mathcal{B}_j = \emptyset$ for $i \neq j$, and $\bigcup_{i=1}^{M} \mathcal{B}_i = \mathcal{S}_{n,m}$.
- $\forall \mathbf{u} \in \bigcup_{i=1}^{M} \mathcal{B}_i$, and $\forall j \in [M], \exists \mathbf{v} \in \mathcal{B}_j$ such that $\mathcal{C}(\mathbf{u} \to \mathbf{v}) \leq r$.

That is, we partition $\mathcal{S}_{n,m}$ into M disjoint sets, each set represents data, and every permutation can be programmed *to* some permutation of any set $\mathcal{B}_i$ for $i \in [M]$ with the cost constraint r.

The rate of the $(n, m, M, r)^{(out)}$ code is defined as $R^{(out)} = \frac{\log_2 M}{nm}$, and its storage capacity, $\mathcal{R}^{out}(n, r)$, is defined as $\mathcal{R}^{(out)}(n, r) = \lim_{m \to \infty} \frac{\log_2 M}{nm}$.

$A^{(out)} \stackrel{def}{=} [a_{i,j}^{(in)}]_{n \times n}$, $a_{i,j}^{(in)} = 1$ if $j - i \leq r$, and 0 otherwise, and $\lambda_{A^{(out)}}$ is the largest eigenvalue of the matrix $A^{(out)}$. The following lemma presents us the characterization and the exact value of $\mathcal{R}^{out}(n, r)$.

Lemma 22. $\mathcal{R}^{(out)}(n, r) = \max_{P_{XY} \in \mathcal{P}^{(out)}(n,r)} H(Y|X) = \log_2 \lambda_{A^{(out)}}$.

4.5 Conclusion

We explore a generalized scheme of RM, CRM. General worst case code construction, average case code construction, and error correction codes are our future work.

5. POLAR CODES ARE OPTIMAL FOR WRITE-EFFICIENT MEMORIES

5.1 Introduction

Write-efficient memories (WEM) are models for storing and updating information on a rewritable medium with constraints. WEM is widely used in data storage area: in flash memories, write-once memories (WOM) [61], and the recently proposed compressed rank modulation (CRM) [46] are examples of WEM; for phase change memories, they are fit for WEM: changing the cell state in one direction does not cost anything, while changing states in the opposite direction has a cost, and some cost constraint during updating is required considering issues of reliability and endurance [44]. The recent proposed scheme, that polar codes are constructed for WOM codes achieving capacity [9], motivates us to construct codes for WEM.

5.1.1 WEM with a maximal rewriting cost constraint

Let $\mathcal{X} = \{0, 1, ..., q-1\}$ be the storage alphabet. $\mathcal{R}_+ = [0, +\infty)$, and $\varphi : \mathcal{X} \times \mathcal{X} \to \mathcal{R}_+$ is the rewriting cost function, measuring the time or enery cost of changing from one state to another state. Suppose that a memory consists of N cells. Given one cell state, $x_0^{N-1} \stackrel{def}{=} (x_0, x_1, ..., x_{N-1}) \in \mathcal{X}^N$, and another cell state $y_0^{N-1} \in \mathcal{X}^N$, the rewriting cost of changing from x_0^{N-1} to y_0^{N-1} is measured by $\varphi(x_0^{N-1}, y_0^{N-1}) = \sum_{i=0}^{N-1} \varphi(x_i, y_i)$.

Let $\mathcal{D} \subseteq \mathbb{N}$. We use $\mathcal{D}$ to denote the $|\mathcal{D}|$ possible values of the data stored in the N cells. Let the decoding function be $\mathbf{D} : \mathcal{X}^N \to \mathcal{D}$, which maps the N cells' levels to the data they represent. Let the rewriting function be $\mathbf{R} : \mathcal{X}^N \times \mathcal{D} \to \mathcal{X}^N$, which changes the N cells' levels to represent the new input data.

Definition 23. [27] An (N, M, q, d) WEM code consists of

- $\mathcal{D} = \{0, 1, \cdots, M-1\}$ and $\bigcup_{i=0}^{M-1} \mathcal{C}_i$, where $\mathcal{C}_i \subseteq \mathcal{X}^N$ is the set of codewords

representing data i. We require $\forall i \neq j, \mathcal{C}_i \bigcap \mathcal{C}_j = \emptyset$;

- A rewriting function $\mathbf{R}(i, x_0^{N-1})$ such that $\varphi(x_0^{N-1}, \mathbf{R}(i, x_0^{N-1})) \leq Nd$ for any $i \in \mathcal{D}$ and $x_0^{N-1} \in \mathcal{X}^N$;

- A decoding function $\mathbf{D}(y_0^{N-1})$ such that $\mathbf{D}(\mathbf{R}(x_0^{N-1}, i)) = i$ for any $i \in \mathcal{D}$.

The rewriting rate of an (N, M, q, d) WEM code is defined as $\mathcal{R} = \frac{\log_2 M}{N}$, $\mathcal{R}$ is achievable if there exists an (N, M, q, d) code as $N \to \infty$, and the *rewriting capacity* function, $\mathcal{R}(q, d)$, is the supremum of all achievable rates.

Let $\mathcal{P}(\mathcal{X} \times \mathcal{X})$ be the set of joint probability distributions over $\mathcal{X} \times \mathcal{X}$. For a pair of random variables $(X, Y) \in (\mathcal{X}, \mathcal{X})$, let P_{XY} denote the joint probability distribution, let P_Y denote the marginal distribution, $P_{Y|X}$ denote the conditional probability distribution, and $E(\cdot)$ denote the expectation operator. If X is uniformly distributed over $\{0, 1, ..., q-1\}$, denote it as $X \sim U(q)$.

Define $\mathcal{P}(q, d) = \{P_{XY} \in \mathcal{P}(\mathcal{X} \times \mathcal{X}) : P_X = P_Y, E(\varphi(X, Y)) \leq d\}$. $\mathcal{R}(q, d)$ is determined as [27]: $\mathcal{R}(q, d) = \max\limits_{P_{XY} \in \mathcal{P}(q,d)} H(Y|X)$.

For WOM codes, the cell state can only increase but not decrease. WOM codes are special cases of WEM codes with the cost function defined appropriately: for a WOM cell if we update it from $x \in \mathcal{X}$ to $y \in \mathcal{X}$, the cost is measured by $\varphi(x, y) = 0$ if $y \geq x$, and ∞ otherwise. Therefore, WOM codes are such WEM codes with $\varphi(\cdot)$ defined previously, and d is equal to 0.

In this work, we focus on symmetric WEM. Recall that the rewriting capacity of WEM is $\mathcal{R}(D) = \max\limits_{P_{XY} \in \mathcal{P}(D)} H(Y|X)$ [2]. Analogous to a symmetric channel, a symmetric WEM is such a WEM that its rewriting capacity is achieved when current cell state alphabet (i.e., X) and updated cell state alphabet (i.e., Y) are uniformly distributed. That is, the rewriting capacity of symmetric WEM is $\mathcal{R}^s(q, d)$:

Definition 24. For $X, Y \in \mathcal{X}$ with P_X, P_Y and P_{XY}, and $\varphi : \mathcal{X} \times \mathcal{X} \to \mathcal{R}_+$, $\mathcal{R}^s(q,d) = \max\limits_{P_{XY} \in \mathcal{P}^s(q,d)} H(Y|X)$, where $\mathcal{P}^s(q,d) \stackrel{def}{=} \{P_{XY} \in \mathcal{P}(\mathcal{X} \times \mathcal{X}) : P_X = P_Y, X \sim U(q), E(\varphi(X,Y)) \leq d\}$.

We present an example of WEM with $\mathcal{R}^s(q,d)$ below.

Denote by $\mathcal{S}_{q,m}$ the set of $\frac{(mq)!}{m!^q}$ permutations over $\{\overbrace{1,1,...,1}^{m}, ..., \overbrace{q,q,...,q}^{m}\}$. We abuse the notation of $u_0^{qm-1} \stackrel{def}{=} [u_0, u_1, ..., u_{qm-1}]$ to denote an element of $\mathcal{S}_{q,m}$, which denotes the mapping $i \to u_i$.

Example 25. A rewriting code for CRM with the maximal rewriting cost constraint, (q, m, M, d), is defined by replacing $\mathcal{X}^N$ by $\mathcal{S}_{q,m}$, $\varphi(\cdot)$ by Chebyshev distance between $u_0^{qm-1}, v_0^{qm-1} \in \mathcal{S}_{q,m}$, $d_\infty(u_0^{qm-1}, v_0^{qm-1}) \stackrel{def}{=} \max\limits_{j \in \{0,1,...,qm-1\}} |u_j - v_j|$, and Nd by d in definition 8.4.1.1.

Note that (q, m, M, d) is actually an instance of WEM by defining $\varphi(\cdot)$ and d appropriately: for $x, y \in \mathcal{X}$, let $\varphi(x,y) = 0$ if $|x - y| \leq d$, and ∞ otherwise. Now the (q, m, M, d) CRM is an $(qm, M, q, 0)$ WEM with $\mathcal{X}^N$ replaced by $\mathcal{S}_{q,m}$, and $\varphi(\cdot)$ is defined previously.

Denote the rewriting capacity function for CRM with the maximal rewriting cost constraint as $\mathcal{R}^c(q,d)$, which is the largest d-admissible rate when $m \to \infty$, and it is proved that $\mathcal{R}^c(q,d) = \mathcal{R}^s(q,d)$ [46].

5.1.2 WEM with an average rewriting cost constraint

Assume the sequence of data written to the storage medium is $\{M_1, \cdots, M_t\}$, where we assume M_i for $1 \leq i \leq t$ is uniformly distributed over $\mathcal{D}$, and the average rewriting cost is $\bar{D} \stackrel{def}{=} \lim\limits_{t \to \infty} \frac{1}{t} \sum\limits_{i=1}^{t} \varphi(x_0^{N-1}(i), \mathbf{R}(M_i, x_0^{N-1}(i)))$, where $x_0^{N-1}(i)$ is the current cell states before the ith update. By assuming the stationary distribution of cell levels x_0^{N-1} is $\pi(x_0^{N-1})$, $\bar{D} = \sum\limits_{x_0^{N-1}} \pi(x_0^{N-1}) \sum\limits_{j \in \mathcal{D}} \bar{D}_j(x_0^{N-1})$, where $\bar{D}_j(x_0^{N-1})$ is the average rewriting cost of updating cell levels x_0^{N-1} to a codeword representing $j \in \mathcal{D}$.

The definition of WEM with an average rewriting cost constraint is defined as follows:

Definition 26. An $(N, M, q, d)_{ave}$ WEM code consists of

- $\mathcal{D} = \{0, 1, \cdots, M-1\}$ and $\bigcup_{i=0}^{M-1} \mathcal{C}_i$, where $\mathcal{C}_i \subseteq \mathcal{X}^N$ is the set of codewords representing data i. We require $\forall i \neq j, \mathcal{C}_i \bigcap \mathcal{C}_j = \emptyset$;
- A rewriting function $\mathbf{R}(i, x_0^{N-1})$ such that $\bar{D} \leq d$.
- A decoding function $\mathbf{D}(y_0^{N-1})$ such that $\mathbf{D}(\mathbf{R}(x_0^{N-1}, i)) = i$ for any $i \in \mathcal{D}$.

The rewriting rate of an $(N, M, q, d)_{ave}$ code is defined as $\mathcal{R}_{ave} = \frac{\log_2 M}{N}$, and its rewriting capacity function, $\mathcal{R}_{ave}(q, d)$, is defined as the largest d-admissible rate when $N \to \infty$. It is proved that $\mathcal{R}_{ave}(q, d) = \mathcal{R}(q, d)$ [2]. Similarly, we focus on the symmetric rewriting capacity function, $\mathcal{R}_{ave}^s(q, d)$, as defined in definition 24.

5.1.3 The outline

The connection between rate-distortion theorem and rewriting capacity theorem is presented in Section 5.2. The binary polar WEM codes with an average rewriting cost constraint and a maximal rewriting cost constraint are presented in subsection A and B of Section 5.3, respectively. The q-ary polar WEM codes, based on the recently proposed q-ary polar codes [57], are presented in subsection A and B of Section 5.4 for an average rewriting cost constraint and a maximal rewriting cost constraint, respectively. The conclusion is obtained in Section 5.5.

5.2 Lossy Source Coding and its Duality with WEM

In this section, we present briefly background of lossy source coding and its duality with WEM, which inspires code constructions for WEM.

Let $\mathcal{X}$ also denote the variable space, and $\mathcal{Y}$ denotes the reconstruction space. Let $d : \mathcal{Y} \times \mathcal{X} \to \mathcal{R}_+$ denote the distortion function, and the distortion among a vector x_0^{N-1} and its reconstructed vector y_0^{N-1} is $d(x_0^{N-1}, y_0^{N-1}) = \frac{1}{N} \sum_{i=0}^{N-1} d(x_i, y_i)$.

A (q^{NR}, N) rate distortion code consists of an encoding function $f_N : \mathcal{X}^N \to \{0, 1, \cdots, q^{NR} - 1\}$ and a reproduction function $g_N : \{0, 1, ..., q^{NR} - 1\} \to \mathcal{Y}^N$. The associated distortion is defined as $E(d(X_0^{N-1}, g_N(f_N(X_0^{N-1}))))$, where the expectation is with respect to the probability distribution on $\mathcal{X}^N$. $R(q, D)$ is the infimum of rates R such that $E(d(X_0^{N-1}, g_N(f_N(X_0^{N-1}))))$ is at most D as $N \to \infty$.

Let $P(q, D) \stackrel{def}{=} \{P_{XY} \in \mathcal{P}(\mathcal{X} \times \mathcal{Y}) : E(d(X, Y)) \leq D\}$, and $R(q, D)$ is determined as $\min\limits_{P_{XY} \in P(q,D)} I(X; Y)$ [16].

We focus on lossy compression with the *double symmetric rate-distortion* $R^s(q, D)$. It is defined as for $(X, Y) \in (\mathcal{X} \times \mathcal{X})$ and $d(x, y)$, $R^s(q, D) = \min\limits_{P_{XY} \in P^s(q,D)} I(Y; X)$, where $P^s(q, D) \stackrel{def}{=} \{P_{XY} \in \mathcal{P}(\mathcal{X} \times \mathcal{X}) : P_X = P_Y, X \sim U(q), E(d(X, Y)) \leq D\}$.

The duality between $\mathcal{R}^s(q, D)$ and $R^s(q, D)$ is captured by the following lemma, the proof of which is ommited due to being straightforward.

Lemma 27. With the same $d(\cdot)$ and $\varphi(\cdot)$,

$$\mathcal{R}^s(q, D) + R^s(q, D) = \log_2 q. \tag{5.1}$$

The inspiration we obtain from the above lemma is that for a q-ary cell with lossy compression requirement, on average $R^s(q, D)$ part of it is to represent data, and the remaining part $\mathcal{R}^s(q, D)$ is not used. Therefore, if we store data in $\mathcal{R}^S(q, D)$ of a cell and the remaining part is equipped with same encoding/reconstruction methods as lossy compression, the rewriting constraint can be satisfied.

5.3 Polar Codes are Optimal for Binary WEM

Inspired by lemma 27, we show that polar codes can be used to construct binary WEM codes with $\mathcal{R}^s(2, D)$ in a way related to the code construction for lossy source coding of [42] in this section.

5.3.1 A code construction for binary WEM with an average rewriting cost constraint

In this part, we start with background of polar code, then polar code construction for lossy compression, and finally we present code construction for WEM as well as its proof.

5.3.1.1 A brief introduction of polar codes [5]

Let $W : \{0,1\} \to \mathcal{Y}$ be a binary-input discrete memoryless channel for some output alphabet $\mathcal{Y}$. Let $I(W) \in [0,1]$ denote the mutual information between the input and output of W with a uniform distribution on the input. Let G_N denote n-th Kronecker product of $\left(\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right)$. Let the Bhattacharyya parameter $Z(W) = \sum_{y \in \mathcal{X}} \sqrt{W_{Y|X}(y|0)W_{Y|X}(y|1)}$.

The polar code is a linear code given by $C_N(F, u_F) = \{x_0^{N-1} = u_0^{N-1}G_N : u_{F^c} \in \{0,1\}^{|F^c|}\}$, where $\forall F \subseteq \{0,1,...,N-1\}$, u_F denotes the subvector $u_i : i \in F$, and $u_F \in \{0,1\}^{|F|}$. By convention, F is called the *frozen set* and u_F is called the *frozen set value*. The polar code ensemble, i.e., collection of polar code with all possible frozen set value, is $C_N(F) = \{C_N(F, u_F), \forall u_F \in \{0,1\}^{|F|}\}$.

The secret of polar codes achieving $I(W)$ lies in how to select F: define $W_N^{(i)} : \{0,1\} \to \mathcal{Y}^N \times \{0,1\}^i$ as sub-channel i with input u_i, output (y_0^{N-1}, u_0^{i-1}) and transition probabilities $W_N^{(i)}(y_0^{N-1}, u_0^{i-1}|u_i) \stackrel{def}{=} \frac{1}{2^{N-1}} \sum_{u_{i+1}^{N-1}} \prod_{i=0}^{N-1} W(y_i|(u_0^{N-1}G_N)_i)$, and $(u_0^{N-1}G_N)_i$ denotes the i-th element of $u_0^{N-1}G_N$; The fraction of $\{W_N^{(i)}\}$ that are approaching noiseless, i.e., $Z(W_N^{(i)}) \leq 2^{-N^\beta}$ for $0 \leq \beta \leq \frac{1}{2}$, approaches $I(W)$; The F is chozen as indecies with large $Z(W_N^{(i)})$, that is $F \stackrel{def}{=} \{i \in \{0,1...,N-1\} : Z(W_N^{(i)}) \geq 2^{-N^\beta}\}$ for $\beta \leq 1/2$.

The encoding is done by a linear transformation, and the decoding is done by successive cancellation (SC).

5.3.1.2 Polar codes on lossy source coding

We sketch the binary polar code construction for lossy source coding as follows, and for more details interested readers can refer to [42].

Note that $R^s(2, D)$ can be obtained through the following optimization function:

$$\begin{aligned} \min \quad &: \quad I(Y;X), \\ \text{s.t.} \quad &: \quad \sum_x \frac{1}{2} P(y|x) = \sum_y \frac{1}{2} P(x|y) = \frac{1}{2}, \\ &\quad \sum_x \sum_y \frac{1}{2} P(y|x) d(y,x) \leq D. \end{aligned} \tag{5.2}$$

Let $P^*(y|x)$ be the probability distribution minimizing the objective function of (5.2). $P^*(y|x)$ plays the role of a channel. By convention, we call $P^*(y|x)$ *test channel*, and denote it as $W(y|x)$.

Now, we construct the source code with $R^s(2, D)$ using the polar code for $W(y|x)$, and denote the rate of the source code by R: set F as $N(1-R)$ sub-channel indexes with the highest $Z(W_N^{(i)})$, set F^c as the remaining NR sub-channel indexes, and set u_F to an arbitrary value.

A source codeword y_0^{N-1} is mapped to a codeword $x_0^{N-1} \in C_N(F, u_F)$, and x_0^{N-1} is described by the index $u_{F^c} = (x_0^{N-1} G_N^{-1})_{F^c}$.

The reproduction process is done as follows: we do SC encoding scheme, $\hat{u}_0^{N-1} = \hat{U}(y_0^{N-1}, u_F)$, that is for each k in the range 0 till $N-1$:

1. If $k \in F$, set $\hat{u}_k = u_k$;

2. Else, set $\hat{u}_k = m$ with the posterior $P(m|\hat{u}_0^{i-1}, y_0^{N-1})$.

The reproduction codeword is $\hat{u}_0^{N-1} G_N$.

Thus, the average distortion $D_N(F, u_F)$ (over the source codeword y_0^{N-1} and the en-

coder randomness for the code $C_N(F, u_F)$) is :

$$\sum_{y_0^{N-1}} P(y_0^{N-1}) \sum_{\hat{u}_{F^c}} d(y_0^{N-1}, \hat{u}_0^{N-1} G_N) \prod_{i \in F^c} P(\hat{u}_i | \hat{u}_0^{i-1}, y_0^{N-1}),$$

where $\hat{u}_F = u_F$.

The expectation of $D_N(F, u_F)$ over the uniform choice of u_F, $D_N(F)$, is $D_N(F) = \sum_{u_F} \frac{1}{2^{|F|}} D_N(F, u_F)$. Let $Q_{U_0^{N-1}, Y_0^{N-1}}$ denote the distribution defined by $Q_{Y_0^{N-1}}(y_0^{N-1}) = P(y_0^{N-1})$, and $Q_{U_0^{N-1}|Y_0^{N-1}}$ by

$$Q(u_i | u_0^{i-1}, y_0^{N-1}) = \begin{cases} \frac{1}{2}, & \text{if } i \in F, \\ P(u_i | u_0^{i-1}, y_0^{N-1}), & \text{otherwise.} \end{cases}$$

Thus, $D_N(F)$ is equivalent to $E_Q(d(y_0^{N-1}, u_0^{N-1} G_N))$, where $E_Q(\cdot)$ denotes the expectation with respect to the distribution $Q_{U_0^{N-1}, Y_0^{N-1}}$.

It is proved that $D_N(F) \leq D + O(2^{-N^\beta})$ for $0 \leq \beta \leq \frac{1}{2}$, the rate of the above scheme is $R = \frac{|F^c|}{N}$, and polar codes achieve the rate-distortion bound by Theorem 3 of [42].

On the other hand, Theorem 2 of [9] further states the strong converse result of the rate distortion theory. More precisely, if y_0^{N-1} is uniformly distributed over $\{0, 1\}^N$, then $\forall \delta > 0$, $0 < \beta < \frac{1}{2}$, N sufficiently large, and with the above SC encoding process and the induced Q, $Q(d(u_0^{N-1} G_N, y_0^{N-1}) \geq D + \delta) < 2^{-N^\beta}$. That is, for $\forall y_0^{N-1}$, the above reproduction process obtains $x_0^{N-1} = u_0^{N-1} G_N$ such that the distortion $d(x_0^{N-1}, y_0^{N-1})$ is less than D almost by sure.

5.3.1.3 The code construction

We formly present our code construction, theoretical performances and experimental performances in this part.

We focus on the code construction with *symmetric rewriting cost function*, which satisfies $\forall x, y, z \in \{0, 1\}, \varphi(x, y) = \varphi(x + z, y + z)$, where $+$ is over GF(2).

To construct codes for WEM with $\mathcal{R}^s(2, D)$, we utilize its related form $R^s(2, D)$ in (5.1) and the test channel $W(y|x)$ for $R^s(2, D)$. Note that $W(y|x)$ is a binary symmetric channel.

The code construction for $(N, M, 2, D)_{ave}$ with rate $\mathcal{R}$ is presented in Algorithm 5.3.1:

Algorithm 5.3.1 A code construction for $(N, M, 2, D)_{ave}$ WEM

1: Set F as $N\mathcal{R}$ sub-channel indexes with the highest $Z(W_N^{(i)})$, and set F^c as the remaining $N(1 - \mathcal{R})$ sub-channel indexes.
2: The $(N, M, 2, D)_{ave}$ code is $\mathcal{C} = \{\mathcal{C}_i : \mathcal{C}_i = C_N(F, u_F(i))\}$, where $u_F(i)$ is the binary representation form of i for $i \in \{0, 1, ..., M - 1\}$.

As G_N is of full rank [5], $\forall u_F(i) \neq u_F(j)$, $C_N(F, u_F(i)) \bigcap C_N(F, u_F(j)) = \emptyset$. That is the polar code ensemble is the WEM code, and each polar code of the ensemble is a set of WEM to represent one data with frozen set value.

The rewriting operation is done by successive cancellation encoder to make sure the rewriting cost constraint is satisfied, and it is presented in Algorithm 5.3.2, where the dither g_0^{N-1} is inspired by [9].

Algorithm 5.3.2 The rewriting operation $y_0^{N-1} = \mathbf{R}(x_0^{N-1}, i)$.

1: Let $v_0^{N-1} = x_0^{N-1} + g_0^{N-1}$, where g_0^{N-1} is a common-known random vector.
2: SC encoding v_0^{N-1}, and this results $u_0^{N-1} = \hat{U}(v_0^{N-1}, u_F(i))$ and $\hat{y}_0^{N-1} = u_0^{N-1} G_N$.
3: $y_0^{N-1} = \hat{y}_0^{N-1} + g_0^{N-1}$.

The decoding operation is to retrieve information bits in frozen set, and it is presented

in Algorithm 5.3.3.

Algorithm 5.3.3 The decoding operation $u_F(i) = \mathbf{D}(x_0^{N-1})$.

1: $y_0^{N-1} = x_0^{N-1} + g_0^{N-1}$.
2: $u_F(i) = (y_0^{N-1} G_N^{-1})_F$.

The following theorem guarantees that decoding operation is a valid operation.

Lemma 28. $\mathbf{D}(\mathbf{R}(y_0^{N-1}, i)) = i$ holds for each rewriting.

Proof. From the rewriting operation, $y_0^{N-1} = \hat{y}_0^{N-1} + g_0^{N-1} = u_0^{N-1} G_N + g_0^{N-1} = \hat{U}(v_0^{N-1}, u_F(i)) G_N + g_0^{N-1}$, from the decoding function $\hat{U}(v_0^{N-1}, u_F(i)) G_N + g_0^{N-1} + g_0^{N-1}$, which is $\hat{U}(v_0^{N-1}, u_F(i)) G_N$, thus the decoding result is i. □

5.3.1.4 The average rewriting cost analysis

From $y_0^{N-1} = \mathbf{R}(x_0^{N-1}, i)$, we know that $y_0^{N-1} = \hat{U}(v_0^{N-1}, u_F(i)) G_N + g_0^{N-1} = \hat{U}(x_0^{N-1} + g_0^{N-1}, u_F(i)) G_N + g_0^{N-1}$, thus $\varphi(x_0^{N-1}, y_0^{N-1})$ is $\varphi(x_0^{N-1}, \hat{U}(x_0^{N-1} + g_0^{N-1}, u_F(i)) G_N + g_0^{N-1})$, which is $\varphi(x_0^{N-1} + g_0^{N-1}, \hat{U}(x_0^{N-1} + g_0^{N-1}, u_F(i)) G_N)$ due to $\varphi(\cdot)$ being symmetric. Denote $w_0^{N-1} = x_0^{N-1} + g_0^{N-1}$, thus $\varphi(x_0^{N-1}, y_0^{N-1}) = \varphi(w_0^{N-1}, \hat{U}(w_0^{N-1}, u_F(i)) G_N)$.

The average rewriting cost $\bar{D}$

$$
\begin{aligned}
&= \lim_{t \to \infty} \frac{1}{Nt} \sum_{i=1}^{t} E(\varphi(x_0^{N-1}(i), x_0^{N-1}(i+1))), \\
&= \lim_{t \to \infty} \frac{1}{Nt} \sum_{i=1}^{t} E(\varphi(w_0^{N-1} \hat{U}(w_0^{N-1}, u_F(M_{i+1})) G_N)), \\
&= \sum_{w_0^{N-1}} \pi(w_0^{N-1}) \sum_{j} \bar{D}_j(w_0^{N-1}). \qquad (5.3)
\end{aligned}
$$

Let us focus on $\bar{D}_j(w_0^{N-1})$, which is the average (in this case, over the probability of rewriting to data j and the randomness of the encoder) rewriting cost of updating w_0^{N-1} to a codeword representing j. Thus $\bar{D}_j(w_0^{N-1}) =$

$$\frac{1}{2^{|F|}} \sum_{u_{F^c}} \varphi(w_0^{N-1}, u_0^{N-1} G_N) \prod_{i \in F^c} P(u_i | u_0^{i-1}, w_0^{N-1}).$$

Therefore, interpreting $\varphi(\cdot)$ as $d(\cdot)$, $\bar{D}$ is actually the average distortion over the ensemble $C_N(F)$, $D_N(F)$.

The following lemma from [42] is to bound $\bar{D}$:

Lemma 29. [42] Let $\beta < \frac{1}{2}$ be a constant and let $\sigma_N = \frac{1}{2N} 2^{-N^\beta}$. When the polar code for the source code with $R^s(2, D)$ is constructed with F:

$$F = \{i \in \{0, 1, ..., N-1\} : Z(W_N^{(i)}) \geq 1 - 2\sigma_N^2\},$$

then $D_N(F) \leq D + O(2^{-N^\beta})$, where D is the average rewriting cost constraint.

Therefore, with the same β, σ_N, F, and polar code ensemble $C_N(F)$, $\bar{D} \leq D + O(2^{-N^\beta})$.

According to [5], $\lim_{N=2^n, n\to\infty} \frac{|F^c|}{N} =$

$$\begin{aligned} & \lim_{N\to\infty} \frac{|\{i \in \{0, 1, ..., N\} : Z(W_N^{(i)}) \leq 2^{-2^{n\beta}}\}|}{N} \\ = \; & I(W) = R^s(2, D), \end{aligned}$$

thus this implies that for N sufficiently large $\exists$ a set F such that $\frac{|F^c|}{N} \geq R^s(2, D) - \epsilon$, $\forall \epsilon > 0$. In other words, the rate of the constructed WEM code, $\mathcal{R} = \frac{|F|}{N} = 1 - \frac{|F^c|}{N} \leq \mathcal{R}^s(2, D) + \epsilon$.

The complexity of the decoding and the rewriting operation is of the order $O(N \log N)$ according to [5].

We conclude the theoretical performance of the above polar WEM code as follows:

Theorem 30. For a binary symmetric rewriting cost function $\varphi : \mathcal{X} \times \mathcal{X} \to \mathcal{R}_+$, fix a rewriting cost D and $0 < \beta < \frac{1}{2}$. For any rate $\mathcal{R} < \mathcal{R}^s(2, D)$, there exists a sequence of polar WEM codes of length N and rate $R \leq \mathcal{R}$, so that under the above rewriting operation, $\bar{D}$ satisfies $\bar{D} \leq D + O(2^{-N^\beta})$. The decoding and rewriting operation complexity of the codes is $O(N \log N)$.

5.3.1.5 Experimental performance

The experimental performance is presented in Figure 5.1, where the rewriting cost function is the Hamming distance between old and new cell states, the upper bound of $\mathcal{C}(2, d)$ is $H(d)$ [2]. The polar code is constructed based on [68], and decoding is based on [67].

We can see that the rates and the average rewriting costs approach those of points of $H(d)$ as the length of codeword increases. Longer codewords are needed for further approaching the lower bound.

5.3.2 A code construction for binary WEM with a maximal rewriting cost constraint

The code construction for WEM with the maximal rewriting cost constraint is an immediate result due to the result for WEM with an average rewriting cost constraint.

The code construction, the rewriting operation and the decoding operation are exactly the same as Algorithm 5.3.1, Algorithm 5.3.2, and Algorithm 5.3.3, respectively.

The rewriting capacity is guaranteed by Lemma 27, the decoding and rewriting operation complexity is the same as polar codes, and the rewriting cost is obtained due to the strong converse result of rate distortion theory, i.e., Theorem 2 of [9]. Thus, we have:

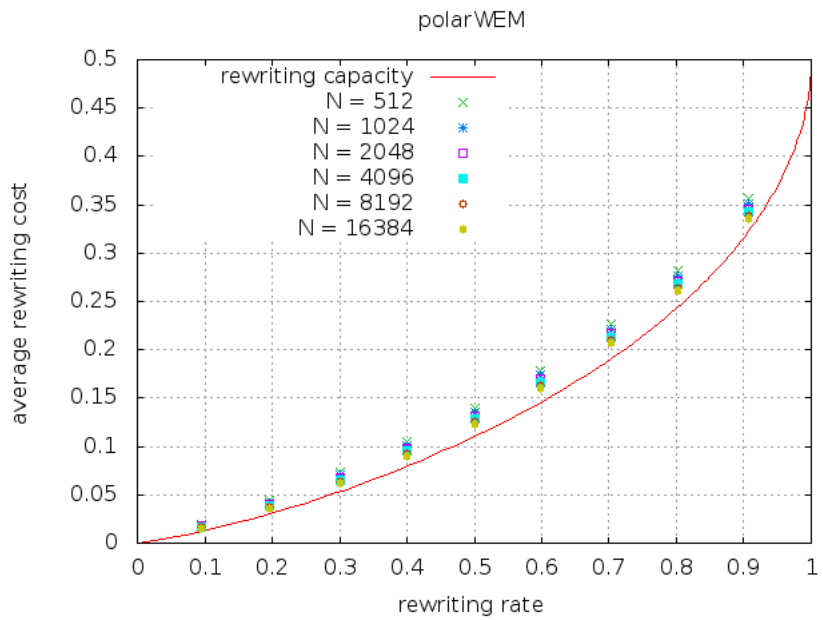


Figure 5.1: Experimental performance for polar WEM with an average cost constraint for polar code with various lengths, where the x-axis is the rewriting rate, the y-axis the average rewriting cost, and the theoretical points are those points (R, d) ($R \in \{0.2, 0.3, \cdots, 0.9\}$) satisfying $R = H(d)$.

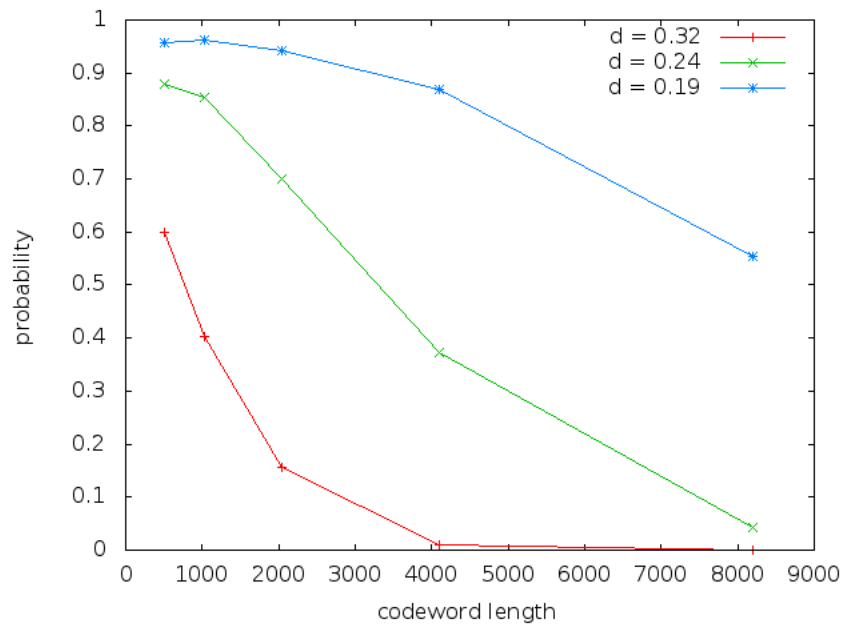


Figure 5.2: Experimental performance for polar WEM with a maximal cost constraint with $d = 0.32, 0.24$ and 0.19, respectively, where the x-axis is the codeword length, and y-axis is the empirical probability $Q(\varphi(y_0^{N-1}, x_0^{N-1}) \geq 1.1d)$.

Theorem 31. For a binary symmetric rewriting cost function $\varphi : \mathcal{X} \times \mathcal{X} \to \mathcal{R}_+$, fix a rewriting cost D, δ, and $0 < \beta < \frac{1}{2}$. For any rate $\mathcal{R} < \mathcal{R}^s(2, D)$, there exists a sequence of polar WEM codes of length N and rate $R \leq \mathcal{R}$, so that under the above rewriting operation and the induced probability distribution Q, the rewriting cost between a current codeword $\forall y_0^{N-1}$ and its updated codeword x_0^{N-1} satisfies $Q(\varphi(y_0^{N-1}, x_0^{N-1}) \geq D + \delta) < 2^{-N^\beta}$. The decoding and rewriting operation complexity of the codes is $O(N \log N)$.

We present our experimental results in Figure 5.2. The rewriting cost function, storage channel $\mathcal{P}$, and $\lambda^{(N)}$ are the same as those of the previous subsection. We let $\delta = 0.1d$, and $d = 0.32$, 0.24, and 0.19, respectively. The empirical probability $Q(\varphi(y_0^{N-1}, x_0^{N-1}) \geq 1.1d)$ is presented in Figure 5.2. As predicted by Theorem 31 it decreases (nearly exponen-

tially) as the length of codeword increases. However, even longer codewords are needed to make the probability to be truly negligible.

5.4 Polar Codes are Optimal for q-ary WEM, $q = 2^r$

In this section, we extend the previous binary scheme to q-ary WEM ($q = 2^r$), considering the length of polar codes, which is $N = 2^n$.

5.4.1 A code construction with an average rewriting cost constraint, $q = 2^r$

Our polar code is based on the recently proposed result for q-ary polar code where $q = 2^r$ [57], and we first present its background here.

5.4.1.1 Background of q-ary polar codes, $q = 2^r$ [57]

The storage alphabet is $\mathcal{X}$, $|\mathcal{X}| = q$, and for $x \in \mathcal{X}$, $(x_0, x_1, ..., x_{r-1})$ is its binary representation. Let $W : \mathcal{X} \to \mathcal{Y}$ be a symmetric discrete memoryless channel. $I(W)$ is $\sum\limits_{x\in\mathcal{X}} \sum\limits_{y\in\mathcal{Y}} \frac{1}{q} W(y|x) \log_2 \frac{W(y|x)}{\sum\limits_{x'\in\mathcal{X}} \frac{1}{q} W(y|x')}$.

Let $x_0^{N-1} = u_0^{N-1} G_N$, where the arithmetic operations are over $GF(q)$. y_0^{N-1} denotes the result of sending x_0^{N-1} through W. The sub-channel i, $W_N^{(i)}$, is defined by $W_N^{(i)}(y_0^{N-1}, u_0^{i-1}|u_i) \stackrel{def}{=} \frac{1}{q^{N-1}} \sum\limits_{u_{i+1}^{N-1}} \prod\limits_{i=0}^{N-1} W(y_i|(u_0^{N-1} G_N)_i)$.

Consider the following *bit channel*: Fix $k \in \{0, 1, \cdots, r\}$, let the channel input be $u \in \{0,1\}^{r-k}$ and output be $y \in \mathcal{Y}$, its transition probability is $W^{[r-k]}(y|u) = \frac{1}{2^k} \sum\limits_{x:x_k^{r-1}=u} W(y|x)$, where $x = (x_0, x_1, \cdots, x_{r-1}) \in \mathcal{X}$.

Futher consider the Bhattacharyya parameter for the bit channel: Define $Z(W_{\{x,x'\}}) = \sum\limits_{y\in\mathcal{Y}} \sqrt{W(y|x)W(y|x')}$, let $Z_v(W) = \frac{1}{2^r} \sum\limits_{x\in\mathcal{X}} Z(W_{\{x,x+v\}})$ for $v \in \mathcal{X}\backslash\{0\}$, and $Z_i(W) = \frac{1}{2^i} \sum\limits_{v\in\mathcal{X}_i} Z_v(W)$, where $i = 0, 1, \cdots, r-1$ and $\mathcal{X}_i = \{v \in \mathcal{X} : i = \arg \max\limits_{0\le j\le r-1} v_j \neq 0\}$ with the binary representation of v, $(v_0, v_1, \cdots, v_{r-1})$.

The channel polarization is as follows: $Z_i(W_N^{(j)})$ $\forall i \in \{0, 1, \cdots, r-1\}$ and $j \in \{0, 1, \cdots, N-1\}$ converges to $Z_{i,\infty} \in \{0, 1\}$, and with probability one $(Z_{0,\infty}, Z_{1,\infty}, \cdots,$

$Z_{r-1,\infty}$) takes one of the values $(Z_{0,\infty}=1,\cdots,Z_{k-1,\infty}=1,Z_{k,\infty}=0,\cdots,Z_{r-1,\infty}=0)$ $\forall k=0,1,\cdots,r-1$, i.e., Theorem 1.b of [57]; Let $k=0,1,\cdots,r-1$ and $n\in\mathbb{N}$, let $\mathcal{A}_{k,n}\subseteq\{0,1,\cdots,N-1\}$, let $\mathcal{R}_k(\epsilon)\overset{def}{=}(\prod_{i=0}^{k-1}D_1)\times(\prod_{i=k}^{r-1}D_0)$ with $D_0=[0,\epsilon)$ and $D_1=(1-\epsilon,1]$ for $\epsilon\in(0,1)$, and $j\in\mathcal{A}_{k,n}$ if $(Z_0(W_N^{(j)}),Z_1(W_N^{(j)}),\cdots,Z_{r-1}(W_N^{(j)}))\in\mathcal{R}_k(\epsilon)$. The channel polarizes in the sense that $\frac{\sum_{i\in\{0,1,\cdots,N-1\}}^{|\mathcal{A}_{k_i,n}|}}{rN}\to I(W)$ as $N\to\infty$.

The communication scheme is as follows: let us sort $j\in\{0,1,...,N-1\}$ by $\sum_{i=0}^{r-1}Z_i(W_N^{(j)})$. If this value is approximately k, j is assumed to be contained in the set $\mathcal{A}_{k,n}$, i.e., $j\in\mathcal{A}_{k,n}$ if $(Z_0(W_N^{(j)}),Z_1(W_N^{(j)}),....,Z_{r-1}(W_N^{(j)}))\in\mathcal{R}_k(\epsilon)$.

We transmit user data in encoding messages $u_0^{N-1}=(u_0,...,u_{N-1})$, i.e., if $j\in\mathcal{A}_{k,n}$, then the symbol $u_j\in\mathcal{X}$ is assembled by frozening the first k bits and user bits in the remaining bits.

Alternatively, for $u_0^{N-1}\in\mathcal{X}^N$, we also represent it by its binary form, that is $u_0^{N-1}=(u_{I(0,0)},\cdots,u_{I(0,r-1)},\cdots,u_{I(N-1,0)},\cdots,u_{I(N-1,r-1)})\in\{0,1\}^{rN}$, where $u_{I(i,0)},\cdots,u_{I(i,r-1)}$ is the binary representation of $u_i\in\mathcal{X}$, and $I(i,j)=i\times r+j$. $\forall i\in\mathcal{A}_{k_i,n}$, let the frozen bit set be determined by $F=\{I(i,j):i\in\{0,1,\cdots,N-1\},j\in\{0,1,\cdots,k_i-1\}\}\subseteq\{0,1,\cdots,rN-1\}$. Frozen bits for u_0^{N-1} are defined as $u_F\in\{0,1\}^{|F|}$ with the subvector $u_i:i\in F$.

Finally, the polar code with frozen set $F\in\subseteq\{0,1,...,rN-1\}$ and frozen set value $u_F\in\{0,1\}^{|F|}$ is $C_N(F,u_F)=\{x_0^{N-1}=u_0^{N-1}G_N:\forall u_{F^c}\in\{0,1\}^{|F^c|}\}$. The polar code ensemble for a frozen set is the collection of polar codes with all possible frozen set value, i.e, $C_N(F)=\{C_N(F,u_F):\forall u_F\in\{0,1\}^{|F|}\}$.

5.4.1.2 The code construction

We focus on the q-ary WEM code construction with a *symmetric rewriting cost function*, which satisfies $\forall x,y,z\in\{0,1,...,q-1\},\varphi(x,y)=\varphi(x+z,y+z)$, where $+$ is over GF(q).

Similar to the binary case, to construct codes for WEM with $\mathcal{R}^s(q, D)$, we utilize its related form $R^s(q, D)$ in Lemma 27 and its test channel $W(y|x)$.

The code construction is presented in Algorithm 5.4.1.

Algorithm 5.4.1 A code construction for $(N, M, q, D)_{ave}$ WEM

1: $\forall i \in \mathcal{A}_{k_i,n}$, $F = \{I(i, j) : i \in \{0, 1, ..., N-1\}, j \in \{0, 1, ..., k_i - 1\}\}$.
2: The $(N, M, q, D)_{ave}$ code is $\mathcal{C} = \{\mathcal{C}_i : \mathcal{C}_i = C_N(F, u_F(i))\}$, where $u_F(i)$ is the binary representation form of i for $i \in \{0, 1, ..., M-1\}$.

The WEM code rate is $\mathcal{R} = \frac{|F|}{rN}$, and the polar code rate is $R = \frac{|F^c|}{rN}$. Similarly, when R approaches $R^s(q, D)$, $\mathcal{R}$ approaches $\mathcal{R}^s(q, D)$ based on Lemma 27.

The rewriting operation is presented in Algorithm 5.4.2, where the SC encoding is a generalization of q-ary lossy source coding.

Algorithm 5.4.2 The rewriting operation $y_0^{N-1} = \mathbf{R}(x_0^{N-1}, i)$.

1: Let $v_0^{N-1} = x_0^{N-1} + g_0^{N-1}$, where g_0^{N-1} is a common-known and random vector.
2: SC encoding v_0^{N-1}, $\hat{u}_0^{N-1} = \hat{U}(v_0^{N-1}, u_F(i))$, that is for each k in the range 0 till $N-1$:

$$\hat{u}_j = \begin{cases} u_j & \text{if } j \in \mathcal{A}_{r,n}, \\ m & \text{with the posterior } P(m|\hat{u}_0^{j-1}, v_0^{N-1}), \end{cases}$$

where in the above $m = 0, 1, ..., q-1$ and if $j \in \mathcal{A}_{k,n}$ for $0 \leq k \leq r-1$, the first k bits of $\hat{u}_j$ are fixed, and let $\hat{y}_0^{N-1} = \hat{u}_0^{N-1} G_N$.
3: $y_0^{N-1} = \hat{y}_0^{N-1} - g_0^{N-1}$, and $-$ is over GF(q).

The correctness of the above rewriting function can be verified similarly to Lemma 28.

The decoding algorithm is still to retrieve bits in frozen set, and it is presented below:

Algorithm 5.4.3 The decoding operation $u_F(i) = \mathbf{D}(x_0^{N-1})$.

1: $y_0^{N-1} = x_0^{N-1} + g_0^{N-1}$.
2: $u_F(i) = (y_0^{N-1} G_N^{-1})_F$.

5.4.1.3 The average rewriting cost analysis

Similar to the analysis of equation (5.3), we obtain that $\bar{D} = \sum_{w_0^{N-1}} \pi(w_0^{N-1}) \sum_j \bar{D}_j(w_0^{N-1})$.

In the following, we first focus on $\bar{D}_j(w_0^{N-1})$. Note that $\hat{u}_0^{N-1} = \hat{U}(v_0^{N-1}, u_F(j))$ is random, i.e., the SC encoding function may result in different outputs for the same input. More precisely, in step i of the SC encoding process, $i \in \bigcup_{k=0}^{r-1} \mathcal{A}_{k,n}$, $\hat{u}_i = m$ with the posterior $P(m|\hat{u}_0^{i-1}, v_0^{N-1})$, where if $i \in \mathcal{A}_{k,n}$, the first k bits of $\hat{u}_i$ are fixed and known. This implies that the probability of picking a vector u_0^{N-1} with $\hat{\mathcal{A}}_{r,n}$ given v_0^{N-1} with $\mathcal{A}_{r,n}$ is equal to

$$\begin{cases} 0 & \text{if } \hat{\mathcal{A}}_{r,n} \neq \mathcal{A}_{r,n}, \\ \prod_{i \in \bigcup_{k=0}^{r-1} \mathcal{A}_{k,n}} P(u_i|u_0^{i-1}, v_0^{N-1}) & \text{otherwise,} \end{cases}$$

where in the second case $\forall i \in \mathcal{A}_{k,n}$, the first k bits of u_i are fixed.

Therefore, the average (in this case, over the probability of rewriting to data j and the randomness of the encoder) rewriting cost of updating w_0^{N-1} to a codeword representing j, $\bar{D}_j(w_0^{N-1})$, is

$$= \frac{1}{2^{|F|}} \sum_{u_{F^c}} \varphi(w_0^{N-1}, u_0^{N-1} G_N) \prod_{i \in \bigcup_{k=0}^{r-1} \mathcal{A}_{k,n}} P(u_i|u_0^{i-1}, w_0^{N-1}),$$

where $u_F = u_F(j)$, $u_{F^c} \in \{0,1\}^{|F^c|}$, and the summation over u_{F^c} takes care of $\mathcal{A}_{r,n}$ and the fact that $i \in \mathcal{A}_{k,n}$, the first k bits of u_i are fixed.

Thus, we obtain that $\bar{D}$

$$
\begin{aligned}
&= \sum_{w_0^{N-1}} \pi(w_0^{N-1}) \sum_j \bar{D}_j(w_0^{N-1}), \\
&= \sum_{w_0^{N-1}} \pi(w_0^{N-1}) \sum_{u_F(j)} \frac{1}{2^{|F|}} \sum_{u_F^c} \prod_{i \in \bigcup_{k=0}^{r-1} \mathcal{A}_{k,n}} P(u_i|u_0^{i-1}, w_0^{N-1}) \varphi(w_0^{N-1}, u_0^{N-1} G_N), \\
&< \sum_{w_0^{N-1}} \pi(w_0^{N-1}) \frac{1}{q^{|\mathcal{A}_{r,n}|}} \sum_{u_0^{N-1}} \prod_{i \in \bigcup_{k=0}^{r-1} \mathcal{A}_{k,n}} P(u_i|u_0^{i-1}, w_0^{N-1}) \varphi(w_0^{N-1}, u_0^{N-1} G_N). \quad (5.4)
\end{aligned}
$$

Let $Q_{U_0^{N-1}, W_0^{N-1}}$ denote the distribution defined by $Q_{W_0^{N-1}}(w_0^{N-1}) = \pi(w_0^{N-1})$, and $Q_{U_0^{N-1}|W_0^{N-1}}$ defined by

$$
Q(u_i|u_0^{i-1}, w_0^{N-1}) = \begin{cases} \frac{1}{q} & \text{if } i \in \mathcal{A}_{r,n}, \\ P(u_i|u_0^{i-1}, w_0^{N-1}) & \text{otherwise.} \end{cases}
$$

Then, inequation (5.4) is equivalent to $\bar{D} < E_Q(\varphi(w_0^{N-1}, u_0^{N-1} G_N))$, where $E_Q(\cdot)$ denotes the expectation with respect to the distribution $Q_{U_0^{N-1}, W_0^{N-1}}$. Similarly, let $E_P(\cdot)$ denote the expectation with respect to the distribution $P_{U_0^{N-1}, W_0^{N-1}}$.

The following three lemmas are already proved in [42] for $q = 2$ and in [41] for primary q, they extend trivially to $q = 2^r$, and we omit their proofs.

Lemma 32. $\sum_{w_0^{N-1}, u_0^{N-1}} |Q(w_0^{N-1}, u_0^{N-1}) - P(w_0^{N-1}, u_0^{N-1})| \leq \sum_{i \in \mathcal{A}_{r,n}} \sum_{u_i=0}^{q-1} E_P(|\frac{1}{q} - P(u_i|u_0^{i-1}, w_0^{N-1})|)$.

Lemma 33. Let F be chosen such that for $i \in \mathcal{A}_{r,n}$,

$$
\sum_{u_i=0}^{q-1} E_P(|\frac{1}{q} - P(u_i|u_0^{i-1}, w_0^{N-1})|) \leq \sigma_N.
$$

Then, the average rewriting cost is bounded by

$$\frac{1}{N}E_Q(\varphi(w_0^{N-1}, u_0^{N-1}G_N)) \leq \frac{1}{N}E_P(\varphi(w_0^{N-1}, u_0^{N-1}G_N)) + |\mathcal{A}_{r,n}|d_{max}\sigma_N,$$

where $d_{max} \stackrel{def}{=} \max_{x,y} \varphi(x, y)$.

Lemma 34. $E_P(\varphi(w_0^{N-1}, u_0^{N-1}G_N)) = ND$.

The following lemma, which is a modification of Lemma 5 [41], presents that it is sufficient to choose the set F as those bits with indexes $I(i, j)$ for which $i \in \{0, 1, ..., N-1\}, j \in \{0, 1, ..., k_i - 1\}$ with $i \in \mathcal{A}_{k_i,n}$.

Lemma 35. If $i \in \mathcal{A}_{r,n}$, that is $(Z_0(W_N^{(i)}), ..., Z_{r-1}(W_N^{(i)})) \in \mathcal{R}_r(\epsilon)$, and let $\epsilon_N \geq r\sqrt{\epsilon}$, then

$$\sum_{u_i=0}^{q-1} E_P(|\frac{1}{q} - P(u_i|u_0^{i-1}, w_0^{N-1})|) \leq \sqrt{2\epsilon_N}.$$

Proof. By Pinsker's inequality, for two distribution functions P and Q defined on $\mathcal{X}$, $||P - Q||_1 \leq \sqrt{2D(P||Q)}$, where $D(P||Q)$ is the Kullback-Leibler divergence between two distributions, that is $D(P||Q) = \sum_i \log_2(\frac{P(i)}{Q(i)})P(i)$, we obtain that:

$$\begin{aligned}
&\sum_{u_i=0}^{q-1} E_P|\tfrac{1}{q} - P(u_i|u_0^{i-1}, w_0^{N-1})| \\
&\stackrel{P(u_i)=\frac{1}{q}}{=} \sum_{w_0^{N-1},u_0^i} |P(u_i)P(u_0^{i-1}, w_0^{N-1}) - P(u_i|u_0^{i-1}, w_0^{N-1})P(u_0^{i-1}, w_0^{N-1})|, \\
&= \sum_{w_0^{N-1},u_0^i} |P(u_i)P(u_0^{i-1}, w_0^{N-1}) - P(u_0^i, w_0^{N-1})|, \\
&\leq \sqrt{2\sum_{w_0^{N-1},u_0^i} P(u_0^i, w_0^{N-1}) \log_2 \frac{P(u_0^i, w_0^{N-1})}{P(u_i)P(u_0^{i-1}, w_0^{N-1})}}, \\
&= \sqrt{2I(W_N^{(i)})}, \\
&\leq \sqrt{2\epsilon_N},
\end{aligned}$$

where the last inequality is based on the conclusion of Lemma 1 [57], that is if $(Z_0(W_N^{(i)}), \cdots, Z_{r-1}(W_N^{(i)})) \in \mathcal{R}_k(\epsilon)\ \forall k = 0, 1, \cdots, r$, then $|I(W_N^{(i)}) - (r-k)| \leq \gamma$ with $\gamma \geq \max(k\sqrt{\epsilon}, (2^{r-k}-1)\epsilon \log_2 e)$. □

Therefore, the performance of the above polar WEM code can be concluded as follows:

Theorem 36. For a q-ary ($q = 2^r$) symmetric rewriting cost function $\varphi : \mathcal{X} \times \mathcal{X} \to \mathcal{R}_+$, fix a rewriting cost D and $0 < \beta < \frac{1}{2}$. For any rate $\mathcal{R} < \mathcal{R}^s(q, D)$, there exists a sequence of polar WEM codes of length N and rate $R \leq \mathcal{R}$, so that under the above rewriting operation, $\bar{D}$ satisfies $\bar{D} \leq D + O(2^{-N^\beta})$. The decoding and rewriting operation complexity of the codes is $O(N \log N)$.

5.4.2 A code construction with a maximal rewriting cost constraint, $q = 2^r$

Similarly, the code construction, the rewriting operation and the decoding operation are exactly the same as Algorithm 5.4.1, Algorithm 5.4.2, and Algorithm 5.4.3, respectively. Next, we mainly focus on its performance.

Theorem 37. For a q-ary ($q = 2^r$) symmetric rewriting cost function $\varphi : \mathcal{X} \times \mathcal{X} \to \mathcal{R}_+$, fix a rewriting cost D, δ, and $0 < \beta < \frac{1}{2}$. For any rate $\mathcal{R} < \mathcal{R}^s(q, D)$, there exists a sequence of polar WEM codes of length N and rate $R \leq \mathcal{R}$, so that under the above rewriting operation and the induced probability distribution Q, the rewriting cost between a current codeword $\forall y_0^{N-1}$ and its updated codeword x_0^{N-1} satisfies $Q(\varphi(y_0^{N-1}, x_0^{N-1}) \geq D + \delta) < 2^{-N^\beta}$. The decoding and rewriting operation complexity of the codes is $O(N \log N)$.

Proof. We mainly focus on the rewriting cost analysis. The proof of this part is based on the *ϵ-strong typical sequence* [16] and Theorem 4 and 5 of [9]. We give the sketch as follows.

We recall ϵ-strong typical sequences $x_0^{N-1} \times y_0^{N-1} \in \mathcal{X}^N \times \mathcal{Y}^N$ with respect to $p(x, y)$ over $\mathcal{X} \times \mathcal{Y}$, and denote it by $A_\epsilon^{*(N)}(X, Y)$. We denote by $C(a, b|x_0^{N-1}, y_0^{N-1})$ the number

of occurrences of a, b in x_0^{N-1}, y_0^{N-1} with the same indexes, and require the following.

- First, $\forall a, b \in \mathcal{X} \times \mathcal{Y}$ with $p(a,b) > 0$, $|C(a,b|x_0^{N-1}, y_0^{N-1})/N - p(a,b)| < \epsilon$;
- Second, $\forall a, b \in \mathcal{X} \times \mathcal{Y}$ with $p(a,b) = 0$, $C(a,b|x_0^{N-1}, y_0^{N-1}) = 0$.

In our case $y_0^{N-1} = u_0^{N-1} G_N$. Due to the full rank of G_N, there is a one-to-one correspondence between u_0^{N-1} and y_0^{N-1}. We say that $u_0^{N-1}, x_0^{N-1} \in A_\epsilon^{*(N)}(U,X)$ if $x_0^{N-1}, u_0^{N-1} G_N \in A_\epsilon^{*(N)}(X,Y)$ with respect to $\frac{1}{q} W(y|x)$, where $W(y|x)$ is the test channel.

The first conclusion is that for N sufficiently large, $Q(A_\epsilon^{*(N)}(U,X)) > 1 - 2^{-N^\beta}$ for $\forall 0 < \beta < \frac{1}{2}$, $\epsilon > 0$, which is a generalization of Theorem 4 of [9], and where $Q(A_\epsilon^{*(N)}(U,X)) = Q(\forall a, b : |\frac{1}{N} C(a,b|u_0^{N-1} G_N, x_0^{N-1}) - \frac{1}{q} W(a|b)| \leq \epsilon)$. The sketch proof is based on Lemma 32 and Lemma 33 we obtain that

$$\sum_{u_0^{N-1}, x_0^{N-1} \in A_\epsilon^{*(N)}(U,X)} |Q(u_0^{N-1}, x_0^{N-1}) - P(u_0^{N-1}, x_0^{N-1})| \leq |\mathcal{A}_{r,n}| \sigma_N d_{max}.$$

Thus, we obtain

$$\begin{aligned} & |\sum_{u_0^{N-1}, x_0^{N-1}} Q(u_0^{N-1}, x_0^{N-1}) - P(u_0^{N-1}, x_0^{N-1})| \\ \leq & \sum_{u_0^{N-1}, x_0^{N-1}} |Q(u_0^{N-1}, x_0^{N-1}) - P(u_0^{N-1}, x_0^{N-1})|, \\ \leq & \ |\mathcal{A}_{r,n}| \sigma_N d_{max}, \end{aligned}$$

where in the above $u_0^{N-1}, x_0^{N-1} \in A_\epsilon^{*(N)}(U,X)$.

By lower bounding $P(A_\epsilon^{*(N)}(U,X)) = 1 - P(\exists a, b : |\frac{1}{N} C(a,b|u_0^{N-1} G_N, x_0^{N-1}) - \frac{1}{q} W(a|b)| \geq \epsilon) \geq 1 - 2q^2 e^{-2N\epsilon^2}$ based on Hoeffding's inequality, and $Q(A_\epsilon^{*(N)}(U,X)) \geq$

$P(A_\epsilon^{*(N)}(U,X)) - |\mathcal{A}_{r,n}|\sigma_N d_{max}$ (we just obtained), we obtain the desired result by setting $\sigma_N = \frac{2^{-N^\beta}}{2Nd_{max}}$.

The second conclusion is that let $y_0^{N-1} = \mathbf{R}(x_0^{N-1}, i)$, then $\forall\delta > 0$, $0 < \beta < \frac{1}{2}$ and N sufficiently large, $Q(\varphi(y_0^{N-1}, x_0^{N-1})/N \geq D + \delta) < 2^{-N\beta}$, which is a generalization of Theorem 5 of [9]. The outline is that $Q(\varphi(x_0^{N-1}, y_0^{N-1})/N \geq D + \delta)$

$$\begin{aligned}
&\leq \quad Q((\varphi(x_0^{N-1}, y_0^{N-1})/N \geq D + \delta)\bigcap x_0^{N-1}, y_0^{N-1} \in A_\epsilon^{*(N)}(X,Y)) + Q(x_0^{N-1}, y_0^{N-1} \notin A_\epsilon^{*(N)}(X,Y)),\\
&\leq \quad 2^{-N\beta},
\end{aligned}$$

where the last inequality is based on the conclusion just obtained $Q(x_0^{N-1}, y_0^{N-1} \notin A_\epsilon^{*(N)}(X,Y)) < 2^{-N\beta}$, and that when $x_0^{N-1}, y_0^{N-1} \in A_\epsilon^{*(N)}(X,Y)$, for ϵ sufficiently small and N sufficiently large, $\varphi(y_0^{N-1}, x_0^{N-1})/N \leq D + \delta$. $\square$

5.5 Conclusion

Code constructions for WEM using recently proposed polar codes have been presented. Future work focuses on exploring error-correcting codes for WEM.

6. CODING FOR NOISY WRITE-EFFICIENT MEMORIES

6.1 Introduction

Nonvolatile memories (such as flash memories and phase-change memories (PCM)) are becoming ubiquitous nowadays. Besides the well-known endurance [35] problem, another serious challenge is the data reliability issue, e.g., retention error [49] in NAND flash memories and resistance drift [34] in PCM.

Write-efficient memory (WEM) [2] is a coding model that can be used to solve the endurance problem for nonvolatile memories. In WEM, codewords are partitioned into several disjointed sets, where codewords of the same set represent the same data. A cost constraint has to be satisfied during the rewriting (namely, updating the data stored in WEM).

WEM is a natural model for PCM [44], and can also be applied to flash memory when data representation scheme such as rank modulation [46] is used. In WEM, there is a cost associated with changing the level of a cell. For nonvolatile memories such as PCM, this cost is important because cells age with programming and have the endurance problem. An optimal code [48] has been proposed to achieve the rewriting capacity of WEM. However, rewriting codes combined with error correction are still limited [37], especially for WEM [27].

In this paper, we propose a joint error correction and rewriting scheme for WEM. While previous results are mainly for Write-Once Memories [9], our work focuses on WEM. We propose a new coding model, noisy WEM, and provide a characterization for its capacity. We present an efficient coding scheme based on polar codes [5] for a special case of noisy WEM. The scheme is related to the coding schemes in [43], [9] and [37]. We also provide a lower bound to noisy WEM's capacity, and experimentally verify the code

construction's performance.

The rest of this part is organized as follows. In Section 5.2, we introduce noisy write-efficient memory model. In Section 5.3, we present characterization of noisy rewriting capacity of noisy WEM. In Section 5.4, we present an efficient code construction for a special case of binary noisy WEM, and verify its performance experimentally. We conclude this paper in Section 5.5.

6.2 Noisy Write-Efficient Memory Model

In this section, we first introduce terms and notations used throughout the paper, and then formally present the definitions of noisy WEM and related parameters.

6.2.1 Terms and notations

Let $\mathcal{X} = \{0, 1, \cdots, q-1\}$ be the alphabet of a symbol stored in a cell. (For example, for PCM, it denotes the q levels of a cell.) $\forall x, y \in \mathcal{X}$, let the rewriting cost of changing a cell's level from x to y be $\varphi(x, y)$. Given N cells and $x_0^{N-1}, y_0^{N-1} \in \mathcal{X}^N$, let $\varphi(x_0^{N-1}, y_0^{N-1}) = \frac{1}{N} \sum_{i=0}^{N-1} \varphi(x_i, y_i)$ be the rewriting cost of changing the N cell levels from x_0^{N-1} to y_0^{N-1}.

Let $M \in \mathbb{N}$ and $\mathcal{D} = \{0, 1, \cdots, M-1\}$. We use $\mathcal{D}$ to denote the M possible values of the data stored in the N cells. Let the decoding function be $\mathbf{D} : \mathcal{X}^N \to \mathcal{D}$, which maps the N cells' levels to the data they represent. Let the rewriting function be $\mathbf{R} : \mathcal{X}^N \times \mathcal{D} \to \mathcal{X}^N$, which changes the N cells' levels to represent the new input data. Naturally, we require $\mathbf{D}(\mathbf{R}(x_0^{N-1}, i)) = i$ for any $x_0^{N-1} \in \mathcal{X}^N$ and $i \in \mathcal{D}$.

Assume the sequence of data written to the storage medium is $\{M_1, \cdots, M_t\}$, where we assume M_i for $1 \leq i \leq t$ is uniformly distributed over $\mathcal{D}$, and the average rewriting cost is $\bar{D} \stackrel{def}{=} \lim_{t \to \infty} \frac{1}{t} \sum_{i=1}^{t} \varphi(x_0^{N-1}(i), \mathbf{D}(M_i, x_0^{N-1}(i)))$, where $x_0^{N-1}(i)$ is the current cell states before the ith update. By assuming the stationary distribution of cell levels $x_0^{N-1}(i)$

is $\pi(x_0^{N-1})$, $\bar{D} = \sum_{x_0^{N-1}} \pi(x_0^{N-1}) \sum_{j\in\mathcal{D}} \bar{D}_j(x_0^{N-1})$, where $\bar{D}_j(x_0^{N-1})$ is the average rewriting cost of updating cell level states x_0^{N-1} to a codeword representing $j \in \mathcal{D}$.

Let $\mathcal{P}(\mathcal{X} \times \mathcal{X})$ be the set of joint probability distributions over $\mathcal{X} \times \mathcal{X}$. For a pair of random variables $(S, X) \in (\mathcal{X}, \mathcal{X})$, let $P_{SX}, P_S, P_{X|S}$ denote the joint probability distribution, the marginal distribution, and the conditional probability distribution, respectively. $E(\cdot)$ denotes the expectation operator. If X is uniformly distributed over $\{0, 1, \cdots, q-1\}$, denote it by $X \sim U(q)$.

6.2.2 Noisy WEM with an average rewriting cost constraint

We formally present the definition of noisy WEM with an average rewriting cost constraint as follows:

Definition 38. An $(N, M, q, d)_{ave}$ noisy WEM code for the *storage channel* $\mathcal{P} = (\mathcal{X}, \mathcal{Y}, P_{Y|X}(y|x))$ consists of

- $\mathcal{D}$ and $\mathcal{C} = \bigcup_{i\in\mathcal{D}} \mathcal{C}_i$, where $\mathcal{C}_i \subseteq \mathcal{X}^N$ is the set of codewords representing data i. We require $\forall i \neq j$, $\mathcal{C}_i \bigcap \mathcal{C}_j = \emptyset$. (Here $P_{Y|X}(y|x)$ represents the transition probability of the noisy channel, which changes a cell's level from x to y.)
- A rewriting function $\mathbf{R}(s_0^{N-1}, i)$ with $d \in \mathbb{R}^+$ an upper bound to the average rewriting cost, i.e.,$\bar{D} \leq d$.
- A decoding function $\mathbf{D}(y_0^{N-1})$.

The noisy WEM model is illustrated in Figure 6.1. Here the N-dimensional vector $s_0^{N-1} \in \mathcal{X}^N$ is the current cell states, and the message M is the new information to write, which is independent of s_0^{N-1}. The rewriter uses both s_0^{N-1} and M to choose a new codeword $x_0^{N-1} \in \mathcal{X}^N$, which will be programmed as the N cells' new states, such that the average rewriting cost satisfies the predefined cost constraint. The codeword x_0^{N-1}

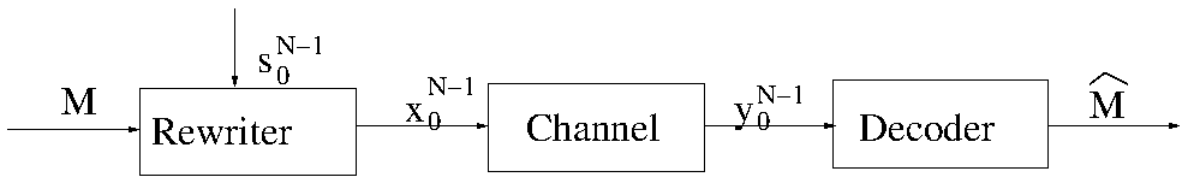


Figure 6.1: The noisy WEM model. M, s_0^{N-1}, x_0^{N-1}, y_0^{N-1} and $\hat{M}$ are , respectively, the message, the current cell states, rewritten codeword, the noisy channel's output, and the estimated message.

passes a noisy channel, and the noisy codeword $y_0^{N-1} \in \mathcal{X}^N$ is its output. The decoder can reliably decode y_0^{N-1} to recover the message M. The cell states s_0^{N-1} are drawn independently and identically from the probability distribution $P_S(s)$. The noisy channel is memoryless, and is characterized by the transition probabilities $P_{Y|X}(y|x)$.

Let $\lambda_i = Pr(\mathbf{D}(y_0^{N-1}) \neq i | x_0^{N-1} = \mathbf{R}(s_0^{N-1}, i))$ be the decoding error probability given data i. Let $\lambda^{(N)}$ be $\max_{i \in \mathcal{D}} \lambda_i$. Let $\mathcal{R} = \frac{\log M}{N}$ be the code rate, and we say $\mathcal{R}$ is achievable if there exists a $(N, 2^{N\mathcal{R}}, q, d)_{ave}$ code such that $\lambda^{(N)} \to 0$ as $N \to \infty$. The *noisy rewriting capacity* $\mathcal{C}(q, d)_{ave}$ is the supremum of all achievable rates.

The noisy WEM problem is: given the average rewriting cost constraint d, find the maximal rate $\mathcal{R}$ of reliable rewriting supported by the rewriter and the decoder despite the noisy channel. Let $\mathcal{P}(q, d)$ be $\{P_{SX} \in \mathcal{P}(\mathcal{X} \times \mathcal{X}) : P_S = P_X, E(\varphi(S, X)) \leq d\}$. When there is no noise, $\mathcal{C}(q, d)_{ave}$ is $\mathcal{R}(q, d)_{ave} = \max_{P_{SX} \in \mathcal{P}(q,d)} H(X|S)$ [2].

6.2.3 Noisy WEM with a maximal rewriting cost constraint

The WEM code in definition 8.4.1.1 puts a constraint on the average rewriting cost. We now define a code with a maximal rewriting cost constraint.

Definition 39. An (N, M, q, d) noisy WEM code for the *storage channel* $\mathcal{P} = (\mathcal{X}, \mathcal{Y}, P_{Y|X}(y|x))$ consists of

- $\mathcal{D}$ and $\mathcal{C} = \bigcup_{i\in\mathcal{D}} \mathcal{C}_i$, where $\mathcal{C}_i \subseteq \mathcal{X}^N$ is the set of codewords representing data i. We require $\forall i \neq j, \mathcal{C}_i \bigcap \mathcal{C}_j = \emptyset$.

- A rewriting function $\mathbf{R}(s_0^{N-1}, i)$ with $d \in \mathbb{R}^+$ an upper bound to the maximal rewriting cost, i.e., $\varphi(s_0^{N-1}, \mathbf{R}(s_0^{N-1}, i)) \leq d$ for any $s_0^{N-1} \in \mathcal{C}$ and $i \in \mathcal{D}$.

- A decoding function $\mathbf{D}(y_0^{N-1})$.

The rate $\mathcal{R}$ and noisy rewriting capacity $\mathcal{C}(q, d)$ can be defined similarly as before. When there is no noise, $\mathcal{C}(q, d)$ is $\mathcal{R}(q, d) = \mathcal{R}(q, d)_{ave}$ [2].

6.3 Characterizing Noisy Rewriting Capacity

In this section, we present the characterization of noisy rewriting capacity of $\mathcal{C}(q, d)$ and $\mathcal{C}(q, d)_{ave}$, respectively.

The characterization of $\mathcal{C}(q, d)$ is presented below. It is effectively the generalization of that of Gel'fand and Pinsker [32], which considers the problem without cost constraint. The direct part proof is based on random coding and typical sequences; the converse part is based on techniques of Fano's inequality [16] and Csiszár sum identity [18], and auxiliary random variables identification.

Lemma 40. For a given rewriting cost function $\varphi(\cdot, \cdot)$, $\mathcal{C}(q, d) = \max\limits_{\substack{P_{U|S}, x(u,s) \\ P_{SX} \in \mathcal{P}(q,d)}} \{I(Y; U) - I(U; S)\}$, where U is an auxiliary random variable, and $U \to (X, S) \to Y$ is a Markov chain.

Proof. We first present some background about strong typical-sequences. For more details, interested readers are referred to [20].

Let x_0^{N-1} be a sequence with N elements drawn from $\mathcal{X}$. Define the type of x_0^{N-1} by $\pi(x|x_0^{N-1}) = \frac{|\{i: x_i = x\}|}{N}$. The set $\mathcal{T}_\epsilon^N(X)$ is defined as:

$$\mathcal{T}_\epsilon^N(X) = \{x_0^{N-1} : |\pi(x|x_0^{N-1}) - P_X(x)| \leq \epsilon, \forall x\}.$$

That is, the set of sequences for which the empirical frequency is within ϵ of the probability $P_X(x)$ for every $x \in \mathcal{X}$.

Let (x_0^{N-1}, y_0^{N-1}) be a pair of sequences with elements drawn from $(\mathcal{X}, \mathcal{Y})$. Define their joint type: $\pi(x, y|x_0^{N-1}, y_0^{N-1}) = \frac{|\{i:(x_i,y_i)=(x,y)\}|}{N}$ for $(x, y) \in \mathcal{X} \times \mathcal{Y}$. We denote $\mathcal{T}_\epsilon^N(XY) = \{(x_0^{N-1}, y_0^{N-1}) : |\pi(x, y|x_0^{N-1}, y_0^{N-1}) - P_{XY}(x, y)| \leq \epsilon, \forall(x, y)\}$.

For $x_0^{N-1} \in \mathcal{T}_\epsilon^N(X)$ and $P_{Y|X}$, we define the conditional typical sequence $\mathcal{T}_{Y|X}^N(x_0^{N-1}) = \{y_0^{N-1} : (x_0^{N-1}, y_0^{N-1}) \in \mathcal{T}_\epsilon^N(XY)\}$.

i For a vector x_0^{N-1}, where x_i is chosen i.i.d. $\sim P_X$,

$$Pr(x_0^{N-1} \in \mathcal{T}_\epsilon^N(X)) \to 1 \text{ as } N \to \infty. \tag{6.1}$$

ii For vectors x_0^{N-1}, y_0^{N-1}, where (x_i, y_i) is chosen i.i.d. $\sim P_{XY}$,

$$Pr((x_0^{N-1}, y_0^{N-1}) \in \mathcal{T}_\epsilon^N(XY)) \to 1 \text{ as } N \to \infty. \tag{6.2}$$

iii For $x_0^{N-1} \in \mathcal{T}_\epsilon^N(X)$, and y_0^{N-1} is independently chosen according to P_Y, then

$$Pr((x_0^{N-1}, y_0^{N-1}) \in \mathcal{T}_{Y|X}^N(x_0^{N-1})) \in [2^{-N(I(X;Y)+\lambda)}, 2^{-N(I(X;Y)-\lambda)}], \tag{6.3}$$

for some $\lambda(\epsilon) > 0$ with $\lambda \to 0$ as $\epsilon \to 0$.

6.3.1 Proof of the direct part

The code construction We present details of the direct part by a random code construction. For a $P_{U|S}(u|s)$, let $P_U(u)$ be the marginal probability distribution of U under the joint probability distribution of $P_S(s)P_{U|S}(u|s)$. Independently generate a set of 2^{NQ} vectors of u_0^{N-1} from the typical sequence $\mathcal{T}_\epsilon^N(U)$, where Q is to be determined. Randomly

partition such 2^{NQ} vectors into 2^{NR} subsets each with size $2^{N(Q-R)}$:

$$\{\mathcal{A}_0, \mathcal{A}_1 ..., \mathcal{A}_{2^{NR}-1}\}.$$

Rewriting function and its analysis For the rewriting part, given a current state vector $s_0^{N-1} \in \bigcup_{i=0}^{2^{NR}-1} \mathcal{A}_i$ and data to rewrite $j \in \mathcal{D}$, randomly choose a vector u_0^{N-1} from $\mathcal{A}_j$ that is jointly typical with s_0^{N-1} with $P_{U,S}$. Select a vector x_0^{N-1} based on the function $x(u, s)$. Due to the definition of strong typical sequence, we know that $\varphi(s_0^{N-1}, x_0^{N-1} = \mathbf{R}(s_0^{N-1}, i)) \leq ND + \epsilon$. Therefore, the successful rewriting depends on whether such u_0^{N-1} exist or not.

Next, we analyze the condition under which such u_0^{N-1} exists almost for sure. The probability that we do not have a vector from $\mathcal{A}_j$ that is jointly typical with s_0^{N-1} is $P(\mathcal{A}_j \bigcap \mathcal{T}_{U|S}^N(s_0^{N-1}) = \emptyset)$

$$\begin{aligned}
&\leq P(\mathcal{A}_j \bigcap \mathcal{T}_{U|S}^N(s_0^{N-1}) = \emptyset | s_0^{N-1} \notin \mathcal{T}_\epsilon^N(S)) + P(\mathcal{A}_j \bigcap \mathcal{T}_{U|S}^N(s_0^{N-1}) = \emptyset | s_0^{N-1} \in \mathcal{T}_\epsilon^N(S)), \\
&\leq (1 - P(u_0^{N-1} \in \mathcal{A}_j \bigcap \mathcal{T}_{U|S}^N(s_0^{N-1}) | s_0^{N-1} \in \mathcal{T}_\epsilon^N(S)))^{|\mathcal{A}_j|}, \\
&\leq (1 - 2^{-N(I(S;U)+\lambda)})^{2^{N(Q-R)}}, \\
&\leq \exp\{2^{-(Q-R-I(S;U)+\lambda)}\},
\end{aligned}$$

where the first inequation is based on the conclusions (6.1) and (6.3).

To ensure the success of rewriting, there have to be vectors in $\mathcal{A}_j$, which are jointly typical with s_0^{N-1}, and this implies that $Q - R > I(U; S)$.

Decoding function and its analysis For the decoding part, given a received vector y_0^{N-1}, decode it is $j \in \mathcal{D}$ if there exists some $u_0^{N-1} \in \mathcal{A}_j$ jointly typical with y_0^{N-1} for a unique j.

Let us consider the probability of decoding error for rewriting data j, $P_e(j)$. There are

two cases: one is there is no vector in $\mathcal{A}_j$ jointly typical with y_0^{N-1}, and the other is there are vectors in $\mathcal{A}_i$ for $i \neq j$ jointly typical with y_0^{N-1}. Based on union bound, $P_e(j)$

$$
\begin{aligned}
&\leq P((u_0^{N-1}, y_0^{N-1}) \in \mathcal{T}_\epsilon^N(UY) | u_0^{N-1} \in \mathcal{A}_i, i \neq j) + P((u_0^{N-1}, y_0^{N-1}) \notin \mathcal{T}_\epsilon^N(UY) | u_0^{N-1} \in \mathcal{A}_j), \\
&\leq \sum_{\substack{u_0^{N-1} \\ i \neq j}} P((u_0^{N-1}, y_0^{N-1}) \in \mathcal{T}_\epsilon^N(UY) | u_0^{N-1} \in \mathcal{A}_i), \\
&\leq \sum_{\substack{u_0^{N-1} \in \mathcal{A}_i \\ i \neq j}} 2^{-N(I(U;Y)-\lambda)}, \\
&\leq 2^{-N(I(U;Y)-Q-\lambda)},
\end{aligned}
$$

where the first inequation is based on the conclusion (6.2), and the second inequation is based on (6.3).

Therefore, the decoding error probability $P_e(j)$ and further λ^N approach 0 if $Q < I(U;Y)$.

6.3.2 Proof of the converse part

The converse part is as follows: $N\mathcal{R}$

$$
\begin{aligned}
&= H(M), && (6.4) \\
&= H(M|y_0^N) + I(M; y_0^N), && (6.5) \\
&\leq N\epsilon_N + I(M; y_0^{N-1}), && (6.6) \\
&\leq N\epsilon_N + \sum_{i=0}^{N-1} I(M; y_i | y_0^{i-1}), && (6.7) \\
&\leq \sum_{i=0}^{N-1} [I(y_i; s_{i+1}^{N-1} | M, y_0^{i-1}) - I(s_i; y_0^{i-1} | M, s_{i+1}^N)] + N\epsilon_N + \sum_{i=0}^{N-1} I(M; y_i | y_0^{i-1}), && (6.8) \\
&= \sum_{i=0}^{N-1} [I(M, s_{i+1}^N; y_i | y_0^{i-1}) - I(s_i; y_0^{i-1} | M, s_{i+1}^N)] + N\epsilon_N, && (6.9)
\end{aligned}
$$

where

(6.4) follows from the assumption that M is uniformly distributed among $\mathcal{D}$;

(6.5) follows from the definition of mutual information;

(6.6) follows from Fano's inequality [16];

(6.7) follows from chain rule for mutual information[16], and so does equation (6.9);

(6.8) follows from Csiszár sum identity [18], which is

$$\sum_{i=0}^{N-1} I(A_i; B_{i+1}^{N-1}|A_0^{i-1}) = \sum_{i=0}^{N-1} I(B_i; A_0^{i-1}|B_{i+1}^{N-1}), \tag{6.10}$$

where in inequation (6.6) $A_i = y_i$, $B_i = s_i$, and by conditioning on M.

Define $u_i = (M, s_{i+1}^{N-1}, y_0^{i-1})$ and we know that $u_i \to (x_i, s_i) \to y_i$, we continue equation (6.7): $I(M, s_{i+1}^{N-1}; y_i|y_0^{i-1}) - I(s_i; y_0^{i-1}|M, s_{i+1}^{N-1})$

$$\begin{aligned} &= H(y_i|y_0^{i-1}) - H(y_i|u_i) - H(s_i|M, s_{i+1}^{N-1}) + H(s_i|u_i), \\ &\leq H(y_i) - H(y_i|u_i) - H(s_i) + H(s_i|u_i), \\ &= I(y_i; u_i) - I(s_i; u_i), \end{aligned} \tag{6.11}$$

where inequation (6.11) is based on the fact that s_i is independent of M and s_{i+1}^{N-1}.

We now show that it suffices to maximize over $P_{U|S}(u|s)$ and functions $x(u, s)$. Fix $P_{U|S}(u|s)$, and note that

$$P_{Y|U}(y|u) = \sum_{x,s} P_{S|U}(s|u) P_{X|U,S}(x|u, s) P_{Y|X,S}(y|x, s)$$

is linear in in $P_{X|U,S}(x|u, s)$. Since $P_{U|S}(u|s)$ is fixed, the maximization over the noisy WEM formula is only over $I(U, Y)$, which is convex in $P_{Y|U}(y|u)$ ($P_U(u)$ is fixed) and

hence in $P_{X|U,S}(x|u,s)$. This implies that the maximum is achieved at an extreme point of the set of $P_{X|U,S}(x|u,s)$, that is, using one of the deterministic mappings $x(u,s)$. This completes the proof of the converse. $\square$

The next lemma presents us the characterization of $\mathcal{C}_{ave}(q,d)$, which is the same as $\mathcal{C}(q,d)$. We omit its proof as any code for (N,M,q,d) is a code for $(N,M,q,d)_{ave}$, therefore $\mathcal{R}_{ave} \geq \mathcal{C}(q,d)$. The converse part is the same as previous one.

Lemma 41. For a given rewriting cost function $\varphi(\cdot,\cdot)$, $\mathcal{C}_{ave}(q,d) = \mathcal{C}(q,d) = \max\limits_{\substack{P_{U|S}, x(u,s) \\ P_{SX} \in \mathcal{P}(q,d)}} \{I(Y;U) - I(U;S)\}$, where $U \to (X,S) \to Y$ is a Markov chain.

6.4 A Code Construction for Binary Degraded and Symmetric Noisy WEM

Let $\mathcal{P}^s(q,d) = \{P_{SX} \in \mathcal{P}(\mathcal{X} \times \mathcal{X}) : P_S = P_X, S \sim U(q), E(\varphi(S,X)) \leq d\}$ be the set of joint probabilities with uniform marginal distributions. Let *symmetric rewriting capacity* be defined as $\mathcal{R}^s(q,d) = \max\limits_{P_{SX} \in \mathcal{P}^s(q,d)} H(X|S)$. Let W_{SX} be $\arg \max\limits_{P_{XS} \in \mathcal{P}^s(q,d)} H(X|S)$. We call $\mathcal{W} = (\mathcal{X}, \mathcal{X}, W_{X|S})$ the *WEM channel*.

We say $\mathbb{Q} = (\mathcal{X}, \mathcal{Z}, Q_{Z|X})$ is *degraded* with respect to $\mathbb{W} = (\mathcal{X}, \mathcal{Y}, W_{Y|X})$ (which we denote by $\mathbb{Q} \preccurlyeq \mathbb{W}$) if there exists a channel $\mathbb{P} = (\mathcal{Y}, \mathcal{Z}, P_{Y|Z})$ such that for all $z \in \mathcal{Z}$ and $x \in \mathcal{X}$, we have $Q_{Z|X}(z|x) = \sum\limits_{y \in \mathcal{Y}} W_{Y|X}(y|x) \cdot P_{Z|Y}(z|y)$.

In this section, we consider symmetric rewriting capacity, and present a code construction for noisy WEM when the WEM channel $\mathcal{W}$ is degraded with respect to the symmetric storage channel $\mathcal{P}$. We focus on binary cells, namely, $|\mathcal{X}| = 2$. We call such WEM a *binary degraded and symmetric noisy WEM*. (Note that when the flipping rates meet $W_{X|S} > P_{Y|X}$, the degradation condition is naturally satisfied.)

6.4.1 A nested polar code construction for binary degraded and symmetric noisy WEM with an average rewriting cost constraint

6.4.1.1 A brief introduction to binary polar codes [5]

Let $W : \mathcal{X} \to \mathcal{Y}$ be a binary-input discrete memoryless (DMC) channel. Let $G_2^{\otimes n}$ be n-th Kronecker product of $\left(\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right)$. Let $Z(W) = \sum_{y \in \mathcal{Y}} \sqrt{W_{Y|X}(y|0) W_{Y|X}(y|1)}$.

The polar code $C_N(F, u_F)$ is $\{x_0^{N-1} = u_0^{N-1} G_2^{\otimes n} : u_{F^c} \in \{0,1\}^{|F^c|}\}$, where $\forall F \subseteq \{0, 1, \cdots, N-1\}$, u_F is the subvector $u_i : i \in F$, and $u_F \in \{0,1\}^{|F|}$.

Let $W_N^{(i)} : \{0,1\} \to \mathcal{Y}^N \times \{0,1\}^i$ be a sub-channel with $W_N^{(i)}(y_0^{N-1}, u_0^{i-1}|u_i) \stackrel{def}{=} \frac{1}{2^{N-1}} \sum_{u_{i+1}^{N-1}} \prod_{i=0}^{N-1} W_{Y|X}(y_i|(u_0^{N-1} G_2^{\otimes n})_i)$, and $(u_0^{N-1} G_2^{\otimes n})_i$ denotes the i-th element of $u_0^{N-1} G_2^{\otimes n}$.

6.4.1.2 The code construction

We focus on the code construction with *symmetric rewriting cost function*, which satisfies $\forall x, y, z \in \{0,1\}, \varphi(x, y) = \varphi(x + z, y + z)$, where $+$ is the XOR operation over GF(2). (Many cost functions, such as the Hamming-distance based cost function satisfies this constraint.)

Algorithm 6.4.1 A code construction for binary degraded and symmetric noisy WEM and storage channel $\mathcal{P}$

1: Let $C_N(F_{\mathcal{P}}, u_{F_{\mathcal{P}}})$ be a polar code [5] designed for the storage channel $\mathcal{P}$, where $F_{\mathcal{P}} = \{i \in \{0, 1, \cdots, N-1\} : Z(\mathcal{P}_N^{(i)}) \geq 2^{-N^\beta}\}$ and $u_{F_{\mathcal{P}}}$ is set to 0.

2: Let $C_N(F_{\mathcal{W}}, u_{F_{\mathcal{W}}})$ be a polar code designed for the WEM channel $\mathcal{W}$, where $F_{\mathcal{W}} = \{i \in \{0, 1, \cdots, N-1\} : Z(\mathcal{W}_N^{(i)}) \geq 2^{-N^\beta}\}$ and $F_{\mathcal{P}} \subseteq F_{\mathcal{W}}$.

3: The $(N, M, 2, d)_{ave}$ code is $\mathcal{C} = C_N(F_{\mathcal{P}}, u_{F_{\mathcal{P}}}) = \{C_N(F_{\mathcal{W}}/F_{\mathcal{P}}, u_{F_{\mathcal{W}/F_{\mathcal{P}}}}(i))\}$, where $u_{F_{\mathcal{W}/F_{\mathcal{P}}}}(i)$ is the binary representation form of $i \in \{0, \cdots, M-1\}$.

The code construction is presented in Algorithm 6.4.1, where we use nested polar codes (i.e., the polar code for channel coding [5] and the polar code for WEM [48]) to de-

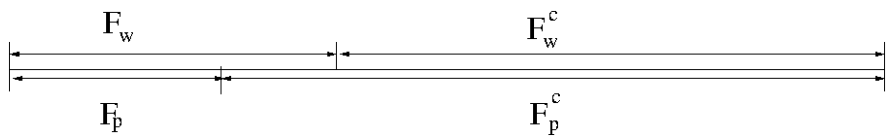


Figure 6.2: Illustration of relationship between $F_{\mathcal{W}}$ and $F_{\mathcal{P}}$. The line represents the indexes, and $F_{\mathcal{W}}$ and $F_{\mathcal{P}}$ are the frozen set for the WEM channel $\mathcal{W}$ and the storage channel $\mathcal{P}$, respectively.

sign noisy WEM codes. The fact that $F_{\mathcal{P}} \subseteq F_{\mathcal{W}}$ follows from [43, lemma 4.7]. Figure 6.2 presents us a pictorial presentation of $F_{\mathcal{W}}$ and $F_{\mathcal{P}}$.

The rewriting function is presented in Algorithm 6.4.2. It is very similar to that of [48] except for how to set u_F.

Algorithm 6.4.2 The rewriting operation $y_0^{N-1} = \mathbf{R}(x_0^{N-1}, i)$.

1: Let $v_0^{N-1} = x_0^{N-1} + g_0^{N-1}$, where g_0^{N-1} is a common and uniformly distributed message, and $+$ is over GF(2).

2: Apply SC (Successive Cancellation) encoding [43] to v_0^{N-1}, and this results in a vector $u_0^{N-1} = \hat{U}(v_0^{N-1}, u_{F_{\mathcal{W}/F_{\mathcal{P}}}}(i))$, that is, $u_j =$

$$\begin{cases} (u_{F_{\mathcal{W}/F_{\mathcal{P}}}}(i))_j & \text{if } j \in F_{\mathcal{W}/F_{\mathcal{P}}} \\ 0 & \text{if } j \in F_{\mathcal{P}} \\ m & \text{with probability } \frac{\mathcal{W}(u_0^{j-1}, v_0^{j-1}|m)}{\sum_{m'} \mathcal{W}(u_0^{j-1}, v_0^{N-1}|m')}, \end{cases}$$

and $\hat{y}_0^{N-1} = u_0^{N-1} G_2^{\otimes n}$.

3: $y_0^{N-1} = \hat{y}_0^{N-1} + g_0^{N-1}$.

The decoding function is presented in Algorithm 6.4.3, where we use the SC (Successive Cancellation) decoding [5] to assure $\lambda^{(N)} \to 0$ as $N \to 0$.

Algorithm 6.4.3 The decoding operation $u_{F_{\mathcal{W}/F_{\mathcal{P}}}}(i) = \mathbf{D}(x_0^{N-1})$.

1: $\hat{y}_0^{N-1} = x_0^{N-1} + g_0^{N-1}$.
2: Apply SC decoding to $\hat{y}_0^{N-1}$, and this results in y_0^{N-1}, i.e., $y_j =$

$$\begin{cases} 0 & \text{if } j \in F_{\mathcal{P}} \\ \arg_m \max \mathcal{P}_N^{(j)}(y_0^{j-1}, \hat{y}_0^{N-1}|m) & \text{else} \end{cases}$$

3: $u_{F_{\mathcal{W}/F_{\mathcal{P}}}}(i) = (y_0^{N-1}(G_2^{\otimes n})^{-1})_{u_{F_{\mathcal{W}/F_{\mathcal{P}}}}}$.

6.4.1.3 Theoretical performance analysis

Theorem 42. For a binary degraded symmetric noisy WEM, fix d, $\forall \mathcal{R} \leq \mathcal{R}^s(2,d)_{ave} - H(P_{Y|X})$ and any $0 < \beta < \frac{1}{2}$, there exists a sequence of nested polar codes of length N with rates $R \leq \mathcal{R}$ so that under the above rewriting and decoding operations, $\bar{D} \leq d + O(2^{-N^\beta})$, $\lambda^{(N)} \leq O(2^{-N^\beta})$, and the rewriting as well as the decoding operation complexity is $O(N \log N)$.

Proof. Let ϵ and $0 < \beta < \frac{1}{2}$ be some constants. $F_{\mathcal{P}}$ and $F_{\mathcal{W}}$ are $F_{\mathcal{W}} = \{i : Z(\mathcal{W}_N^{(i)}) \geq 2^{-N^\beta}\}$, $F_{\mathcal{P}} = \{i : Z(\mathcal{P}_N^{(i)}) \geq 2^{-N^\beta}\}$. Based on [43, lemma 2.6] $\lim_{n\to\infty} Pr(Z(\mathcal{W}_N^{(i)}) \geq 2^{-N^\beta}) = 1 - I(\mathcal{W}) = \mathcal{R}^s(2,d)$, thus $\frac{|F_{\mathcal{W}}|}{N} \leq \mathcal{R}^s(2,d) + \epsilon$ for sufficiently large N. Similarly, $\frac{|F_{\mathcal{P}}|}{N} \geq H(P_{Y|X}) - \epsilon$ for sufficiently large N.

As mentioned, $F_{\mathcal{P}} \subseteq F_{\mathcal{W}}$, thus $R = \frac{|F_{\mathcal{W}}| - |F_{\mathcal{P}}|}{N} \leq \mathcal{R}^s(2,d) - H(P_{Y|X})$.

Since the rewriting function is similar to that of [48], the rewriting cost is guaranteed by [48, theorem 8], i.e., $\bar{D} \leq d + O(2^{-N^\beta})$.

The error probability $\lambda^{(N)}$ is guaranteed by [5, theorem 4], that is $\lambda^{(N)} \leq \sum_{i \in F_{\mathcal{P}}^c} Z(\mathcal{P}_N^{(i)}) \leq O(2^{-N^\beta})$. □

Covering radius of polar codes In the following, we present *covering radius* to theoretically upper bound the maximal rewriting cost when the cost metric is the Hamming

distance between old and new cell levels.

Let the polar code ensemble be $C_N(F) = \{C_N(F, u_F) : u_F \in \{0,1\}^{|F|}\}$. Let the covering radius of $C_N(F)$ (which we denote $c_H(C_N(F))$) be $\max_{i,j} \min_{\substack{x_0^{N-1} \in C_N(F,u_F(i)) \\ y_0^{N-1} \in C_N(F,u_F(j))}} d_H(x_0^{N-1}, y_0^{N-1})$, where $d_H(x_0^{N-1}, y_0^{N-1})$ is the Hamming distance between x_0^{N-1} and y_0^{N-1}.

Lemma 43. $c_H(C_N(F)) = \min_{l \in F^c} 2^{wt(l)}$, where $wt(l)$ is the number of ones in (i.e., Hamming weight of) the binary representation of l.

Proof. $c_H(C_N(F))$

$$
\begin{aligned}
&= \max_{i,j} \min_{\substack{x_0^{N-1} \in C_N(F,u_F(i)) \\ y_0^{N-1} \in C_N(F,u_F(j))}} d_H(x_0^{N-1}, y_0^{N-1}), \\
&= \max_{i,j} \min_{\substack{x_0^{N-1} \in C_N(F,u_F(i)) \\ y_0^{N-1} \in C_N(F,u_F(j))}} wt(x_0^{N-1} - y_0^{N-1}), \\
&= \max_{i,j} \min_{z_0^{N-1} \in C_N(F,u_F(i)+u_F(j))} wt(z_0^{N-1}), \\
&= \max_{k} \min_{l \in F^c} 2^{wt(l)}, \qquad (6.12) \\
&= \min_{l \in F^c} 2^{wt(l)},
\end{aligned}
$$

where eq. (6.12) is based on [43, lemma 6.2], i.e., the minimal distance of polar code $C_N(F, u_F)$ is $\min_{l \in F^c} 2^{wt(l)}$. □

The above results can be generalized to the following polar codes $C_{N,M}(F) = \bigcup_{i=0}^{M-1} C_N(F, u_F(i))$, where $\{u_F(i)\}$ forms a group under binary operations in GF(2).

6.4.1.4 Experimental performance

The experimental performance is presented in Figure 6.3, where the rewriting cost function is the Hamming distance between old and new cell states, the upper bound of $\mathcal{C}(2, d)$ is $H(d)$ [2], the storage channel $\mathcal{P}$ is the binary symmetric channel with flipping rate $p = 0.001$, and the lower bound is $H(d) - H(p)$. $\lambda^{(N)}$ is set to be around 10^{-5}.

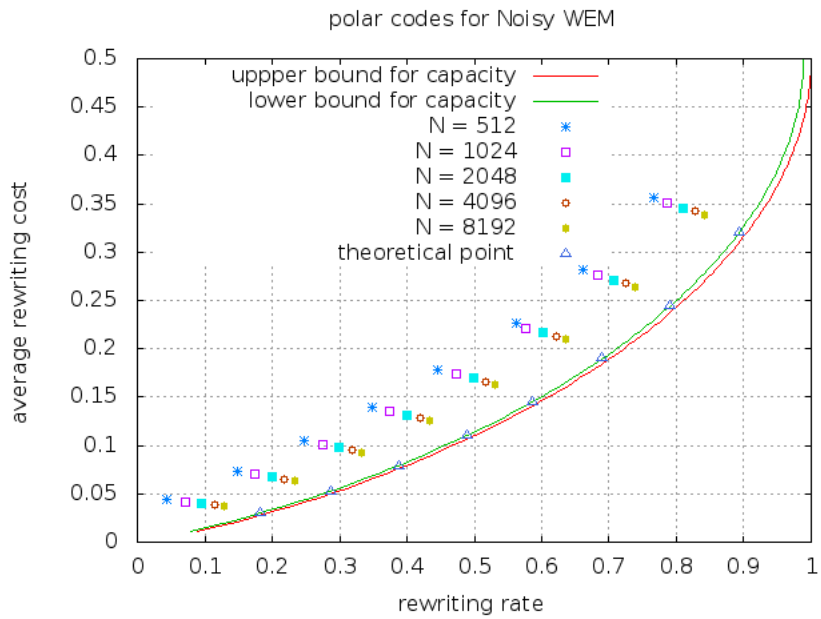


Figure 6.3: Experimental performance for noisy WEM with an average cost constraint for polar code with various lengths, where the x-axis is the rewriting rate, the y-axis the average rewriting cost, and the theoretical points are those points (R, d) ($R \in \{0.2, 0.3, \cdots, 0.9\}$) satisfying $R = H(d) - H(0.001)$.

We can see that the rates and the average rewriting costs approach those of points of $H(d) - H(0.001)$ as the length of codeword increases. Longer codewords are needed for further approaching the lower bound.

6.4.2 A nested polar code construction for binary degraded and symmetric noisy WEM with a maximal rewriting cost constraint

The code construction in Algorithm 6.4.1, the rewriting function in Algorithm 6.4.2 and the decoding function in Algorithm 6.4.3 can be applied to noisy WEM codes with a maximal rewriting cost constraint as well.

Similar to the analysis of Theorem 42, we obtain the following result for the theoretical

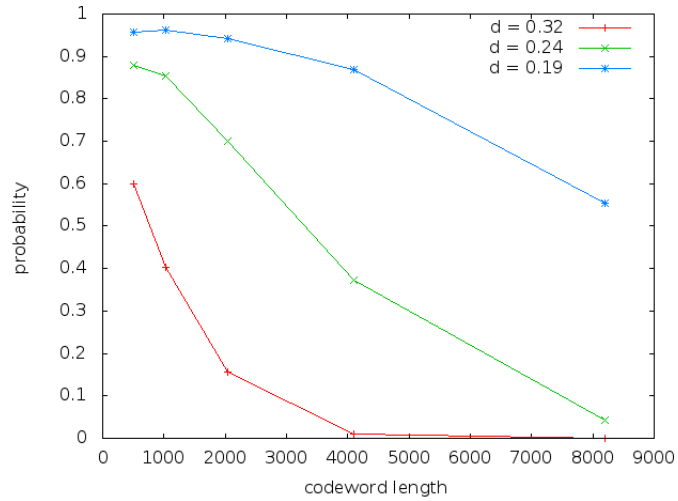


Figure 6.4: Experimental performance for noisy WEM with a maximal cost constraint with $d = 0.32, 0.24$ and 0.19, respectively, where the x-axis is the codeword length, and y-axis is the empirical probability $Q(\varphi(y_0^{N-1}, x_0^{N-1}) \geq 1.1d)$.

performance of the proposed code construction.

Theorem 44. For a binary degraded symmetric noisy WEM, fix d, δ, $\forall \mathcal{R} \leq \mathcal{R}^s(2, d) - H(P_{Y|X})$ and any $0 < \beta < \frac{1}{2}$, there exists a sequence of nested polar codes of length N with rates $R \leq \mathcal{R}$, so that under the above rewriting operation and decoding operation, the probability that the rewriting cost between a current codeword $\forall y_0^{N-1}$ and its updated codeword x_0^{N-1} larger than $d + \delta$ is bounded by $Q(\varphi(y_0^{N-1}, x_0^{N-1}) \geq d + \delta) < 2^{-N^\beta}$, $\lambda^{(N)} \leq O(2^{-N^\beta})$, and the decoding and rewriting operations' complexity of the code is $O(N \log N)$.

We present our experimental results in Figure 6.4. The rewriting cost function, storage channel $\mathcal{P}$, and $\lambda^{(N)}$ are the same as those of the previous subsection. We let $\delta = 0.1d$, and $d = 0.32$, 0.24, and 0.19, respectively. The empirical probability $Q(\varphi(y_0^{N-1}, x_0^{N-1}) \geq 1.1d)$ is presented in Figure 6.4. As predicted by Theorem 44 it decreases (nearly exponen-

tially) as the length of codeword increases. However, even longer codewords are needed to make the probability to be truly negligible.

6.5 Concluding Remarks

In this work, we analyse the capacity for noisy WEM, and present a code construction for binary degraded and symmatric noisy WEM. The code construction is both theoretically analyzed and experimentally verified. We are interested in extending the code construction to q-ary cells, and to more general settings regarding channel degradation. Those remain as our future research directions.

7. WOM CODES AGAINST INTER-CELL INTERFERENCE IN NAND MEMORIES

7.1 Introduction

Flash memories, especially the NAND flash memories, have become more and more important during recent years. Their outstanding features – such as high density and so on – facilitated their widespread use. Two combined technological efforts, aggressive scaling and the introduction of multi-level-cell (MLC), are taken to attain a further higher density.

However, the high density does not come without a cost. In flash memories, electrons are stored in floating gates (cells). The number of electrons stored in a floating gate determines the cell level, therefore single-level-cell (SLC) and MLC are distinguished by the number of levels in per cell. In order to get a measurable cell level, the threshold voltage, V_{th}, must be applied to the floating gate. Generally, the more electrons are stored in per cell, the higher V_{th} is. As the ever-growing application of MLC NAND flash memories, the precise controlling of V_{th} is becoming important. However, in reality V_{th} is disturbed by three well-known major parasitic effects, of which cell-to-cell interference is dominant [45]. In NAND Flash memories, the V_{th} shift of one floating-gate transistor can disturb the threshold voltages of its neighboring floating-gate transistors through parasitic capacitance – coupling effect, which is referred to as cell-to-cell interference.

Besides interference, another well-known challenge of NAND flash memories is its limited life time, measured by the number of erasures experienced by a floating gate. In NAND flash memories, cells are organized as the hierarchy of arrays, blocks and pages, where one array is partitioned into many blocks, and each block contains a certain number of pages. Data is programmed and fetched in the unit of page, where the page size ranges from 512-B to 8- kB user data in current design practice. However, all memory cells within

the same block must be erased at the same time, which is time and energy consuming. Therefore, one research topic of flash memories is to treat flash memories as write-once memories (WOM) [61], and we reuse them as many times as possible. WOM consists of a number of "write-once" cells where each cell initially is in a '0' state and can be changed to a '1' state irreversibly. SLC flash memories follow this model. WOM can be generalized to q states per cell [25], and MLC flash memories follow this model. Since Rivest and Shamir presented that we can write a 2-bit variable twice via a WOM-code using 3 cells, lots of research has been done, e.g., [71], [72] and so on.

Currently, there is some research about interference, but most focuses on circuit [56] or architecture level [22]. There has been some work [7] on constrained coding for combatting inter-cell interference. Generally speaking, constrained code [54] imposes a constraint on the sequence that are allowed to be recorded or transmitted. Shannon cover is of great help in constrained coding, which uses the minimal number of states to represent the constrained system. In this work, we study the joint coding for both rewriting data and combatting inter-cell interference. To the best of our knowledge, this is the first work that addresses this joint coding problem.

In this work, we explore WOM codes against cell-to-cell interference. We firstly define the model in Section 7.2. We then introduce one generalized WOM codes, Delta-WOM, in Section 7.3, which plays an important role in our analysis of code performance. We present various bounds of the codes in Section 7.4. Finally we present code constructions in Section 7.5, one of which is based on the efficient t successive writes Diamond-WOM codes, which is proved to approach its rewriting capacity.

7.2 Cell-to-cell Interference Model

Figure 7.1 presents the basic structure of a block and the model of cell-to-cell interference used in this paper. Each word-line forms a page, by adding voltages on certain word-

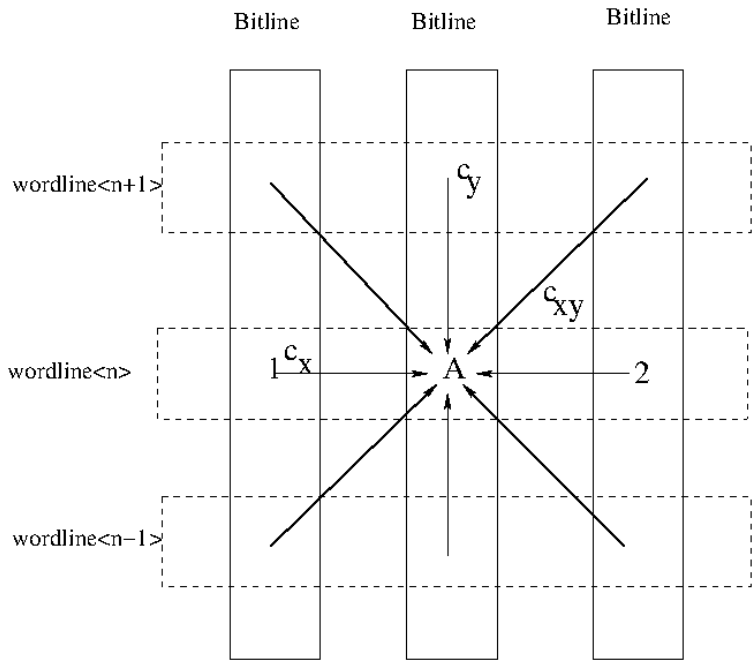


Figure 7.1: Cell-to-cell interference

line and bit-line we can program a specific cell. The victim cell, A, can be disturbed by its eight neighbors. According to [45], the threshold-voltage change of the victim cell due to interference can be estimated as $\sum_k \Delta V_t^{(k)} \frac{C^{(k)}}{C_{total}}$, where $\Delta V_t^{(k)}$ is the threshold-voltage change of its kth neighbor, $C^{(k)}$ is the parasitic capacitance between the kth interfering cell and the victim cell, and C_{total} is the total capacitance of victim cell. In this paper, we only consider the one dimension inter-cell interference, or more specifically the noise from the same word-line. Because the cell level is proportional to its threshold-voltage, we can use the change of cell level to estimate interference. In order to have inter-cell interference in a controllable manner, we require that during each round of programming the sum of cell level changes of victim's neighbors be not greater than the change of the victim's for some predetermined threshold value, or equivalently for the q-ary flash cell

$$\Delta q_1 + \Delta q_2 - \Delta q_A \leq D, \tag{7.1}$$

where D is the given threshold value, and $\Delta q_1, \Delta q_2$ and Δq_A are the cell level change of cell $1, 2$ and A. We let $q-1 \leq D \leq 2(q-1)-1$ such that all cells can be programmed to level $q-1$ at the same time. The intuition behind this model is quite straightforward: if the cell level gap between the victim cell and its neighbors is too large, then the neighbors will 'drag' the victim cell level upwards. Similar model has appeared in [7], where specific programming orders are considered and constrained codes is used. Our model is a general one, that is we do not consider specific programming order, and our research focus is that we construct WOM codes to overcome interference, that is our codes satisfy (7.1).

7.3 The Rewriting Capacity of Delta-WOM

In this section, we introduce Delta-WOM. A t-write WOM codes can be written as $< v_1, v_2, ..., v_t > /n$, that is the a variable of cardinality v_i for the ith write, its ith rate is $R_i = \frac{\log_2 v_i}{n}$ and total rates of t writes is $\sum_{i=1}^{t} R_i$. The maximal total rates, meaning the maximal number of information bits stored in one cell during successive rewrites, is also called rewriting capacity. The achievable values of the vector $(R_1, R_2, ..., R_t)$ form the rewriting capacity region.

The Delta-WOM is such a WOM code that cells at lower states can be programmed directly to at most D higher states, or equivalently whose transition model is $0 \to 0, 0 \to 1, ..., 0 \to D, 1 \to 1, 1 \to 2, ..., 1 \to D+1, ...$ (See (a) of Fig.7.2, where D equals 2). We notice that when D is $q-1$, this is actually the multi-level-cell WOM [25] (Any cells of lower states can be changed irreversibly to all higher states), which can also be verified at the end of this section.

To obtain the rewriting capacity of Delta-WOM, we use connectivity matrix, $\mathcal{A} = (\alpha_{\mathbf{ij}})_{\mathbf{q}\times\mathbf{q}}$, where $\alpha_{ij} = 1$ if there is an edge between vertex i and vertex j, and $\alpha_{ij} = 0$

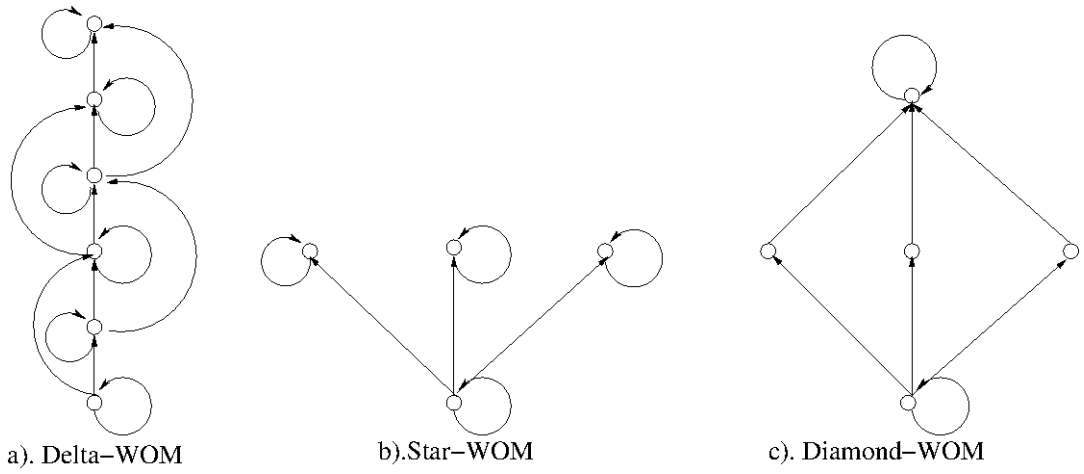


Figure 7.2: Delta-WOM, Star-WOM and Diamond-WOM

otherwise. For instance, the connectivity matrix of (a) in Fig. 7.2 is

$$\mathcal{A} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Generally, $\mathcal{A}$ can be written as $\mathcal{A} = \mathcal{B} - \mathcal{C}$, where $\mathcal{B}$ is the upper triangular matrix with all ones in the upper, and $\mathcal{C}$ is the following matrix

$$\begin{pmatrix} \mathbf{0} & \mathcal{B}_{(1+D)\times(1+D)} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}.$$

Fu and Yeung in [2] presented the rewriting capacity formula of deterministic WEM – of which WOM is a special case (see [2] for more details) – during t successive rewrites,

$C(t) = \log(\mathbf{a_0} \cdot \mathcal{A}^{\mathbf{t-1}} \cdot \ \mathbf{1_q}^T)$, where $\mathbf{a_0}$ is the first row of $\mathcal{A}$, and $(\cdot)^T$ is the transpose of an vector or matrix. Now we use this to obtain the rewriting capacity for q-ary Delta-WOM during t successive rewrites, $C(q, D, t)$, and for the detailed proof please see the appendix.

Theorem 45. $C(q, D, t) =$

$$\log_2 \sum_{i=0}^{p} \binom{t-1}{i} (-1)^i \cdot \{ \binom{t-1+q-i(1+D)}{t-1} - \binom{t-1+q-(i+1)(1+D)}{t-1} \},$$

where $p = \lfloor \frac{q}{1+D} \rfloor$, and $\binom{m}{n}$ is 0 if $m < n$.

Clearly, if $D = q-1$, $C(q, D, t) = \log_2 \binom{t-1+q}{t-1}$, which is exactly the rewriting capacity of multi-level-cell WOM [61].

7.4 Bound Rewriting Capacity for Rewriting Codes

In this section we provide various bounds of the rewriting capacity for codes against inter-cell interference.

7.4.1 Lower bounds based on delta-WOM

One strategy to eliminate interference is to use the Delta-WOM codes, that is having some constraint on the cell level changed of each write, and if necessary combining with the strategy of dividing cells into non-interfering groups.

7.4.1.1 One lower bound based on grouping cells and constraint on the cell level change

Given n $q-$ary WOM cells, $\vec{c} = (c_0 c_1 ..., c_{n-1})$, the constraint parameter D as in (7.1) and requirement of t rewrites, we divide $\vec{c}$ into three groups: $G_i = \{c_j | j \bmod 3 = i\}$, where $i = 0, 1, 2$ and $j = 0, 1, ..., n-1$, every time we only program one group cells with the constraint, D, on cell level change of every rewrite. In this way, we can make sure there is no interference during each rewriting. Clearly, according to Theorem 45, we have

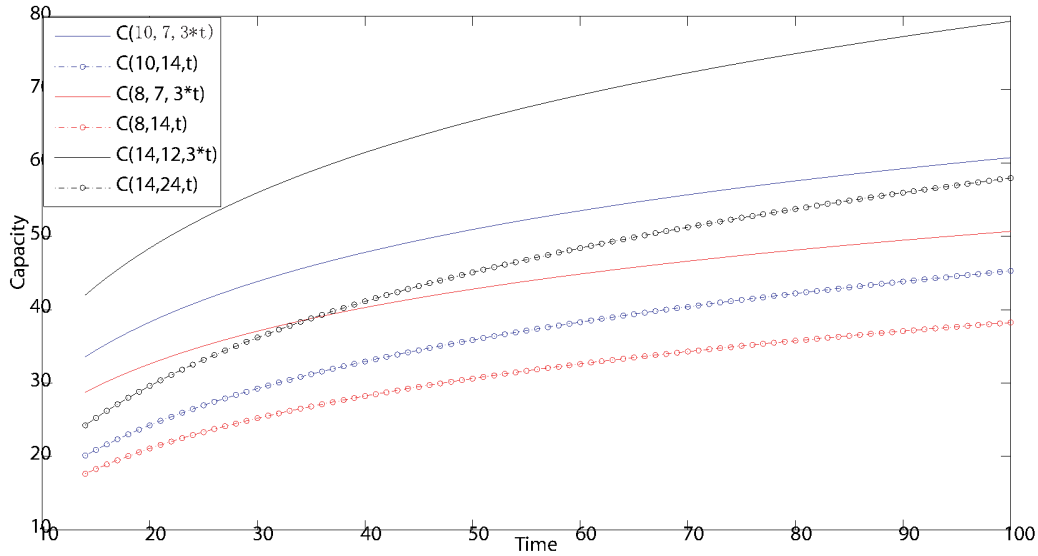


Figure 7.3: Comparison of lower bounds: The solid lines are for the lower bounds with grouping as well as constraint on the cell level change, and the lines with circles are for the lower bounds with the constraint on the cell level change only.

a lower bound for the rewriting capacity, cap

$$cap \geq C(q, D, t/3). \tag{7.2}$$

Figure 7.3 compares the two lower bounds we obtained above, from which we can see that the bound (7.2) is tighter.

7.4.2 Lower bound based on constrained codes

We now bound the rewriting capacity based on the strategy of constrained codes. It is clear that constrained codes are of great help if the codes are used to write data only once [7]. In this paper, we extend constrained codes from one write to multiple writes, which is

a highly nontrivial extension. For simplicity, we take the example of two writes.

The scheme is as follows: For the first write we require all the cell levels are in the range $[0, q_1)$ for some cell level q_1, and the second one is in $[0, q)$. Denote the rewriting capacity of q-ary constrained codes with the respect to the constraint (7.1) by $\mathcal{C}(q, D)$. It is clear that the lower bound of the sum rate of the scheme is $\max_{q_1}\{\mathcal{C}(q_1, D)+\mathcal{C}(q-q_1, D)\}$, however, according to [1], we can show that it is exactly the rewriting capacity of this scheme.

We now briefly present how to obtain the above result. Let S, X and Y be some finite sets of states, the input alphabet and the output alphabet. We suppose that for any stuck cells state $\vec{s} \in S$, meaning that those stuck cell states will not be lowered in future writes, there is a deterministic function $\phi_{\vec{s}}$ defined on some subset $X_{\vec{s}} \subset X$, such that $\phi_{\vec{s}}(x) \in Y$ for any $x \in X_{\vec{s}}$. That is $\phi_{\vec{s}}(x)$ transfers the input $\vec{x}$ to the output according to the state $\vec{s}$. Let $Y_{\vec{s}} = \{y = \phi_{\vec{s}}(x) | x \in X_{\vec{s}}, \vec{s} \in S\}$. The encoding f and decoding g are as follows, $f(u, \vec{s}) : \{1, 2, ..., M\} \times S \to X$, and $g(y) : Y \to \{1, 2, ..., M\}$. We consider the maximum number M_ϕ of messages that can be transmitted with respect to the stuck cells. The following lemma bounds the M_ϕ and the corresponding maximum transmission rate $R_\phi = \log_2 M_\phi/n$.

Lemma [1] 46. Let $N_\phi = \min_{\vec{s}\in S} |Y_s|$. M_ϕ is bounded as $\frac{N_\phi}{\ln(N_\phi|S|)} \leq M_\phi \leq N_\phi$, where $|\cdot|$ means the cardinality of a set.

For our WOM codes, it is easy to know the maximal transmission rate of the first write R_1 is $\mathcal{C}(q_1, D)$. For the second write, we know

$$X = Y = E^n, E = \{0, 1, 2, ..., q-1\}$$

$$S = \{\vec{s} \in E^n | \vec{s} \text{ satisfies the constraint (7.1)}\}$$

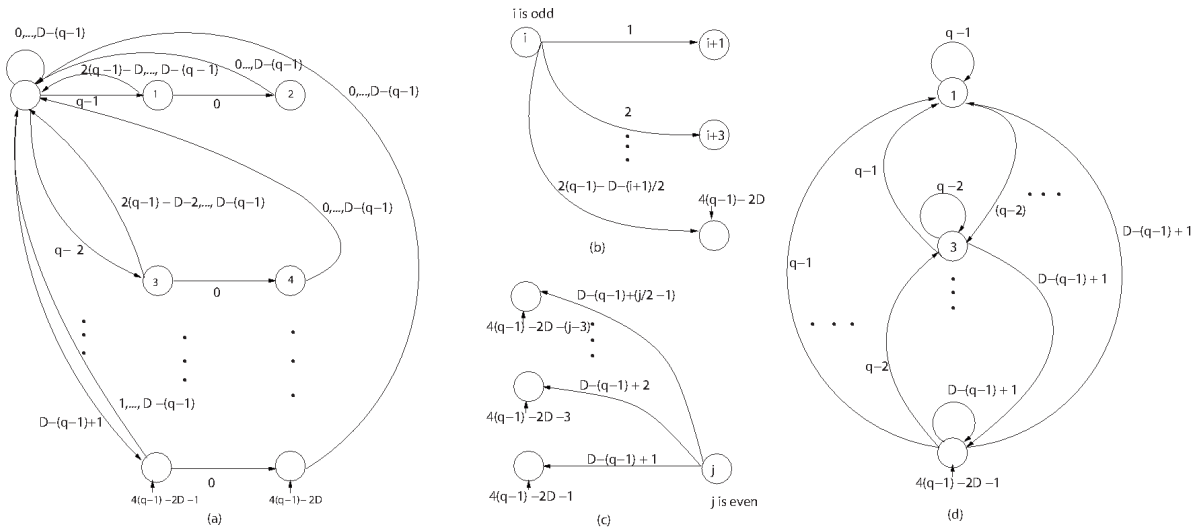


Figure 7.4: Shannon cover of q-ary constrained codes with constraint (7.1). (a) Graph representation of the constrained q-ary codes. There are 1 + 4(q-1) -2D vertices. (b) Additional edges between odd vertex i and even vertices $i+1$, $i+3$,..., and $4(q-1)-2D$. (c) Additional edges between even vertex j and odd vertices $4(q-1)-2D-1$,..., $4(q-1)-2D-(j-3)$. (d) Additional edges between any odd vertex and all other odd vertices (including itself).

$$X_{\vec{s}} = Y_{\vec{s}} = \{\vec{y} \in E^n | \vec{y} - \vec{s} \text{ satisfies (7.1) and } \vec{y} \geq \vec{s}\}$$

$$\phi_{\vec{s}}(\vec{x}) = \vec{x}, \text{ for } \vec{x} \in X_{\vec{s}}, \vec{s} \in S$$

We know that $N_\phi = \min_{\vec{s} \in S} |Y_s| =$ number of $(q-q_1)$-ary constrained codewords satisfying the constraint (7.1), denoting as M, and $|S| = 2^{n\mathcal{C}(q_1,D)}$. According to the above lemma, we obtain $M/\ln(M2^{n\mathcal{C}(q_1,D)}) \leq M_\phi \leq M$, and R_2 approaches $\mathcal{C}(q-q_1,D)$ as n is infinite. Thus the rewriting capacity of our WOM codes is $\max_{q_1}\{\mathcal{C}(q_1,D)+\mathcal{C}(q-q_1,D)\}$.

Figure (7.4) presents the Shannon cover of q-ary codes with respect to the constraint (7.1), which has the same structure as [7]. We can easily obtain its capacity $C(q,D)$

according to the knowledge of constrained codes [54]. Obviously, this method can be extended to general cases as long as we reuse the codes less than $q-1$ times.

Naturally, there exist other techniques to bound the rewriting capacity using both Delta-WOM and constrained codes: for example, we can divide cells into three groups G_0, G_1 and G_2, and we let G_1, and G_0 together with G_2 be programmed separately. For G_1, we use Delta-WOM to bound its rewriting sum rate; for G_0 and G_2, we use constrained codes to bound the rewriting sum rate.

7.4.3 An explicit upper bound scheme

In this part, we obtain an upper bound of the rewrite capacity. The idea is as follows: Instead of dividing cells into non-interfering groups, we form a giant non-overlapped 'cell' by grouping three consecutive cells, e.g., cell $0, 1$ and 2 form one giant 'cell', and another 'cell' consists of cell $3, 4$ and 5. It is easy to obtain the connectivity matrix, $\mathcal{A}_{\mathbf{3}} = (\alpha_{\mathbf{ij}})_{\mathbf{q^3 \times q^3}}$, where $\alpha_{ij} = 1$ if we can program the giant 'cell' from 'level' i to 'level' j with respect to the constraint (7.1) and WOM codes. For example, for the binary WOM codes with the constraint $\Delta q_0 + \Delta q_2 - \Delta q_1 \leq 1$, we obtain the connectivity matrix of giant 'cell' as follows:

$$\mathcal{A}_{\mathbf{3}} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

where the first row shows the ways of programming the giant 'cell' from 'level' 000 to 8 possible 'levels' (including itself). Seven of them are possible except for the transferring from 000 to 101, which violates the constraint. Similar analysis can be applied to other rows. Since all transformation ways of consecutive three cells in the WOM codes against interference are contained in this matrix, but the codes formed by those giant

'cells' might still generate interference due to the noise among boundaries of the giant 'cells', clearly it is the upper bound of WOM Codes against interference. We compute rewriting capacity of the giant 'cell' based on the formula demonstrated by Fu and Yeung, $C(t) = \log(\mathbf{a_0} \cdot \mathcal{A}_{\mathbf{3}}^{\mathbf{t-1}} \cdot \mathbf{1_q}^T)$. Thus we can upper bound it by $\frac{\log(\mathbf{a_0} \cdot \mathcal{A}_{\mathbf{3}}^{\mathbf{t-1}} \cdot \mathbf{1_q}^T)}{3}$. Further more, we can obtain its rewriting capacity as the size of giant 'cell' grows larger and larger.

7.5 Code Constructions

In this section, we present code constructions. According to the two strategies bounding the rewriting capacity, there are two methods to construct WOM codes against interference.

7.5.1 Code construction based on diamond-WOM

One method to construct the codes is naturally using Delta-WOM codes, that is letting non-interfering cell be programmed at most D or $D/2$ more levels. For the former codes, it is the multi-level-cell WOM codes, however, no existing WOM codes reach its rewriting capacity, although Wu [71] and others presented code constructions to reach the rewriting capacity of binary WOM of twice rewrites; for the later (having the constraint of $D/2$), there are no known WOM codes approaching its rewriting capacity either. In this subsection, we present an efficient code construction of Diamond-WOM, which can be proved to approach its rewriting capacity, and construct WOM codes against interference based on it.

Diamond-WOM is such a WOM codes that only cells at the '0' state can be changed directly to other states except the highest state $(q-1)$ irreversibly, all the non-zero states can only be changed to (remain at) the high states $(q-1)$, or equivalently whose transition model is $0 \rightarrow 0, 0 \rightarrow 1, ..., 0 \rightarrow q-2, 1 \rightarrow (q-1), ..., (q-1) \rightarrow (q-1)$ (see (c) of Figure (7.2)). Actually, the Diamond-WOM can be regarded as *position modulation code* [72] with fixed weight, where a certain number of binary WOM cells are grouped

together as a symbol, and at the beginning of each rewrite all the non-zero symbols are erased by setting them to all-one value. Similarly, for Diamond-WOM every time before each rewrite, we erasure all the non-zero cells by programming them to the highest state $(q-1)$, which can be thought of as a *dummy state*. Thus we get 'clean' cells every time before each rewrite. Using these remaining cells, we encode the message by the number of initial cells, and the positions of non-zero (except the 'dummy state' $(q-1)$) cells. Also, we can see the connection between Diamond-WOM and Star-WOM (see (b) of Figure (7.2)): the q-ary Diamond-WOM is equivalent to the $(q-1)$-ary Star-WOM in that the non-zero Star-WOM cells are stuck cells and they remain at their original states, while the non-zero Diamond-WOM cells also are stuck cells but they have to stuck at state $(q-1)$. This can also be verified from their rewriting capacity regions and rewrite capacities. Fu and Vinck [26] formulated q-ary Star-WOM's rewriting capacity region, which is

$$C_t = \{(R_1...R_t)|R_1 \leq h(\vec{P}^{(1)})...R_t \leq \prod_{i=1}^{t-1} {P_0}^{(i)} h(\vec{P^{(t)}})\},$$

where $\vec{P^{(i)}} = (P_0^{(i)}, P_1^{(i)}, ..., P_{q-1}^{(i)})(i = 1, 2, 3, ...)$ is a probability vector, and $P_j^{(i)}$ implies that at the ith rewrite at most $P_j^{(i)}$ fraction of available cells can be programmed from initial state to state j. Its rewriting capacity is $\log_2[1 + (q-1)t]$.

The following two lemmas present the rewriting capacity region of Diamond-WOM and its rewriting capacity, respectively.

Lamma 47. The rewriting capacity region of q-ary Diamond-WOM during t-successive rewrites is

$$C_t = \{(R_1...R_t)|R_1 \leq h(\vec{p^{(1)}})...R_t \leq \prod_{i=1}^{t-1} p_0^{(i)} h(\vec{p^{(t)}})\}, \tag{7.3}$$

where $\vec{p^{(i)}} = (p_0^{(i)}, p_1^{(i)}, ..., p_{q-1}^{(i)})(i = 1, 2, 3, ...)$ still is a probability vector, however, according to the definition of Diamond-WOM, $p_{q-1}^{i} = 0$ for $i \in \{1, 2, ..., t\}$.

Proof. Let $S_1, S_2, ..., S_t$ be the random variables form a Markov chain, denoted as $S_1 \Rightarrow S_2 \Rightarrow ... \Rightarrow S_t$. According to [25] the rewriting capacity region is $R_1 \leq h(S_1), R_2 \leq h(S_2|S_1), ..., R_t \leq h(S_t|S_{(t-1)})$. Let

$$p_j^{(1)} = Pr\{S_1 = j\}, j = 0, 1, ..., q-1$$

$$p_j^{(i)} = Pr\{S_i = j|S_{i-1} = 0\}, j = 0...q-1, i = 2, ..., t$$

$$\vec{p^i} = (p_0^{(i)}, p_1^{(i)}...p_{q-1}^{(i)}), i = 1, 2, ..., t$$

We note that $Pr\{S_1 = q-1\} = 0$ and

$$Pr\{S_{i+1} = q-1|S_i = j\} = 1, i = 1...t-1, j = 1...q-1.$$

Using the above known facts and the same proof of the rewriting capacity region for Star-WOM [25], we can obtain the result. □

Lemma 48. The rewriting capacity of successive t rewrites of q-ary Diamond-WOM is $\log_2(1 + (q-2)t)$.

We briefly present its proof: the connectivity matrix of Diamond-WOM is

$$\mathcal{A}_{\mathcal{D}} = \begin{pmatrix} \mathbf{1_{q-2}} & 0 \\ \mathbf{0_{(q-2)\times(q-2)}} & \mathbf{1_{q-2}} \end{pmatrix}.$$

Then it is easy to obtain the result according to $C(t) = \log(\mathbf{a_0} \cdot \mathcal{A}_{\mathcal{D}}{}^{\mathbf{t-1}} \cdot \ \mathbf{1_q}^T)$ [2].

We summarize the optimal parameters in Table 7.1, and code construction of Diamond-WOM in the following, and for more details interested readers please refer to the appendix.

We now illustrate how to use t successive writes Diamond-WOM codes to construct codes against interference. Taking the example of having constraint $D/2$ on the cell level

Algorithm 7.5.1 Code construction for Diamond-WOM.

1: Encode: For the data to write y ($y \in [0, 2^{n(\prod_{j=0}^{i-1} p_0^{*(j)})h(p_0^{*(i)}, \frac{1-p_0^{*(i)}}{q-1}, \ldots, \frac{1-p_0^{*(i)}}{q-1})} - 1]$), first flip all non-zero state cells to state $q-1$, and then flip cells according to encoding function $f_i^{-1}(y)$.

2: Decode: With the cell state vector $\vec{c}_i$, the value y is $y = f_i(\vec{c}_i)$.

Table 7.1: Optimal parameters for Diamond-WOM

i	$p_0^{(i)}$	$p_1^{(i)}$	...	p_{q-2}^{i}	$p_{q-1}^{(i)}$
1	$\frac{1}{1+\frac{q-2}{2^{RC_1}}e^{-(q-3)}}$	$\frac{1-p_0^{(1)}}{q-2}$	...	$\frac{1-p_0^{(1)}}{q-2}$	0
2	$\frac{1}{1+\frac{q-2}{2^{RC_2}}e^{-(q-3)}}$	$\frac{1-p_0^{(2)}}{q-2}$	...	$\frac{1-p_0^{(2)}}{q-2}$	0
...	...	...	...	∴	0
t	$\frac{1}{q-1}$	$\frac{1}{q-1}$	$\frac{1}{q-1}$	$\frac{1}{q-1}$	0

change, we first use the Diamond-WOM codes in the cell level range $[0, D/2-1]$ for t_1 times, and then reuse the remaining cell in the level range $[D/2, q-1]$ for t_2 times. It is easy to know that using this method, the rewriting capacity is $\log_2[1+(D/2-2)t_1] + \log_2[1+(q-D/2-2)t_2]$.

7.5.2 Code construction based on constrained codes

Another code construction is based on constrained codes. We take the example of two writes WOM codes. Since the rewriting capacity is $\max_{q_1}\{C(q_1, D) + C(q-q_1, D)\}$, we can take the following simple approach. The first write is an q_1-ary constrained codes with respect to (7.1). We briefly discuss how to generate encoder and decoder according to the Shannon cover graph, and see [54] for more details. Firstly, for the rate R, we find the input data size k and the output data size of encoder l such that $R = k/l \leq C(q_1, D)$; Secondly, we multiply the Shannon cover graph such that every node has q_1^l output edges;

Finally, we assign input data to edges. Encoding and decoding are done by traversing the graph. After the first write all cells are programmed to level q_1, and we use the similar method to generate the encoder and decoder for the $(q-q_1)-$ary constrained codes with respect the constraint (7.1).

7.6 Conclusion

WOM codes against cell-to-cell interference WOM codes are explored in this paper. Based on Delta-WOM, and constrained coding, we present various bounds of the codes. Also we present code constructions, one of which is based on the efficient t successive writes Diamond-WOM codes, which is proved to approach its rewriting capacity.

7.7 Appendix

7.7.1 The proof of theorem 45

Proof. $\mathcal{A}^{\mathbf{n}} = (\mathcal{B}-\mathcal{C})^{\mathbf{n}} = \sum_{\mathbf{i=0}}^{\mathbf{n}} \binom{\mathbf{n}}{\mathbf{i}} \cdot \mathcal{B}^{\mathbf{n-i}} \cdot \mathcal{C}^{\mathbf{i}} \cdot (-\mathbf{1})^{\mathbf{i}}$, thus we next try to get the expression for $\mathcal{B}^{\mathbf{n-i}} \cdot \mathcal{C}^{\mathbf{i}}$. By induction, we obtain that $\mathcal{C}^{\mathbf{i}}$ equals

$$\begin{pmatrix} \mathbf{0} & \mathcal{Q}^{i}_{(q-i(1+D))\times(q-i(1+D))} \\ \mathbf{0} & \mathbf{0} \end{pmatrix},$$

where $\mathcal{Q}^{i}_{(q-i(1+D))\times(q-i(1+D))} =$

$$\begin{pmatrix} \binom{i-1}{0} & \binom{i}{1} & \cdots & \binom{i+q-i(1+D)-2}{q-i(1+D)-1} \\ 0 & \binom{i-1}{0} & \cdots & \binom{i+q-i(1+D)-3}{q-i(1+D)-2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \binom{i-1}{0} \end{pmatrix},$$

and $\mathcal{B}^{n-i}$ equals $\mathcal{Q}^{n-i}_{q\times q}$. Thus after computation, we get $\mathcal{B}^{n-i}\cdot\mathcal{C}^{\mathbf{i}}$ is

$$\begin{pmatrix} \mathbf{0} & \mathcal{Q}^{n}_{(q-i(1+D))\times(q-i(1+D))} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}.$$

Denoting $\lfloor\frac{q}{1+D}\rfloor = p$, we obtain $\mathbf{a_0}\cdot\mathbf{A^n}\cdot\ \mathbf{1_q} = \sum_{\mathbf{i=0}}^{\mathbf{p}}\mathbf{a_0}\binom{\mathbf{n}}{\mathbf{i}}\mathcal{B}^{\mathbf{n-i}}\mathcal{C}^{\mathbf{i}}\mathbf{1_q}^{\mathbf{T}}(-\mathbf{1})^{\mathbf{i}}$. Then we compute the sum of the first $(1+D)$ rows of $\mathcal{B}^{n-i}\mathcal{C}^{i}$. By induction, we compute the sum of the jth row is,

$$\binom{n-1+q-i(1+D)-(j-1)}{q-i(1+D)-1-(j-1)},$$

where $1\leq j\leq 1+D$. That leads to $\mathbf{a_0}\mathcal{B}^{n-i}\mathcal{C}^{i}\mathbf{1_q} = \sum_{j=1}^{1+D}\binom{n-1+q-i(1+D)-(j-1)}{q-i(1+D)-1-(j-1)} = \binom{n+q-i(1+D)}{n}-\binom{n+q-(i+1)(1+D)}{n}$. □

7.7.2 Code construction for diamond-WOM

In this section we present the code construction, t successive writes Diamond-WOM codes.

7.7.2.1 Determining optimal parameters

Suppose we have n Diamond-WOM cells, all possible states are $0, 1, ..., q-1$, and initially all cells are at state 0. For the jth $(1\leq j\leq t)$ rewrite, the number of cells flipped from initial state to state i $(0<i\leq q-2)$ is a preset faction $p_i^{(j)}$ of the remaining cells, and $p_{q-1}^{(j)}$ is always 0.

The following lemmas present the optimal choices of parameters $p_i^{(j)}$ $(i\neq q-1)$ such that the total rate is maximal.

First we show that if we fix $p_0^{(i)}$ $(i=1,2,...,t-1)$, the formula (7.3) can be expressed in a succinct way, and it presents the optimal choices for $p_j^{(i)}$ $(j\neq 0, q-1$ and $i=1,2,...,t)$.

Lemma 49. $C_t = \{(R_1, \cdots, R_t) | R_1 \leq h(p_0^{(1)}, \frac{1-p_0^{(1)}}{q-2}, \cdots, \frac{1-p_0^{(1)}}{q-2}, 0), \cdots,$
$R_i \leq (\prod_{j=1}^{i-1} p_0^{(j)}) h(p_0^{(i)}, \frac{1-p_0^{(i)}}{q-2}, \cdots, \frac{1-p_0^{(i)}}{q-2}, 0), \cdots, R_t \leq (\prod_{i=1}^{t-1} p_0^{(i)}) \log_2 (q-1)\}$.

Proof. Once $p_0^{(i)}$ $(i = 1, 2, \cdots, t-1)$ is fixed, we derive that $h(\vec{p^{(i)}}) \leq h(p_0^{(i)}, \frac{1-p_0^{(i)}}{q-2}, \cdots, \frac{1-p_0^{(i)}}{q-2}, 0)$ by Jensen's inequality and the fact that $\log_2 x$ is a strictly convex function. Similarly, we get $h(\vec{p^{(t)}}) \leq h(\frac{1}{q-1}, \cdots, \frac{1}{q-1}, 0) = \log_2 (q-1)$. □

The next lemma presents the optimal choices of $p_0^{(j)}$ $(1 \leq j \leq t-1)$.

Lemma 50. The total rate of t successive writes of Diamond-WOM of previous lemma, $C(t) = \sum_{i=1}^{t} R_i$, is maximal when $p_0^{(j)} = \frac{1}{1+\frac{(q-2)}{2^{RC_j}} e^{q-3}} \triangleq p_0^{*(j)}$, where $j = 1, 2, \cdots, t-1$, RC_j can be recursively obtained by $RC_{j-1} = h(p_0^{*(j-1)}, \frac{1-p_0^{*(j-1)}}{q-2}, \cdots, \frac{1-p_0^{*(j-1)}}{q-2}, 0) + p_0^{*(j-1)} RC_j$ and $RC_{t-1} = \log_2 (q-1)$.

Proof. Proving this result involves finding $p_0^{(1)}, p_0^{(2)}, ..., p_0^{(t-1)}$ that maximize

$$h(p_0^{(1)}, \frac{1-p_0^{(1)}}{q-2}, \cdots, \frac{1-p_0^{(1)}}{q-2}, 0) + \cdots + (\prod_{i=1}^{t-2} p_0^{(i)}) h(p_0^{(t-1)}, \frac{1-p_0^{(t-1)}}{q-2}, \cdots, \frac{1-p_0^{(t-1)}}{q-2}, 0)$$

$$+ (\prod_{i=1}^{t-1} p_0^{(i)}) \log_2 (q-1).$$

We first fix $p_0^{(1)}, ..., p_0^{(t-2)}$, and find out $p_0^{(t-1)}$ maximizing $h(p_0^{(t-1)}, \frac{1-p_0^{(t-1)}}{q-2}, ..., \frac{1-p_0^{(t-1)}}{q-2}, 0) + p_0^{(t-1)} \log_2 (q-1)$. The answer is $p_0^{*(t-1)} = \frac{1}{1+\frac{q-2}{q-1} e^{-(q-3)}}$ and

$$RC_{t-1} = h(p_0^{*(t-1)}, \frac{1-p_0^{*(t-1)}}{q-2}, ..., \frac{1-p_0^{*(t-1)}}{q-2}, 0) + p_0^{*(t-1)} RC_t.$$

Similarly, using this result and fixing $p_0^{(1)}, ..., p_0^{(t-3)}$, we will find $p_0^{*(t-2)} = \frac{1}{1+\frac{q-2}{2^{C_{t-1}}} e^{-(q-3)}}$. Repeating this process, we finish verifying the correctness of this lemma. □

Combining the last two lemmas, the following theorem, which determines the optimal parameters, holds immediately.

Theorem 51. The total rate of t successive writes of Diamond-WOM of EQ.(7.3), $C(t) = \sum_{i=1}^{t} R_i$, is maximal when $p_i^{(j)}$ $(1 \leq j \leq t$ and $0 \leq i \leq q-2)$ are determined by the last two lemmas.

The ith $(0 \leq i \leq t)$ write is first to flip all non-zero cells to the state $q-1$, and then flip exactly the fraction $(1 - p_0^{*(i)})$ of remaining initial cells, which is $n \prod_{j=0}^{i-1} p_0^{*(j)}$, evenly to nonzero states except the state $q-1$.

Next, we analyze rate of the ith write, R_i $(1 \leq i \leq t)$.

Theorem 52. $R_i < \prod_{j=0}^{i-1} p_0^{(j)} h(p_0^{*(i)}, \frac{1-p_0^{*(i)}}{q-2}, ..., \frac{1-p_0^{*(i)}}{q-2})$.

Proof. We denote n' as remaining cells at state 0 before the ith write, which is actually $n \prod_{j=0}^{i-1} p_0^{*(j)}$. According to the writing scheme of the ith write, $R_i = \frac{\log_2 \binom{n'}{k_1, k_1, ..., k_{q-2}}}{n}$, where $k_1 = k_2 = ...k_{q-2} = n' \frac{1-p_0^{*(i)}}{q-2}$ and we use the notation, Permutations of Multiset:

$$\binom{n}{k_1, ..., k_{q-2}} = \frac{n!}{k_1! k_2! ... k_{q-2}! (n - k_1 - ... - k_{q-2})!}.$$

Using this notation, the following polynomial can be expressed in a clean form: $(1 + 2^{r_1} + 2^{r_2} + ... + 2^{r_{q-2}})^n = \sum_{\vec{k}=\vec{0}}^{\vec{n}} 2^{\vec{r}\vec{k}} \binom{n}{k_1, ..., k_{q-2}}$, where $\vec{r}\vec{k}$ is the inner product of two vectors $(r_1, ..., r_{q-2})$ and $(k_1, ..., k_{q-2})$, $\vec{k} = \vec{0}$ means all its elements start from 0, and similarly, $\vec{k} = \vec{n}$ indicates every element k_i ends at n.

The outline of the proof is $R_i \overset{(a)}{<} \frac{\log_2 \sum_{\vec{k}=\vec{0}}^{\vec{q}^{(i)} n'} \binom{n'}{k_1, k_1, ..., k_{q-2}}}{n} \leq \prod_{j=0}^{i-1} p_0^{*(j)} h(p_0^{*(i)}, \frac{1-p_0^{*(i)}}{q-2}, ..., \frac{1-p_0^{*(i)}}{q-2}, 0)$, where $\vec{q}^{(i)} = (\frac{1-p_0^{*(i)}}{q-2}, ..., \frac{1-p_0^{*(i)}}{q-2})$. It is quit straightforward to see the correctness of the first inequality (a). Now we prove the latter inequality.

Since we have

$$
\begin{aligned}
2^{\vec{r}\vec{q}^{(1)}n'} \sum_{\vec{k}=q^{\vec{(i)}}n'}^{\vec{n'}} \binom{n'}{k_1, k_2, ..., k_{q-2}} &\leq \sum_{\vec{k}=q^{\vec{(i)}}n'}^{\vec{n'}} \binom{n'}{k_1, k_2, ..., k_{q-2}} 2^{\vec{r}\vec{k}} \\
&\leq \sum_{\vec{k}=\vec{0}}^{\vec{n'}} \binom{n'}{k_1, k_2, ..., k_{q-2}} 2^{\vec{r}\vec{k}} \\
&= (1 + 2^{r_1} + 2^{r_2} + ... + 2^{r_{q-2}})^{n'},
\end{aligned} \tag{7.4}
$$

$\sum_{\vec{k}=q^{\vec{(i)}}n'}^{\vec{n'}} \binom{n'}{k_1,k_2,...,k_{q-2}} \leq (2^{-\vec{r}\vec{q}^{(i)}} + 2^{r_1-\vec{r}\vec{q}^{(i)}} + ... + 2^{r_{q-2}-\vec{r}\vec{q}^{(i)}})^{n'}$.

Choose $r_j = \log_2 \frac{p_j^{(i)}}{1-\sum_{k=1}^{p-2} p_j^{(i)}} = \log_2 \frac{p_j^{(i)}}{p_0^{(i)}}$.

Then the sum above is not bigger than $2^{n'h(\vec{p}^{(i)})}(p_0^{(i)} + p_1^{(i)} + ... + p_{q-2}^{(i)})^{n'}$, which is equal to $2^{n'h(\vec{p}^{(i)})}$.

Since $\sum_{\vec{k}=\vec{0}}^{\vec{q}^{(i)}n'} \binom{n'}{k_1,k_2,...,k_{q-2}} = \sum_{\vec{k}=\vec{q}^{(i)}n'}^{\vec{n'}} \binom{n'}{k_1,k_2,...,k_{q-2}}$, we get $R_i < \prod_{j=0}^{i-1} p_0^{*(j)} h(p_0^{*(i)}, \frac{1-p_0^{*(i)}}{q-2}, ..., \frac{1-p_0^{*(i)}}{q-2}, 0)$.

□

Clearly, the following theorem holds:

Theorem 53. Our construction approaches the rewriting capacity of Diamond-WOM.

7.7.2.2 The index of cell state vector

One unsolved problem of the ith write is the decoding function and encoding function: for the encoding we map the index to the cells state vector; for the decoding we obtain its index from the cells state vector. Let

$$
f_i : \vec{c} = (c_0, c_1, ..., c_{n-1}) \rightarrow \{0, 1, ..., 2^{nR_i} - 1\}, \tag{7.5}
$$

be the function computing the index of a vector for the ith rewrite. The following example presents the basic idea, which is an extension of [15].

Consider the example where we have 7 Diamond-WOM 5-ary cells, and we can use 3 cells for the first write according to some distribution. We compute the index of the sequence $\vec{c} = (0030201)$. Before that, we define some notations: the support for a q-ary n-tuple $\vec{c}$, $supp(\vec{c})$, that is $supp(\vec{c}) = \{i | i \in \{0, 2, ..., n-1\}; c_i \neq 0\}$, and its complementary set $\overline{supp}(\vec{c}) = \{i | i \in \{0, 2, ..., n-1\}; c_i = 0\}$. For example, for previous $\vec{c}$, $supp(\vec{c}) = \{k_0, k_1, k_2\} = \{2, 4, 6\}$. The basic idea of $f_i(\cdot)$ is to count the number of sequences of lower indexes, that is indexing them according to their alphabet order. We consider every $i \in supp(\vec{c})$. For $k_0 = 2$, corresponding to the state 3, if we flip it to lower states, all the sequences are lexically before $\vec{c}$: if we flip it to a nonzero state, there are $(3-1)\binom{6-2}{3-1}(4-1)^{3-1} = 108$ sequences; flipped to zero state, there are $\binom{6-2}{3}(4-1)^3 = 108$ sequences. Similarly, for k_1, k_2 flipping them to lower states and leaving the preceding cells unchanged, there are $1\binom{6-4}{2-1}(4-1)^1 + \binom{6-4}{2}(4-1)^2 = 15$ cells. Therefore, the index of $\vec{c}$ is $f(\vec{c}) = 108 + 108 + 15 = 221$.

More generally, consider a vector $\vec{c}_i = (c_0 c_1 ... c_{n-1})$ of q states after the ith write, we can obtain its sub-vector $\vec{c}'_i = (c'_0 c'_1 ... c'_{n \prod_{j=0}^{i-1} p_0^{*(j)} - 1})$, where c'_i is some c_j, not at state $(q-1)$, and the support of $\vec{c}'$ is $\{k_0, k_1, ..., k_{l-1}\}$. We have $f_i(\vec{c}_i) = f_i(\vec{c}'_i) =$:

$$\sum_{i=0}^{l-1} \binom{n \prod_{j=0}^{i-1} p_0^{*(j)} - 1 - k_i}{l-i} (q-1)^{l-i} + (c'_{k_i} - 1) \binom{n \prod_{j=0}^{i-1} p_0^{*(j)} - 1 - k_i}{l-1-i} (q-1)^{l-1-i}.$$

Conversely, we define $f_i^{-1}(y)$ to the process of determining cell state vector corresponding to data y. Given the data to write, y, and the cell state vector $\vec{c}_{i-1}$ after the $(i-1)^{th}$ write, in order to determine $\Delta\vec{c}_i = (c_0 c_1 ... c_{n-1})$ (according to which we flip cells), we do the following things: first we flip all non-zero state cells to the state $(q-1)$, and denote now the cell state vector as $\vec{c}_{fi}$; similarly, we can obtain its sub-vector $\vec{c}'_{fi}$, which consists of all initial state cells, thus flipping cells according to $\Delta\vec{c}_i$ is equiva-

lent to flip cells as $\Delta\vec{c}'_{fi}$; we decide l, which is the number of cells we flip in the ith time, $n(1-p_0^{(i)})\prod_{j=0}^{i-1} p_0^{*(j)}$, and ease to obtain if we set a cell to remember the writing times; $c_{k_0} \in \{0, 1, ..., q-2\}$ is the largest integer and k_0 is the smallest integer such that $(c_{k_0}-1)\binom{n\prod_{j=0}^{i-1} p_0^{*(j)}-1-k_0}{l-1}(q-1)^{l-1} + \binom{n\prod_{j=0}^{i-1} p_0^{*(j)}-1-k_0}{l}(q-1)^{l} \leq y$..., until all k_i and c_{k_i} are determined. $\Delta\vec{c}'_{fi}$ consists of all c_{k_i} $(i \in [0, l-1])$. We use the function $f_i^{-1}(y)$ to denote the above process.

8. CODING FOR SECURE WRITE-EFFICIENT MEMORIES

8.1 Introduction

Flash memories are becoming ubiquitous due to the advantages such as higher data density, scaling size and non-volatility. The two most conspicuous challenges of flash memories are its limited lifetime, i.e., the so called *endurance* problem, and the difficulty of secure deletion, i.e., the so called *insecure deletion*. Such characteristics are different from traditional storage media, and posing a threat to their further usages. In this work, we propose a novel coding model here, secure write-efficient memory (WEM), to address the two challenges jointly, and focus on information theoretical results, i.e., rewriting-rate-equivocation region and its secrecy rewriting capacity.

In the following, we present the two challenges in detail (i.e.,endurance and insecure deletion), which motivate us to propose the secure WEM model to solve them jointly.

8.1.1 Endurance and rewriting codes

Flash memories are significant non-volatile memory techniques. The unit of flash memory is a cell. Each flash chip is composed of blocks, each block consists of pages, and each page is made up of cells. There are three operations on flash cells, read, write/program and erase. The granularity of read/write and erase is a page and a block, respectively.

The first challenge in flash memories is *endurance*. Endurance means flash memory can only experience a limited number of program/erase cycles, beyond which the cell quality degradation can no longer be accommodated by the memory system fault tolerance capacity.

Rewriting code is a powerful coding technology to solve the endurance problem from information theory and coding theory perspective. Figure 8.1 presents us the rewriting code model, where the rewriter selects a new codeword $y_0^{N-1} = (y_0, y_1, \cdots, y_{N-1})$ based

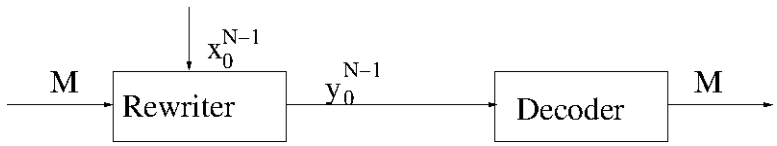


Figure 8.1: Rewriting code model, where M is the message to rewrite, x_0^{N-1} is the current cell state, and y_0^{N-1} is the rewrite codeword.

on the message – which is M– to rewrite to the underlying storage medium, and current cell state of the storage medium $x_0^{N-1} = (x_0, x_1, \cdots, x_{N-1})$ such that the predefined constraint between x_0^{N-1} and y_0^{N-1} is satisfied.

Based on various constraints, different rewriting code models such as write-once memory (WOM) codes [61] and WEM codes [2] have been proposed, and optimal code constructions [9], [65] and [48] have been constructed for them, respectively. For WOM, the constraint is $y_i \geq x_i$ for $i = 0, 1, \cdots, N-1$, that is the cell level can only increase but not decrease. We repeat the definition of WEM as follows, before which we present some notations.

Let $\mathcal{X}$ be the alphabet of the symbol stored in a cell. $\forall x, y \in \mathcal{X}$, let the rewriting cost of changing a cell's level from x to y be $\varphi(x, y)$, which may be time or energy taken. Given N cells and $x_0^{N-1}, y_0^{N-1} \in \mathcal{X}^N$, let $\varphi(x_0^{N-1}, y_0^{N-1}) = \frac{1}{N} \sum_{i=0}^{N-1} \varphi(x_i, y_i)$ be the rewriting cost of changing the N cell levels from x_0^{N-1} to y_0^{N-1}.

Let $\mathcal{D} \subseteq \mathbb{N}$. We use $\mathcal{D}$ to denote the $|\mathcal{D}|$ possible values of the data stored in the N cells. Let the decoding function be $\mathbf{D} : \mathcal{X}^N \to \mathcal{D}$, which maps the N cells' levels to the data they represent. Let the rewriting function be $\mathbf{R} : \mathcal{X}^N \times \mathcal{D} \to \mathcal{X}^N$, which changes the N cells' levels to represent the new input data. (Note that the rewriting function can be either deterministic or stochastic.)

Definition 54. [2] An (N, M, D) write-efficient memory code consists of

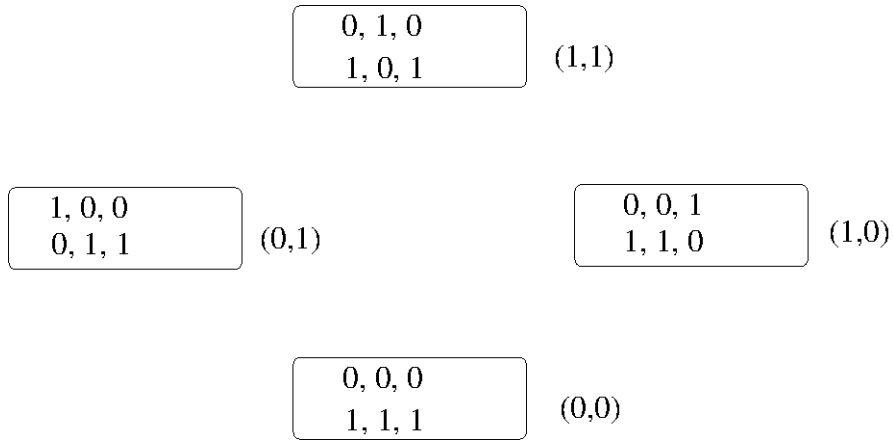


Figure 8.2: An example of (3, 4, 1) WEM, where two sequences of numbers inside a box are codewords, the number outside a box is the data represented by the codewords inside the box, e.g., both codewords $(0, 0, 0)$ and $(1, 1, 1)$ represent data $(0, 0)$, the rewriting cost metric is the Hamming distance, that is $\varphi(0, 0) = \varphi(1, 1) = 0$ and $\varphi(0, 1) = \varphi(1, 0) = 1$.

- $\mathcal{D} = \{0, 1, \cdots, M-1\}$ and $\bigcup_{i=0}^{M-1} \mathcal{C}_i$, where $\mathcal{C}_i \subseteq \mathcal{X}^N$ is the set of codewords representing data i. We require $\forall i \neq j, \mathcal{C}_i \bigcap \mathcal{C}_j = \emptyset$;
- A rewriting function $\mathbf{R}(i, x_0^{N-1})$ such that $\varphi(x_0^{N-1}, \mathbf{R}(i, x_0^{N-1})) \leq D$ for any $i \in \mathcal{D}$ and $x_0^{N-1} \in \mathcal{X}^N$;
- A decoding function $\mathbf{D}(y_0^{N-1})$ such that $\mathbf{D}(\mathbf{R}(x_0^{N-1}, i)) = i$ for any $i \in \mathcal{D}$.

That is, the first condition indicates that each data is represented by a group of codewords, the second one indicates that during each rewrite the average rewriting cost between the current codeword x_0^{N-1} and the updated codeword y_0^{N-1} is less than a predefined number, and the third one indicates that the decoder knows the rewritting message given a rewriting codeword. A concrete example of WEM is presented in Figure 8.2.

Although WEM is a reasonable model for solving endurance in phase-change memory [44], it is worth noting that WEM can also be used in flash memories when the data

representation scheme is rank modulation [36, 46]. On the other hand, as pointed out by Fu et al [27], "the binary WOM and the generalized WOM are special cases of deterministic WEM". Therefore, in this work we focus on the WEM as our main tool for rewriting codes.

8.1.2 Insecure deletion and wiretap codes

Flash memory is commonly accessed through a *Flash Translation Layer* (FTL) [31], which is used in USB sticks, solid state drives, etc. One core function of FTL is to maintain a physical-to-logical mapping table. FTLs access the raw flash memory directly by a Physical Address (PA), and the PA is mapped to a Logical Address (LA) that computer system uses to access data. Other functions of FTL are wear leveling and garbage collection, etc.

The second challenge in flash memories is *insecure deletion* (or *insecure erasure*) ([70, 58, 39]). Insecure deletion means FTL produces multiple copies of data that can not be deleted completely as they are either impossible or costly, however, a sophisticated attacker can recover and obtain information about the data.

We illustrate the insecure deletion in detail here and we use Figure 8.3 to further illustrate it. The first reason causing this is the existence of multiple copies of codewords in flash memories. Flash memories are not perfect as there are various errors [49], thus a strong error correcting code (e.g., BCH code or LDPC code) is used to combat errors. *Memory scrubbing* [64] is also used to protect flash memories, which is to correct a noisy codeword and write a new error-free codeword back to memories. However, due to the *out-of-place* rewriting policy, the updated codeword is stored at a new physical address and the original codeword remains in memories. Those mechanisms lead to multiple copies of codewords existing in memories. Other reasons causing this are weal leveling and garbage collection.

When the flash is attacked by an eavesdropper, who is able to trace any copy of code-

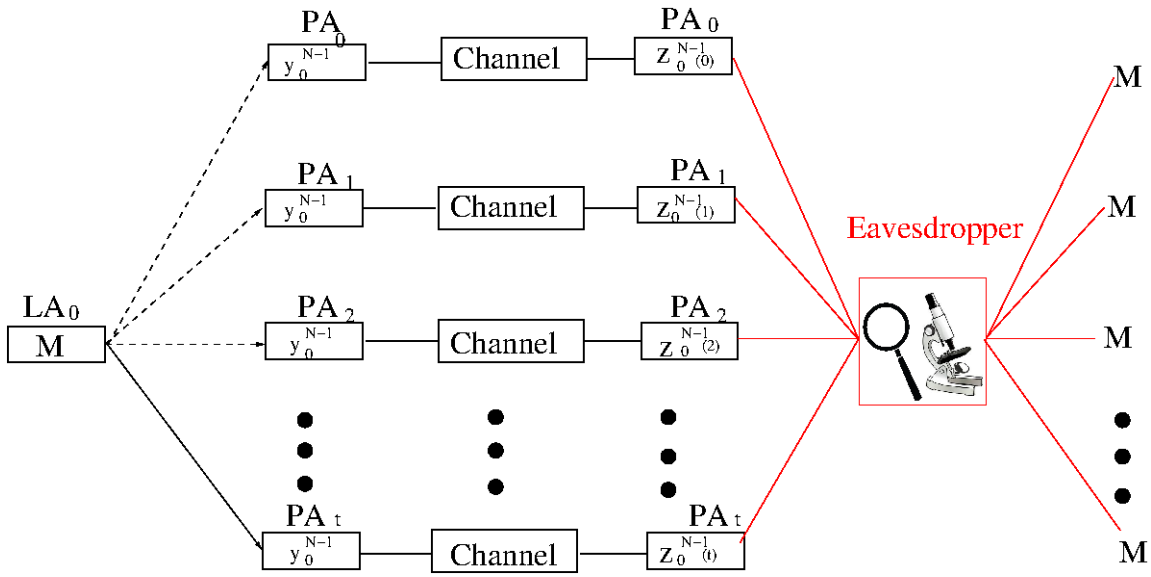


Figure 8.3: Illustration of insecure deletion in flash memories

words, and is aware of all encoding and decoding algorithms, the sensitive information can be leaked. Due to the imperfections of the physical erasure process and the FTL, perfecting erasure data is either impossible or costly [39].

Recently, insecure deletion has attracted intensive research attention due to wild usage of non-volatile memories and the underlying log-structured file system [62]. Different approaches from different levels, such as [70] from architecture level, [58] from operating system level, etc, have been proposed to solve this problem. Interested readers are encouraged to refer to [39], a comprehensive survey of secure data deletion.

Wiretap codes [73] provide unconditional information-theoretic security. More precisely, in the wiretap code setting (see Figure 8.4), Alice wishes to send message M to Bob through a *main channel*, but her transmissions are also accessible to an eavesdropper Eve through another channel, *wiretap channel*. That is, Alice selects a codeword y_0^{N-1} based on the message M and random bits to send through the main channel and the wiretap channel. w_0^{N-1} and z_0^{N-1} are noisy codewords of y_0^{N-1} passing through the two

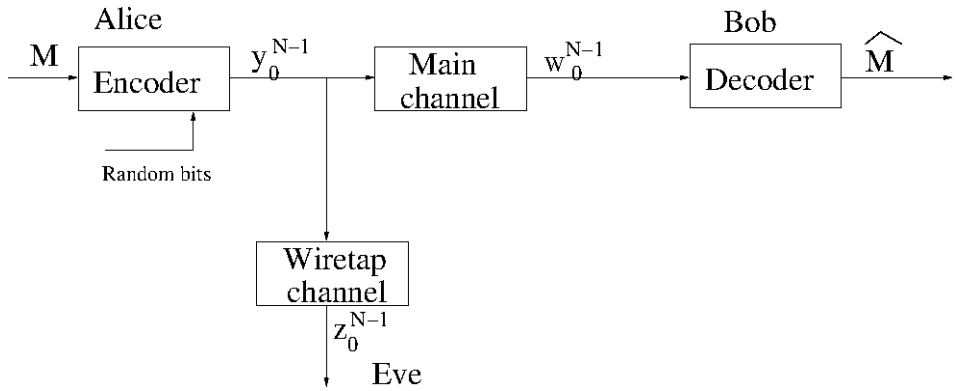


Figure 8.4: Illustration of wiretap channel code.

channels, respectively. After receiving w_0^{N-1}, Bob maps it to an estimate of the original message. The goal of wiretap channel codes is to design a *reliable* and *secure* communication scheme, that is, Bob can reliably recover the message, while the information leaked to Eve is negligible.

Wiretap codes have been gaining escalating practical interest due to its two striking benefits over conventional cryptography. One is no computational assumptions, which provides long-term security even facing with the incoming quantum computing era, and the other is no keys distribution, which is attractive for vulnerable and low-power devices. Popular as wiretap code is for secure wireless communication [51], there is barely no research work [14] considering its application to non-volatile memory storages.

8.1.3 Contribution and structure

In this paper, we first propose a novel coding model here – secure write efficient memory– which has both properties of rewriting codes as well as wiretap channel codes to jointly solve the endurance and the insecure deletion problem. Figure 8.5 presents us the big picture of this setting, where the sensitive data M is encoded using *rewriting code* y_0^{N-1}, noisy codewords of y_0^{N-1} are accessible to both a legal decoder, who can reliably

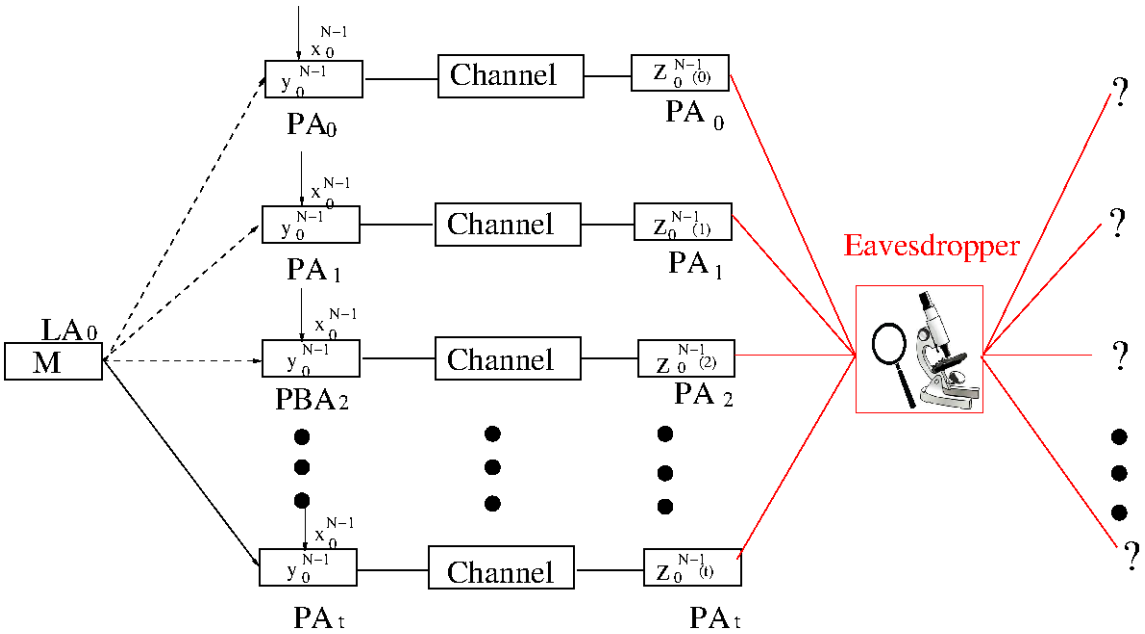


Figure 8.5: Illustration of rewriting codes with security constraint in flash memories

retrieve M, and an eavesdropper, whose knowledge of M is negligible to *satisfy the security constraint*. Rigorous definition of the codes is deferred to the later section. To the best knowledge of authors, this is the first work to study rewriting code with security concern under the wiretap channel setting. To that end, in this work we mainly explore the fundamental information theoretical results, i.e., achievable rate region and its capacity.

The rest of this paper is structured as follows. In Section 8.2, we formally define the secure write-efficient memory model and we list the main results of this paper. In Section 8.3, we study the achievable regions for secure WEM. In Section 8.4, we study the secrecy rewriting capacities. The conclusion and future work are obtained in Section 8.5.

8.2 Definition of Secure WEM

In this section, we formally present the secure WEM model. We first present a secure WEM model, where there is an upper bound on the rewriting cost of each rewriting. We

then generalize the secure WEM by loosing the rewriting cost constraint, i.e., here only an upper bound on the average rewriting cost constraint over all rewritings is required.

8.2.1 Secure WEM with a maximal rewriting cost constraint

The secure WEM model is illustrated in Figure 8.6. In this setting, Alice wishes to store messages to a limited lifetime storage medium using a rewriting code, WEM [2], the messages are accessible to Bob through a storage channel CH_1, $\mathbb{W} = (\mathcal{X}, \mathcal{W}, W_{W|X})$, but her transmissions also reach an eavesdropper Eve through a wiretap channel CH_2, $\mathbb{P} = (\mathcal{X}, \mathcal{Z}, P_{Z|Y})$, $Y \in \mathcal{X}$. This is illustrated in Figure 8.6, wherein M is the message that Alices wishes to store. Based on the message M and the current cell level vector x_0^{N-1}, the rewriter maps M to a N-bit codeword y_0^{N-1}. This codeword is transmitted across the storage channel and the wiretap channel resulting w_0^{N-1} and z_0^{N-1}. Finally, Bob estimates y_0^{N-1} to recover the message M.

The goal of secure WEM codes is to design a coding scheme, i.e., a rewriting function and a decoding function, such that it is possible to store messages cost-effectively and securely as the length of codeword tends to infinity. Being cost-effective means for each rewrite the defined rewriting cost, i.e., which is measured by $\varphi(x_0^{N-1}, y_0^{N-1})$ for a defined cost $\varphi(\cdot)$, has to be less than a predefined number to solve the endurance problem. Being secure means the uncertainty of the eavesdropper about the message M after observing the wiretap channel output z_0^{N-1}, i.e., which is measured by $\frac{1}{N}H(M|z_0^{N-1})$ [73], also satisfies a predefined constraint to solve the insecure deletion problem.

Note that CH_1 and CH_2 model various errors proposed in [49] and CH_2 is usually more noisier than CH_1. This is due to the fact that errors in flash memory are accumulated with each flash operation, that is w_0^{N-1} at a legitimate decoder is usually the latest valid codeword copy and thus has accumulated relatively few errors, while z_0^{N-1} is usually one copy of out-of-date codewords and thus has accumulated relatively more errors [49]. For

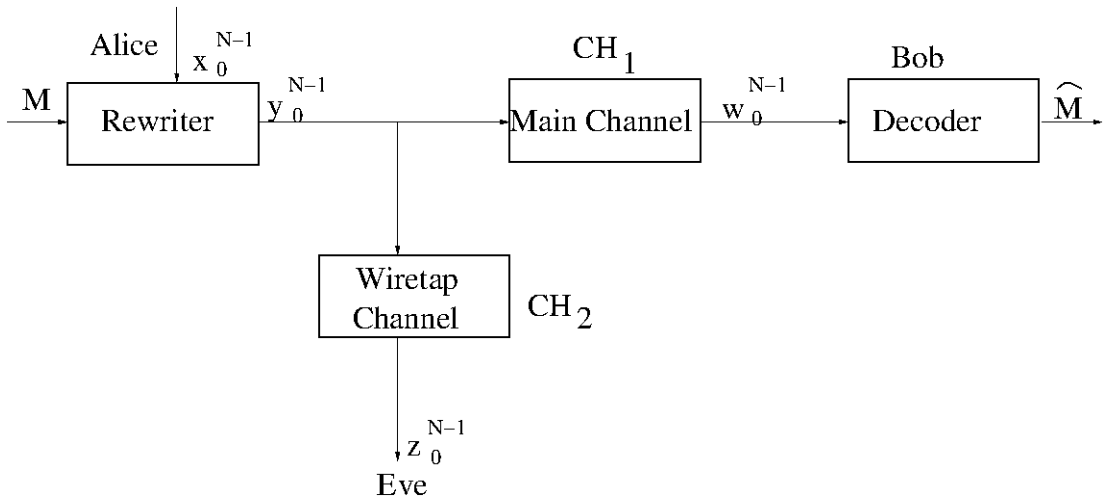


Figure 8.6: The secure WEM model. CH_1, CH_2 are the main channel and the wiretap channel, respectively. $M, x_0^{N-1}, y_0^{N-1}, z_0^{N-1}, w_0^{N-1}$ and $\hat{M}$ are the message to rewrite, the current cell states, the rewrite codeword, the wiretap channel's output, the main channel's output and the estimated message, respectively.

simplicity, we assume that CH_1 is noiseless, and leave the opposite case as the future work. For this reason, we omit the rigorous definition of the notion more noisier, and interested readers are referred to [8].

The formal definition of the secure WEM model is below.

Definition 55. An $(N, 2^{NR}, R_e, D)$ secure write-efficient memory code with a wiretap channel $\mathbb{P} = (\mathcal{X}, \mathcal{Z}, P_{Z|Y})$ and the rewriting cost function $\varphi(\cdot)$ consists of

- $\mathcal{D} = \{0, 1, \cdots, 2^{NR} - 1\}$ and $\bigcup_{i=0}^{2^{NR}-1} \mathcal{C}_i$ $(\forall i \neq j, \mathcal{C}_i \bigcap \mathcal{C}_j = \emptyset)$;
- $\mathbf{R}(M, x_0^{N-1})$ such that
  - $\varphi(x_0^{N-1}, \mathbf{R}(M, x_0^{N-1})) \leq D$ for any $M \in \mathcal{D}$ and $x_0^{N-1} \in \mathcal{X}^N$;
  - $\frac{1}{N} H(M|z_0^{N-1}) \geq R_e - \epsilon$ for any $M \in \mathcal{D}, z_0^{N-1} \in \mathcal{Z}^n$, $\epsilon > 0$ as $N \to \infty$.

- $\mathbf{D}(y_0^{N-1})$ such that $\mathbf{D}(\mathbf{R}(x_0^{N-1}, M)) = M$ for all $M \in \mathcal{D}$ and $x_0^{N-1} \in \mathcal{X}^N$.

Note that the first requirement of the rewriting function is the same as that of WEM [2], and the second one is added here to consider the uncertainty of the message at the eavesdropper, therefore $(N, 2^{NR}, R_e, D)$ codes are actually a subset of write-efficient memory codes [2], and we term the code as secure WEM code.

Also note that in the above the security measure is the weak security condition. Besides it, other security measures, such as the strong security condition [8] and the recently proposed semantic security measure [6], also exist, and we leave them as future work.

Fixed D, the rewriting cost function $\varphi(\cdot)$ and the wiretap channel $\mathbb{P} = (\mathcal{X}, \mathcal{Z}, P_{Z|X})$, a tuple $(R, R_e) \in \mathbb{R}^2$ is said to be *achievable* if there exists an $(N, 2^{NR}, R_e, D)$ codes. When $R_e = R$, we say it achieves full secrecy. The set of all achievable tuples is denoted by $\mathcal{R}^{swem}$, *rewriting-rate-equivocation region*. The secrecy rewriting capacity is $C^{swem}(D) \stackrel{def}{=} \sup_R\{R : (R, R) \in \mathcal{R}^{swem}\}$.

8.2.2 Secure WEM with an average rewriting cost constraint

The secure WEM code in definition 55 puts a constraint on the maximal rewriting cost for each rewriting. We now define a code with an average rewriting cost constraint. Before formally presenting its definition, we define the following terms.

Assume the sequence of data written to the storage medium is $\{M_1, \cdots, M_t\}$, where we assume M_i for $1 \leq i \leq t$ is uniformly distributed over $\mathcal{D}$, and the average rewriting cost is $\bar{D} \stackrel{def}{=} \lim_{t\to\infty} \frac{1}{t} \sum_{i=1}^{t} \varphi(x_0^{N-1}(i), \mathbf{R}(M_i, x_0^{N-1}(i)))$, where $x_0^{N-1}(i)$ is the current cell states before the ith update. By assuming the stationary distribution of cell levels x_0^{N-1} is $\pi(x_0^{N-1})$, $\bar{D} = \sum_{x_0^{N-1}} \pi(x_0^{N-1}) \sum_{j\in\mathcal{D}} \bar{D}_j(x_0^{N-1})$, where $\bar{D}_j(x_0^{N-1})$ is the average rewriting cost of updating cell levels x_0^{N-1} to a codeword representing $j \in \mathcal{D}$.

Definition 56. An $(N, 2^{NR}, R_e, D)_{ave}$ secure write-efficient memory code for wiretap channel $\mathbb{P} = (\mathcal{X}, \mathcal{Z}, P_{Z|Y})$ and the rewriting cost function $\varphi(\cdot)$ consists of

- $\mathcal{D} = \{0, 1, \cdots, 2^{NR} - 1\}$ and $\bigcup_{i=0}^{2^{NR}-1} \mathcal{C}_i$ ($\forall i \neq j, \mathcal{C}_i \bigcap \mathcal{C}_j = \emptyset$);
- $\mathbf{R}(M, x_0^{N-1})$ such that
  - $\bar{D} \leq D$;
  - and $\frac{1}{N} H(M|z_0^{N-1}) \geq R_e - \epsilon$ for any $M \in \mathcal{D}, z_0^{N-1} \in \mathcal{Z}^N$, $\epsilon > 0$ and $N \to \infty$.
- $\mathbf{D}(y_0^{N-1})$ such that $\mathbf{D}(\mathbf{R}(x_0^{N-1}, M)) = M$ for any $M \in \mathcal{D}$ and $x_0^{N-1} \in \mathcal{X}^N$.

That is, compared with $(N, 2^{NR}, R_e, D)$ code, the rewriting cost constraint for each rewrite is replaced by the average rewriting cost constraint over all rewritings.

Similarly, a tuple $(R, R_e)_{ave} \in \mathbb{R}^2$ is said to be *achievable* if there exists an $(N, 2^{NR}, R_e, D)_{ave}$ codes. When $R_e = R$, we say it achieves full secrecy. The set of all achievable tuples is denoted by $\mathcal{R}_{ave}^{swem}$, and $C_{ave}^{swem}(D) \stackrel{def}{=} \sup_R \{R : (R, R)_{ave} \in \mathcal{R}_{ave}^{swem}\}$.

8.3 Achievable Region of Secure WEM

In this section, we present one of our main contributions, that is the achievable region for secure WEM and its secrecy rewriting capacity and the proof of the achievable region is defered to Section 7.4.

8.3.1 Characterizing the achievable region for $\mathcal{R}^{swem}$

Let $\mathcal{P}(\mathcal{X} \times \mathcal{X})$ be the set of joint probability distributions over $\mathcal{X} \times \mathcal{X}$. For a pair of random variables $(X, Y) \in (\mathcal{X}, \mathcal{X})$, let $P_{XY}, P_X, P_{X|Y}$ denote the joint probability distribution, the marginal distribution, and the conditional probability distribution, respectively. $E(\cdot)$ denotes the expectation operator. If X is uniformly distributed over $\{0, 1, \cdots, q-1\}$, denote it by $X \sim U(q)$.

Theorem 57. Define $\mathcal{R}(P_{XY}) =$

$$\{(R, R_e): \begin{array}{ll} R & \leq H(Y|X) \\ R_e & \leq H(Y|Z) \\ R_e & \leq R \end{array} \},$$

where $P_{XY} \in \mathcal{P}(D) \stackrel{def}{=} \{P_{XY} : P_X = P_Y, E(\varphi(X,Y)) \leq D\}$, the joint distribution of X, Y, Z factorizes as $P_X P_{Y|X} P_{Z|Y}$, and the $P_{Z|Y}$ is given by wiretap channel $\mathbb{P} = (\mathcal{X}, \mathcal{Z}, P_{Z|Y})$.

Then, the rewriting-rate-equivocation region of the secure WEM is the convex region: $\mathcal{R}^{swem} = \bigcup_{P_{XY}} \mathcal{R}(P_{XY})$.

The first inequality in Theorem 57 is the same as the rewriting rate for write-efficient memories [2, Theorem 2], which is an immediate result as secure WEM is an especial case of WEM. The second inequality is the major contribution of this paper.

The typical shape of the above achievable region $\mathcal{R}(P_{XY})$ is presented in Figure 8.7: type one is the case where $H(Y|Z) \leq H(Y|X)$ for a given $P_{XY} \in \mathcal{P}(D)$, and type two is the other case.

By specializing Theorem 57 to full secrecy, we obtain the following result for secrecy rewriting capacity.

Corollary 58. The secrecy rewriting capacity of secure WEM $(N, 2^{NR}, R_e, D)$ code with wiretap channel $\mathbb{P} = (\mathcal{Z}, \mathcal{Y}, \mathcal{P}_{\mathcal{Z}|\mathcal{Y}})$ and the rewriting cost function $\varphi(\cdot)$ is:

$$C^{swem}(D) = \max_{P_{XY} \in \mathcal{P}(D)} \{\min\{H(Y|X), H(Y|Z)\}\},$$

where the definition of $\mathcal{P}(D)$ is the same as that of Theorem 57.

Let us examine some extreme cases: when the eavesdropper obtains the same obser-

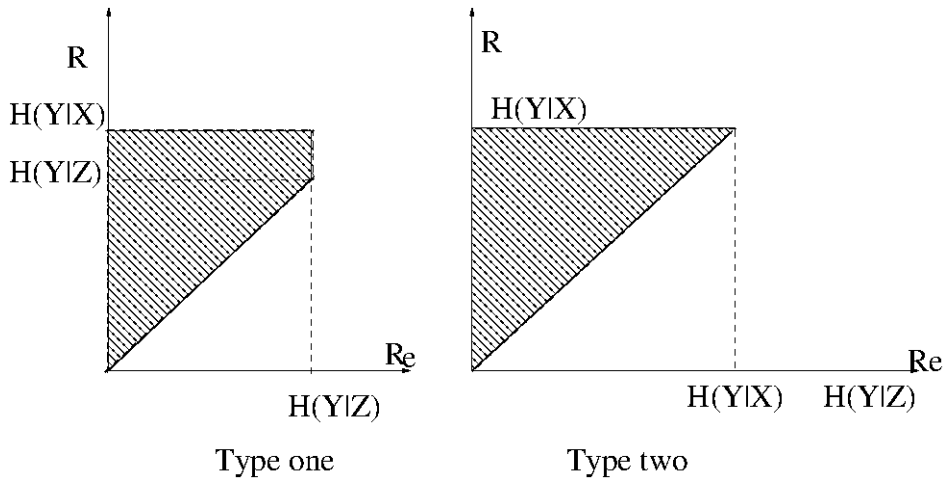


Figure 8.7: Typical shape of the achievable region in Theorem 57.

vation as the legitimate decoder, clearly no confidential messages can be securely transmitted. From the above theorem, we know that $Y = Z$, then $H(Y|Z) = 0$, and thus $C^{swem}(D) = 0$. On the other hand, when there is no eavesdropper, i.e., $Z \in \emptyset$, the result should be coinciding with original WEM code [2]. From theorem 57, we know that $C^{wem}(D) = \max_{P_{XY} \in \mathcal{P}(D)} H(Y|X)$, which is exactly the rewriting capacity of WEM.

We define the following terms to obtain further simpler results for secrecy rewriting capacity.

Definition 59. The WEM is *more capable* than wiretap channel $\mathbb{P} = (\mathcal{Z}, \mathcal{Y}, P_{\mathcal{Z}|\mathcal{Y}})$ if $I(X;Y) \geq I(Y;Z)$ for every $P_{XY} \in \mathcal{P}(D)$; The WEM is *less capable* than wiretap channel $\mathbb{P} = (\mathcal{Z}, \mathcal{Y}, \mathcal{P}_{\mathcal{Z}|\mathcal{Y}})$ if $I(X;Y) \leq I(Y;Z)$ for every $P_{XY} \in \mathcal{P}(D)$.

With the above notations, we have the following results for secrecy rewriting capacity.

Corollary 60. The secrecy rewriting capacity $C^{swem}(D)$ is $\max_{P_{XY} \in \mathcal{P}(D)} H(Y|X)$ if WEM is less capable than wiretap channel $\mathbb{P}$, (which is effectively the rewriting capacity of write-efficient memory [2, Theorem 2]) and $H(Y|Z)$ for $P_{XY} \in \mathcal{P}(D)$ if WEM is more capable than wiretap channel.

We present the following concrete example:

Example 61. Consider the following binary secure WEM $(N, 2^{NR}, R_e, D)$, where the rewriting cost function is the Hamming distance, that is $\varphi(0,0) = \varphi(1,1) = 0$, and $\varphi(0,1) = \varphi(1,0) = 1$, and wiretap channel $\mathbb{P}$ is the binary symmetric channel with flipping rate p. Based on [2, Theorem 4], the WEM rewriting capacity is $\mathcal{R}(D) = H(D)$ in this case. Therefore, if WEM is less capable, then $C^{swem}(D) = H(p)$; if WEM is more capable, then $C^{swem}(D) = H(D)$.

8.4 Proof of Theorem 57

8.4.1 Achievable regions for secure WEM

In this part, we show that the region presented in Theorem 57 is achievable. For simplicity, we only present details of type one region of Figure 9.9, and present the sketch for type two region of Figure 9.9 as it is similar to the previous one.

The proof for type one region is divided into the following three steps and we present them in detail in the following parts:

- Step 1: We use a random-coding argument and show that the existence of a sequence $(N, 2^{NR}, R_e, D)$ code such that $\frac{1}{N}L \stackrel{def}{=} \frac{1}{N}H(M) - \frac{1}{N}H(M|z_0^{N-1}) \leq \epsilon$ for some $\epsilon > 0$ and $R \leq H(Y|Z)$. This shows that the following sub-region of type one region is achievable: $\mathcal{R}'(P_{XY}) \stackrel{def}{=}$

$$\{(R, R_e) : \begin{array}{ll} R & \leq H(Y|Z) \\ R_e & \leq R \end{array} \},$$

  where $P_{XY} \in \mathcal{P}(D)$.

- Step 2: We show that the entire type one region in Theorem 57 is achievable with a minor modification of the code construction presented in step 1.

- Step 3: We show that the $\mathcal{R}^{swem}$ is convex.

8.4.1.1 Step 1: Achieving region $\mathcal{R}'(P_{XY})$

Background of strong typical-sequences We first present some background about strong typical-sequences. For more details, interested readers are referred to [20].

Let x_0^{N-1} be a sequence with N elements drawn from $\mathcal{X}$. Define the type of x_0^{N-1} by $\pi(x|x_0^{N-1}) = \frac{|\{i:x_i=x\}|}{N}$. The set $\mathcal{T}_\epsilon^N(X)$ is defined as $\mathcal{T}_\epsilon^N(X) = \{x_0^{N-1} : |\pi(x|x_0^{N-1}) - P_X(x)| \leq \epsilon, \forall x\}$. That is, the set of sequences for which the empirical frequency is within ϵ of the probability $P_X(x)$ for every $x \in \mathcal{X}$.

Let (x_0^{N-1}, y_0^{N-1}) be a pair of sequences with elements drawn from $(\mathcal{X}, \mathcal{Y})$. Define their joint type: $\pi(x, y|x_0^{N-1}, y_0^{N-1}) = \frac{|\{i:(x_i,y_i)=(x,y)\}|}{N}$ for $(x, y) \in \mathcal{X} \times \mathcal{Y}$. We denote $\mathcal{T}_\epsilon^N(XY) = \{(x_0^{N-1}, y_0^{N-1}) : |\pi(x, y|x_0^{N-1}, y_0^{N-1}) - P_{XY}(x, y)| \leq \epsilon, \forall (x, y)\}$. That is, the set of sequence pairs for which the empirical frequency is within ϵ of the probability $P_{XY}(x, y)$ for every $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.

For $x_0^{N-1} \in \mathcal{T}_\epsilon^N(X)$ and $P_{Y|X}$, we define the conditional typical sequence $\mathcal{T}_{Y|X}^N(x_0^{N-1}) = \{y_0^{N-1} : (x_0^{N-1}, y_0^{N-1}) \in \mathcal{T}_\epsilon^N(XY)\}$.

The following results will be used:

i For a vector x_0^{N-1}, where x_i is chosen i.i.d. $\sim P_X$,

$$Pr(x_0^{N-1} \in \mathcal{T}_\epsilon^N(X)) \to 1 \text{ as } N \to \infty. \tag{8.1}$$

ii For vectors x_0^{N-1}, y_0^{N-1}, where (x_i, y_i) is chosen i.i.d. $\sim P_{XY}$,

$$Pr((x_0^{N-1}, y_0^{N-1}) \in \mathcal{T}_\epsilon^N(XY)) \to 1 \text{ as } N \to \infty. \tag{8.2}$$

iii For $x_0^{N-1} \in \mathcal{T}_\epsilon^N(X)$, and y_0^{N-1} is independently chosen according to P_Y, then $Pr((x_0^{N-1}, y_0^{N-1}) \in$

$\mathcal{T}^N_{Y|X}(x_0^{N-1})) \in$

$$[2^{-N(I(X;Y)+\lambda)}, 2^{-N(I(X;Y)-\lambda)}], \tag{8.3}$$

for some $\lambda(\epsilon) > 0$ with $\lambda \to 0$ as $\epsilon \to 0$.

iv For any $x_0^{N-1} \in \mathcal{T}^N_\epsilon(X)$, we have $|\mathcal{T}^N_{Y|X}(x_0^{N-1})| \in$

$$[(N+1)^{-|\mathcal{X}||\mathcal{Y}|}2^{NH(Y|X)}, 2^{NH(Y|X)}]. \tag{8.4}$$

Rewriting function being random to achieve full secrecy In this part, we explore one desired property of rewriting function, i.e., it should be stochastic to achieve full secrecy.

For convenience, we write the rewriting function as $y_0^{N-1} = \mathbf{R}(M, x_0^{N-1}, M_1, M_2)$ where M_1 and M_2 are independent of M and x_0^{N-1}, are constant if $\mathbf{R}(\cdot)$ is deterministic, and at least one of them is a random variable otherwise. M_1 and M_2 play significant roles in deriving the rewriting-rate-equivocation region, i.e., whether only M_1, M_2, or both M_1 and M_2 should be random, and how to determine their random values.

In the following, we bound L using M, M_1, M_2 as follows, L

$$
\begin{aligned}
&= I(M; z_0^{N-1}), \\
&= I(M x_0^{N-1} M_1 M_2; z_0^{N-1}) \\
&- I(M_1 M_2 x_0^{N-1}; z_0^{N-1} | M), \\
&= I(y_0^{N-1}; z_0^{N-1}) - I(M_1 M_2 x_0^{N-1}; z_0^{N-1} M), \\
&= I(y_0^{N-1}; z_0^{N-1}) - H(M_1 M_2 x_0^{N-1}) \\
&+ H(M_1 M_2 x_0^{N-1} | z_0^{N-1} M), \\
&= I(y_0^{N-1}; z_0^{N-1}) - H(M_1 M_2) - H(x_0^{N-1}) \\
&+ H(M_1 M_2 x_0^{N-1} | z_0^{N-1} M), \\
&= I(y_0^{N-1}; z_0^{N-1}) - I(y_0^{N-1}; x_0^{N-1}) - H(M_1) \\
&- H(M_2) - H(x_0^{N-1} | y_0^{N-1}) \\
&+ H(M_1 M_2 | M z_0^{N-1}) + H(x_0^{N-1} | M_1 M_2 M z_0^{N-1}), \\
&= NI(Y; Z) - NI(Y; X) - H(M_1) \\
&- H(M_2) - H(x_0^{N-1} | y_0^{N-1}) \\
&+ H(M_1 M_2 | M z_0^{N-1}) + H(x_0^{N-1} | M_1 M_2 M z_0^{N-1}), \\
&\leq NI(Y; Z) - NI(Y; X) - H(M_1) \\
&- H(x_0^{N-1} | y_0^{N-1}) + H(x_0^{N-1} | M_1 M_2 M z_0^{N-1}) \\
&+ H(M_1 M_2 | M z_0^{N-1}),
\end{aligned}
\tag{8.5}
$$

where the third equation is due to $y_0^{N-1} = \mathbf{R}(M, x_0^{N-1}, M_1, M_2)$, and M_1, M_2 and x_0^{N-1} are independent of M; the last equation is due to (y_0^{N-1}, x_0^{N-1}) is i.i.d according to $P_{XY} \in \mathcal{P}(D)$, and the wiretap channel is memoryless.

Therefore, if

$$\frac{1}{N}H(M_1) = I(Y;Z) - I(X;Y) + \sigma_1, \tag{8.6}$$

which implies that the rewriting function $\mathbf{R}(M, x_0^{N-1}, M_1, M_2)$ is random,

$$\frac{1}{N}H(M_1M_2|z_0^{N-1}M) \leq \sigma_2, \tag{8.7}$$

and

$$H(x_0^{N-1}|M_1M_2Mz_0^{N-1}) - H(x_0^{N-1}|y_0^{N-1}) \leq \sigma_3 \tag{8.8}$$

for $\sigma_i \geq 0$ for $i = 1, 2, 3$, the full secrecy is possible.

In the following parts, we present a code construction having all those properties to achieve full secrecy.

Enhanced WEM The achievability of the region $\mathcal{R}'(P_{XY})$ is obtained by designing a specific random code construction for the following enhanced WEM such that the eqution (8.6), and inequtions (8.7) and (8.8) hold.

We define the enhanced WEM (as shown in Figure 8.8) as follows:

Definition 62. $(N, 2^{NR}, 2^{NR_1}, 2^{NR_2}, D)$ code for type one enhanced WEM with a wiretap channel $\mathbb{P} = (\mathcal{Y}, \mathcal{Z}, P_{Y|Z})$ and a rewriting cost function $\varphi(\cdot)$ consists of:

- A *primary* message set $\mathcal{D} = \{0, 1, \cdots, 2^{NR} - 1\}$, a *auxiliary* message set $\mathcal{R}_1 = \{0, 1, \cdots, 2^{NR_1} - 1\}$ and a *random* message set $\mathcal{R}_2 = \{0, 1, \cdots, 2^{NR_2} - 1\}$;

- A *stochastic* rewriting function for Alice: $\mathbf{R}_A : \mathcal{R}_1 \times \mathcal{D} \times \mathcal{X}^N \to \mathcal{Y}^N$ such that $\varphi(x_0^{N-1}, \mathbf{R}_A(M_1, M, x_0^{N-1})) \leq D$ for all $M \in \mathcal{D}, M_1 \in \mathcal{R}_1$ and $x_0^{N-1} \in \mathcal{X}^N$;

- An *auxiliary* function for Alice to determine the random factor in $\mathbf{R}_A$, $f : \mathcal{Y}^N \to \mathcal{R}_2$. And a *deterministic* rewriting function for Alice: $\mathbf{R}'_A : \mathcal{R}_1 \times \mathcal{R}_2 \times \mathcal{D} \times \mathcal{X}^N \to$

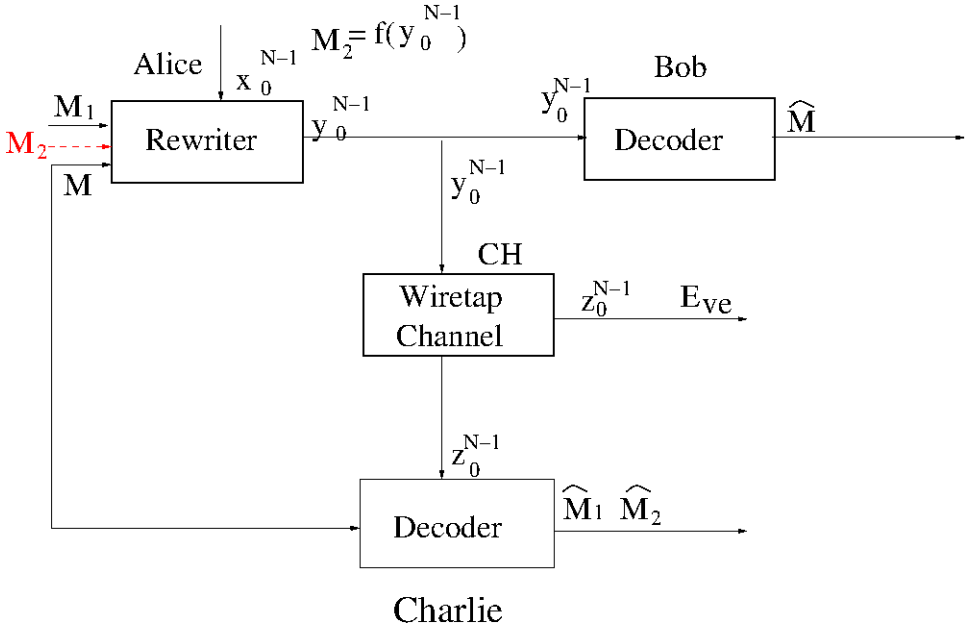


Figure 8.8: Type one enhanced WEM model. CH is the wiretap channel. M, M_1 are messages to rewrite, where M is the primary message, M_1 is the auxiliary message and may not carry information, x_0^{N-1} is the current cell states, y_0^{N-1} is the rewriting codeword, M_2 is the random factor determined by $f(y_0^{N-1})$, z_0^{N-1} is the wiretap channel's output, $\hat{M}_1$, $\hat{M}_2$ and $\hat{M}$ are estimated messages corresponding to M_1, M_2 and M, respectively.

$\mathcal{Y}^N$ such that $\mathbf{R}'_A(M_1, f(\mathbf{R}_A(x_0^{N-1}, M, M_1)), M,\ x_0^{N-1}) = \mathbf{R}_A(x_0^{N-1}, M, M_1)$ for all $M_1 \in \mathcal{R}_1, M \in \mathcal{D}, x_0^{N-1} \in \mathcal{X}^N$;

- A decoding function for Bob: $\mathbf{D}_B : \mathcal{Y}^N \to \mathcal{D}$ such that $\mathbf{D}_B(\mathbf{R}_A(M_1, M, x_0^{N-1})) = M$ for all $M \in \mathcal{D}, M_1 \in \mathcal{R}_1$ and $x_0^{N-1} \in \mathcal{X}^N$;

- A *virtual* decoding function for Charlie: $\mathbf{D}_C : \mathcal{Z}^N \times \mathcal{D} \to \mathcal{R}_1 \times \mathcal{R}_2$.

That is, the original WEM is enhanced by 1) splitting the message set into $\mathcal{D}$ and $\mathcal{R}_1$, and introducing a random variable $M_2 \in \mathcal{R}_2$. Note that $M_1 \in \mathcal{R}_1$ is a dummy message to achieve full secrecy in this part, and carries partial information otherwise (see the following part). That is, we scarify rewriting rate to gain full secrecy. M_2 does not carry any information; 2) for each stochastic rewriting codeword $y_0^{N-1} = \mathbf{R}_A(M_1, M, x_0^{N-1})$, the implicit random variable M_2 can be obtained by the auxiliary function $f(\cdot)$; 3) the same rewriting codeword $y_0^{N-1} = \mathbf{R}_A(M_1, M, x_0^{N-1})$ can also be obtained by the deterministic rewriting function $\mathbf{R}'_A(M_1, M_2, M, x_0^{N-1})$; and 4) introducing a virtual decoder Charlie, who accesses to z_0^{N-1} and the message M, and is to give estimates of M_1 and M_2, $\hat{M}_1$ and $\hat{M}_2$.

The reliability of Charlie is measured by $P_e = Pr((M_1, M_2) \neq (\hat{M}_1, \hat{M}_2))$.

We present a random code construction for the above enhanced WEM and theoretical analysis of the conditions under which the rewriting cost constraint is satisfied and $P_e \to 0$ as $N \to \infty$ as follows, and we illustrate the code construction using Figure 8.9:

Random code construction for type one enhanced WEM

- Codebook generation: Random divide $\mathcal{T}_\epsilon^N(X)$ into $2^{N(R+R_1)}$ bins $\mathcal{B}(M, M_1)$ where $M \in \mathcal{D}$ and $M_1 \in \mathcal{R}_1$. Let $R_2 = H(X) - R - R_1$, and for each codeword in bin $\mathcal{B}(M, M_1)$, index it by $M_2 \in \{0, 1, ..., 2^{NR_2} - 1\}$. Abusing of notation, we index x_0^{N-1} by $\mathcal{B}(M, M_1, M_2)$ or $x_0^{N-1}(M, M_1, M_2)$;

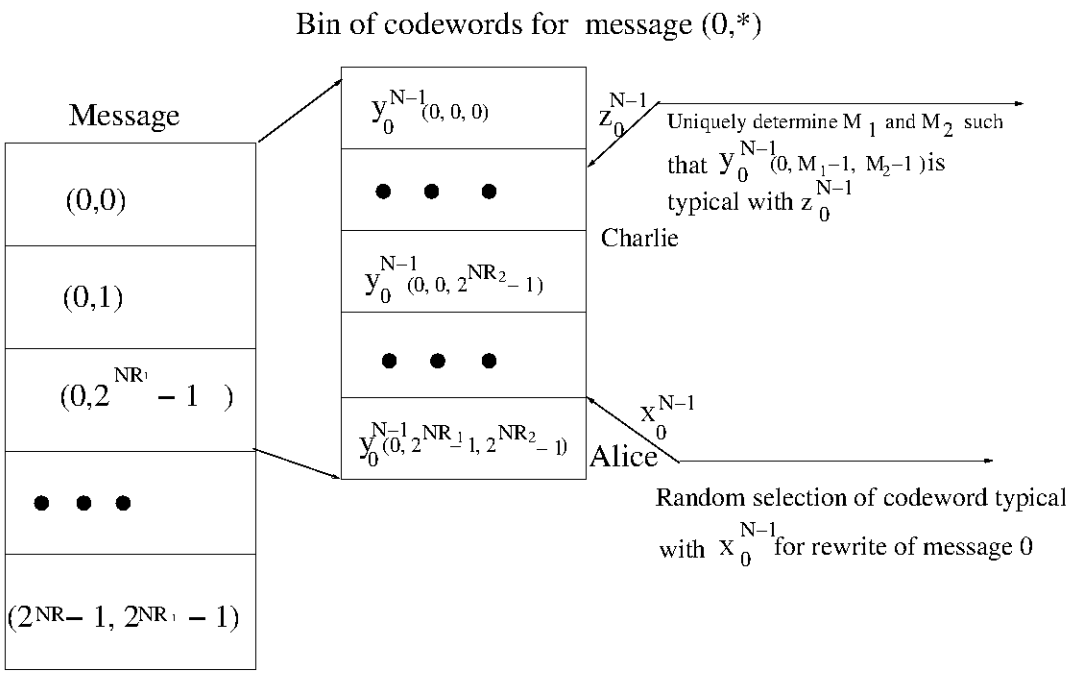


Figure 8.9: Illustration of binning structure, rewriting process for Alice and decoding process for Charlie for type one enhanced secure WEM.

- $\mathbf{R}_A$: given M, a dummy message M_1 and x_0^{N-1}, random choose M_2 such that $y_0^{N-1} = \mathcal{B}(M, M_1, M_2) \in \mathcal{T}^N_{P_{Y|X}}(x_0^{N-1})$ for any M_2;

- f: given the rewriting codeword $y_0^{N-1} = \mathcal{B}(M, M_1, M_2)$, output M_2. $\mathbf{R}'_A$ is to output $\mathcal{B}(M, M_1, M_2)$ with M, M_1, M_2;

- $\mathbf{D}_B$: given y_0^{N-1}, output M such that $y_0^{N-1} = \mathcal{B}(M, M_1, M_2)$ for any M_2;

- $\mathbf{D}_C$: given M, z_0^{N-1}, output a unique $\hat{M}_1, \hat{M}_2$ such that $y_0^{N-1} = \mathcal{B}(M, \hat{M}_1, \hat{M}_2) \in \mathcal{T}^N_{P_{Y|Z}}(z_0^{N-1})$.

Theoretical analysis of the random code construction Clearly, $\mathbf{D}_B$ satisfies the constraint $\mathbf{D}_B(\mathbf{R}_A(M_1, M, x_0^{N-1})) = M$. We next consider the rewriting function and virtual decoding function of Charlie such that the rewriting cost constraint and the reliability requirement of Charlie are satisfied.

Firstly, let us consider the probability of rewriting failure, i.e., Pr(no $y_0^{N-1} \in \mathcal{B}(M, M_1)$ such that $y_0^{N-1} \in \mathcal{T}^N_{P_{Y|X}}(x_0^{N-1})$)

$$\begin{aligned} &= (1 - \frac{1}{2^{N(R+R_1)}})^{|\mathcal{T}^N_{P_{Y|X}}(x_0^{N-1})|}, \\ &= (1 - \frac{1}{2^{N(R+R_1)}})^{2^{N(R+R_1)}|\mathcal{T}^N_{P_{Y|X}}(x_0^{N-1})|2^{-N(R+R_1)}}, \\ &\leq e^{-(2^{NH(Y|X)-N(R+R_1)})}, \end{aligned} \tag{8.9}$$

where inequation (8.9) is based on the property (8.4). Therefore, if $R + R_1 \leq H(Y|X)$, the above probability tends to be 0 and we have a desired y_0^{N-1}. We further know that $R_2 \geq I(X;Y)$ since $R_2 = H(X) - R - R_1$.

Secondly, we analyze the condition under which the average error probability $E(P_e) = E(Pr(M_1, M_2) \neq (\hat{M}_1, \hat{M}_2)) = Pr((M_1, M_2) = (j, k))E(Pr((\hat{M}_1, \hat{M}_2) \neq (j, k)|(M_1, M_2) = (j, k)))$ tends to be 0 as $N \to 0$.

By symmetry of the code construction, the average error probability does not depend on (M_1, M_2), thus we assume $(M_1, M_2) = (1, 1)$. Further, without less of generality, we assume that $M = 1$.

Define the following events: $\mathcal{E}_{1,1} \stackrel{def}{=}$

$$\{(y_0^{N-1}, z_0^{N-1}) \in \mathcal{T}_\epsilon^N(YZ) \text{ and } y_0^{N-1} = \mathcal{B}(1,1,1)\},$$

and $\mathcal{F}_{j,k} \stackrel{def}{=}$

$$\{(y_0^{N-1}, z_0^{N-1}) \in \mathcal{T}_\epsilon^N(YZ) \text{ and } y_0^{N-1} \in \mathcal{B}(1,j,k)\}.$$

By the union bound, $E(Pr((\hat{M}_1, \hat{M}_2) \neq (1,1)|(M_1, M_2) = (1,1))$

$$\begin{aligned}
&\leq Pr(\mathcal{E}_{1,1}^c) + \bigcup_{(j,k)\neq(1,1)} Pr(\mathcal{F}_{j,k}), \\
&\leq \sum_{j,k} Pr((y_0^{N-1}, z_0^{N-1}) \in \mathcal{T}_\epsilon^N(YZ)|y_0^{N-1} \\
&= \mathcal{B}(1,j,k)) + \epsilon', \qquad (8.10) \\
&\leq 2^{N(R_1+R_2-I(Y;Z)+\lambda)} + \epsilon', \qquad (8.11)
\end{aligned}$$

where inequation (8.10) is based on the property (8.1), and inequation (8.11) is based on the property (8.3).

Therefore, for our enhanced WEM codes when $R_1 + R_2 \leq I(Y;Z)$, that is $R_1 = I(Y;Z) - I(X;Y) + \sigma_1$, $E(Pr((\hat{M}_1, \hat{M}_2) \neq (1,1)|(M_1, M_2) = (1,1))) \leq \epsilon$.

Finally, let us back to analysis of eqution (8.6), and inequtions (8.7) and (8.8). Based on the above analysis, we know that $R \leq H(Y|Z) + \sigma$ and $R_1 = I(Y;Z) - I(X;Y) + \sigma_1$. Based on Fano's inequality[16, lemma 7.9.1], we obtain that $\frac{1}{N}H(M_1M_2|z_0^{N-1}M) \leq \frac{1}{N} + Pr((\hat{M}_1, \hat{M}_2) \neq (M_1, M_2))(R_1 + R_2) \leq \sigma_2$. Based on our code construction, y_0^{N-1} is uniquely determined by M, M_1, M_2, therefore $H(x_0^{N-1}|MM_1M_2z_0^{N-1}) = H(x_0^{N-1}|y_0^{N-1}z_0^{N-1}) \leq$

$H(x_0^{N-1}|y_0^{N-1}) + \sigma_3$. That is, $\frac{1}{N}L \leq \sigma_1 + \sigma_2 + \sigma_3$ based on inequation (8.5). Therefore, (R, R) is achievable for $R \leq H(Y|Z)$.

8.4.1.2 Step 2: Achieving the entire type one region $\mathcal{R}(P_{XY})$

The key idea is to modify step 1 such that we let the dummy message M_1 transmit additional information.

The code construction is modified as follows,

- $\mathbf{D}_B$: given y_0^{N-1}, output M and M_1 such that $y_0^{N-1} = \mathcal{B}(M, M_1, M_2)$ for any M_2.

The remaining parts are the same as step 1.

The analysis of the above code construction is as follows.

By checking the analysis for rewriting cost constraint of step 1, we know that as long as $R + R_1 \leq H(Y|X)$, there exists a codeword satisfying the rewriting cost constraint.

Next, consider the equivocation rate:

$$\begin{aligned}\frac{1}{N}H(MM_1|z_0^{N-1}) &\geq \frac{1}{N}H(M|z_0^{N-1}),\\ &= \frac{1}{N}H(M) - \frac{1}{N}I(M; z_0^{N-1}).\end{aligned}$$

With similar techniques to step 1, i.e. $I(M; z_0^{N-1}) \leq \sigma$, we can prove that $\frac{1}{N}H(MM_1|z_0^{N-1}) \geq R - \sigma$. Thus, we obtain that $(R + R_1, R - \sigma)$ is achievable, where $R + R_1 \leq H(Y|X)$ and $R \leq H(Y|Z)$.

8.4.1.3 Step 3: $\mathcal{R}^{swem}$ is convex

We show that $\mathcal{R}^{swem}$ is convex by proving that, for any $P_{X_1Y_1}, P_{X_2Y_2} \in \mathcal{P}(D)$, the convex hull of $\mathcal{R}(P_{X_1Y_1})$ and $\mathcal{R}(P_{X_2Y_2})$ is in $\mathcal{R}^{swem}$.

Let $(R_1, R_{e1}) \in \mathcal{R}(P_{X_1Y_1})$ for some random variables X_1, Y_1 and Z_1 whose joint distribution is such that $\forall(x, y, z) \in \mathcal{X}\times\mathcal{Y}\times\mathcal{Z}$, $P_{X_1Y_1Z_1}(x, y, z) = P_{X_1}(x)P_{Y_1|X_1}(y|x)P_{Z|Y}(z|y)$.

Similarly, let $(R_2, R_{e2}) \in \mathcal{R}(P_{X_2Y_2})$ for some random variables X_2, Y_2 and Z_2 whose joint distribution is such that $\forall (x,y,z) \in \mathcal{X}\times\mathcal{Y}\times\mathcal{Z}$, $P_{X_2Y_2Z_2}(x,y,z) = P_{X_2}(x)P_{Y_2|X_2}(y|x)P_{Z|Y}(z|y)$.

Let

$$\theta = \begin{cases} 1 & \text{with probability } \lambda, \\ 2 & \text{with probability } 1-\lambda, \end{cases}$$

thus we know that $\theta \to X_\theta \to Y_\theta \to Z_\theta$ forms a Markov chain and the joint distribution of X_θ, Y_θ and Z_θ satisfies $\forall (x,y,z) \in \mathcal{X}\times\mathcal{Y}\times\mathcal{Z}$, $P_{X_\theta Y_\theta Z_\theta}(x,y,z) = P_{X_\theta}(x)P_{Y_\theta|X_\theta}(y|x)P_{Z|Y}(z|y)$ and $P_{X_\theta Y_\theta} \in \mathcal{P}(D)$. Let $X = X_\theta$, $Y = Y_\theta$ and $Z = Z_\theta$. Then

$$\begin{aligned} H(Y|X) &= H(Y_\theta|X_\theta), \\ &\geq H(Y_\theta|X_\theta,\theta), \\ &= \lambda H(Y_1|X_1) + (1-\lambda)H(Y_2|X_2), \\ &= \lambda R_1 + (1-\lambda)R_2, . \end{aligned}$$

Similarly, we can prove that $H(Y|Z) \geq \lambda R_{e1} + (1-\lambda)R_{e2}$. Hence, for any $\lambda \in [0,1]$, there exist X, Y such that $(\lambda R_1 + (1-\lambda)R_2, \lambda R_{e1} + (1-\lambda)R_{e2}) \in \mathcal{R}(P_{XY}) \subseteq \mathcal{R}^{swemn}$, which finishes the proof.

8.4.1.4 Sketch proof of achieving the entire region $\mathcal{R}(P_{XY})$ for type two region

In this case, we rewrite equation (8.5) as follows: L

$$
\begin{aligned}
&= I(M; z_0^{N-1}), \\
&= I(y_0^{N-1}; z_0^{N-1}) - I(y_0^{N-1}; x_0^{N-1}) - H(M_1) \\
&- H(M_2) - H(x_0^{N-1}|y_0^{N-1}) \\
&+ H(M_1M_2|Mz_0^{N-1}) + H(x_0^{N-1}|M_1M_2Mz_0^{N-1}), \\
&\leq NI(Y;Z) - NI(X;Y) + H(M_1M_2|Mz_0^{N-1}) \\
&+ H(x_0^{N-1}|M_1M_2Mz_0^{N-1}) - H(x_0^{N-1}|y_0^{N-1}), \\
&\leq NI(Y;Z) - NI(X;Y) + H(M_1) \\
&+ H(M_2|MM_1z_0^{N-1}) \\
&+ H(x_0^{N-1}|MM_1M_2z_0^{N-1}) - H(x_0^{N-1}|y_0^{N-1}),
\end{aligned}
$$

where some repeated steps of equation (8.5) are skipped.

Therefore, if $\frac{1}{N}H(M_1) = I(Y;X) - I(Y;Z) + \sigma_1$, $\frac{1}{N}H(M_2|z_0^{N-1}MM_1) \leq \sigma_2$, and $H(x_0^{N-1}|M_1M_2Mz_0^{N-1}) - H(x_0^{N-1}|y_0^{N-1}) \leq \sigma_3$ for $\sigma_i \geq 0$ for $i = 1, 2, 3$, the full secrecy is possible.

This motivates us to redefine the enhanced WEM (see Figure 8.10) as follows:

Definition 63. $(N, 2^{NR}, 2^{NR_1}, 2^{NR_2}, D)$ code for type two enhanced WEM with wiretap channel $\mathbb{P} = (\mathcal{Y}, \mathcal{Z}, P_{Y|Z})$ and the rewriting cost function $\varphi(\cdot)$ consists of:

- A *primary* message set $\mathcal{D} = \{0, 1, \cdots, 2^{NR} - 1\}$, an *auxiliary* random message set $\mathcal{R}_1 = \{0, 1, \cdots, 2^{NR_1} - 1\}$ and a *primary* random message set $\mathcal{R}_2 = \{0, 1, \cdots, 2^{NR_2} - 1\}$;

- A *stochastic* rewriting function for Alice: $\mathbf{R}_A : \mathcal{D} \times \mathcal{X}^N \to \mathcal{Y}^N$ such that $\varphi(x_0^{N-1}, \mathbf{R}_A(M, x_0^{N-1})) \leq$

D for all $M \in \mathcal{D}$ and $x_0^{N-1} \in \mathcal{X}^N$;

- An *auxiliary* function for Alice to determine the random factor in $\mathbf{R}_A$, $f : \mathcal{Y}^N \to \mathcal{R}_1 \times \mathcal{R}_2$. And a *deterministic* rewriting function for Alice: $\mathbf{R}'_A : \mathcal{R}_1 \times \mathcal{R}_2 \times \mathcal{D} \times \mathcal{X}^N \to \mathcal{Y}^N$ such that $\mathbf{R}'_A(f(\mathbf{R}_A(x_0^{N-1}, M)), M,\ x_0^{N-1}) = \mathbf{R}_A(x_0^{N-1}, M)$ for all $M \in \mathcal{D}$ and $x_0^{N-1} \in \mathcal{X}^N$;

- A decoding function for Bob: $\mathbf{D}_B : \mathcal{Y}^N \to \mathcal{D}$ such that $\mathbf{D}_B(\mathbf{R}_A(M, x_0^{N-1})) = M$ for all $M \in \mathcal{D}$ and $x_0^{N-1} \in \mathcal{X}^N$;

- A *virtual* decoding function for Charlie: $\mathbf{D}_C : \mathcal{Z}^N \times \mathcal{D} \to \mathcal{R}_2 \times \mathcal{R}_1$.

That is, compared with type one enhanced WEM, we let M_1 be the random variable instead of auxiliary message variable. The reliability for Charlie is measured by $P_e = Pr(M_1 \neq \hat{M}_1)$.

The remaining proof details are similar to those of previous subsections, and we omit them.

8.4.2 Proof of the converse part

The proof for R is the same as that of [2], and for completeness, we present it here. We first digress to prove the following conclusion:

$$NR = H(y_0^{N-1}|x_0^{N-1}). \tag{8.12}$$

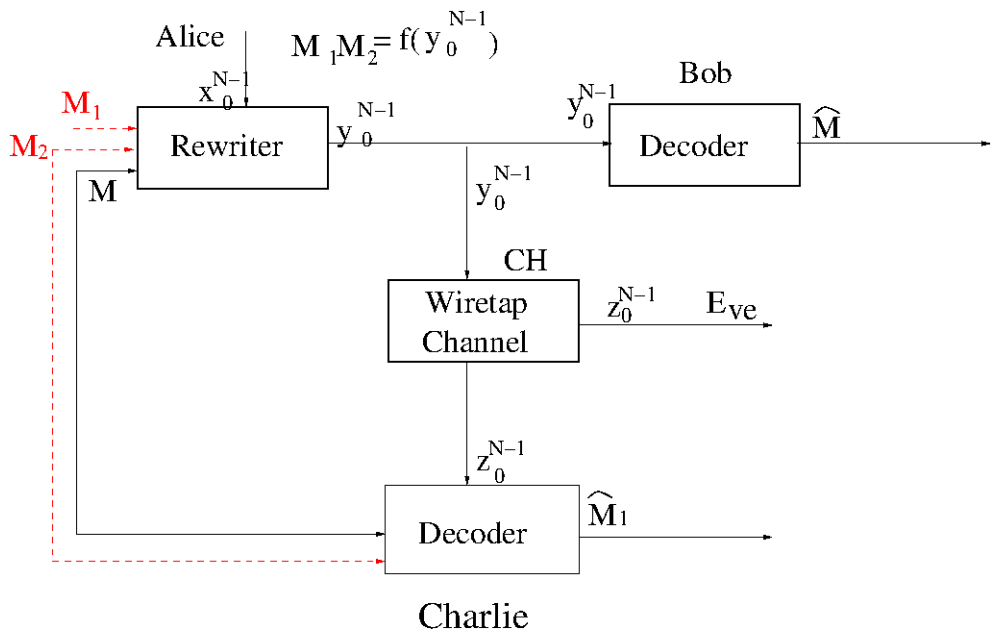


Figure 8.10: Type two enhanced WEM model. CH is the wiretap channel. M are messages to rewrite, x_0^{N-1} is the current cell states, y_0^{N-1} is the rewrite codeword, M_1 and M_2 are two random variables determined by $f(y_0^{N-1})$, z_0^{N-1} is the wiretap channel's output, $\hat{M}_1$ and $\hat{M}$ are estimated messages corresponding to M_1 and M, respectively.

NR

$$
\begin{aligned}
&= H(M), && (8.13)\\
&= H(M|x_0^{N-1}), && (8.14)\\
&= H(Mx_0^{N-1}|x_0^{N-1}), && (8.15)\\
&\geq H(y_0^{N-1}|x_0^{N-1}), && (8.16)\\
&\geq H(M|x_0^{N-1}), && (8.17)\\
&= NR,
\end{aligned}
$$

where

(8.13) follows from the assumption that M is uniformly distributed among $\mathcal{D}$;

(8.14) follows from the fact that M is independent of x_0^{N-1};

(8.16) follows from $y_0^{N-1} = \mathbf{R}(M, x_0^{N-1})$ and the fact that function never increases entropy;

(8.17) follows from $M = \mathbf{D}(y_0^{N-1})$.

Next, we proceed the proof as follows: $R = \frac{1}{N}H(y_0^{N-1}|x_0^{N-1}) \leq \frac{1}{N}\sum_{i=0}^{N-1} H(y_i|x_i) \leq H(Y|X)$.

Then, we consider the rewriting cost, $\varphi(x_0^{N-1}, y_0^{N-1}) = \frac{1}{N}\sum_{i=0}^{N-1} \varphi(x_i, y_i) = E(\varphi(X,Y)) \leq D$, thus $P_{XY} \in \mathcal{P}(D) = \{P_{XY} : P_X = P_Y, E(\varphi(X,Y)) \leq D\}$, where the fact that $P_X = P_Y$ follows from the assumption that stationary distribution of x_0^{N-1} exists. Therefore, $R \leq H(Y|X)$ for $P_{XY} \in \mathcal{P}(D)$.

Let us consider $R_e \leq \frac{1}{N}H(M|z_0^{N-1}) \leq \frac{1}{N}H(y_0^{N-1}|z_0^{N-1}) \leq \sum_{i=0}^{N-1} H(y_i|z_i) \leq H(Y|Z)$.

Meanwhile, we know that $R_e \leq \frac{1}{N}H(M|z_0^{N-1}) \leq \frac{1}{N}H(M) = R$, where the last inequality is based on the conclusion just obtained for $H(M)$. Therefore, $R_e \leq \min\{R, H(Y|Z)\}$.

9. POLAR CODES FOR SECURE WRITE-EFFICIENT MEMORIES

9.1 Introduction

In this section, we present the background of secure Write-Efficient Memories including their motivations and formal definitions, brief obtained information theory results in [47], and show our contributions.

9.1.1 Motivation of secure write-efficient memories

Flash memories are significant non-volatile memory techniques. The smallest unit of flash memory is a cell, which contains a control gate, a floating gate and so on. Data is represented by the number of electrons trapped in the floating gate. There are three basic operations on a cell, program, i.e., to eject electrons into the floating gate, read, i.e., to measure the number of electrons in the floating gate, and erase, i.e., to remove electrons from the floating gate. Each flash chip is composed of blocks, each block consists of pages, and a page is made up of cells. Similarly, there are three operations for a block, i.e., program, read and erase, however, the unit of programming and reading is a page, and the unit of erasing is a block.

There are two challenges in flash memories, one is the well-known *endurance* problem and the other one is the less well-known *insecure deletion* problem. The endurance problem means flash memory can only experience a limited number of program/erase cycles after which its reliability can not be guaranteed. The current code solution for endurance is the rewriting codes, e.g., Write-Once Memories [61], and Write-Efficient Memories (WEM) [2], etc. Recently there is a large amount of work for rewriting code [9, 48, 30] showing the existance of optimal constructions for them and system work [79, 55, 78] showing various benefits rewriting code bringing to flash memories.

Insecure deletion means *Flash Translation Layer* (FTL) produces multiple copies of

data that can not be deleted completely as they are either impossible or costly, however, a sophisticated attacker can recover and obtain information about the data.

We illustrate the insecure deletion in detail here. The first reason causing this is the existence of multiple copies of codewords in flash memories. Flash memories are not perfect as there are various errors, thus a strong error correcting code is used to combat errors. *Memory scrubbing* is also used to protect flash memories, which is to correct a noisy codeword and write a new error-free codeword back to memories. However, due to the *out-of-place* rewriting policy, the updated codeword is stored at a new physical address and the original codeword remains in memories. Those mechanisms lead to multiple copies of codewords existing in memories. Other reasons causing this are weal leveling and garbage collection. A recent study by Desnoyers [21] theoretically estimates that on overage $3 \sim 13$ copies of codewords can be generated for one write issued by a user, and the exact number depends on the work load traffic and various algorithms (e.g., garbage collection algorithms) used.

For current flash memory solutions, when to delete data, it is either impossible or costly to delete all copies of codewords corresponding to the data due to the imperfections of the physical erasure process and the FTL [39, 70]. However, when the flash memory is attacked by an eavesdropper, (who is able to trace all copies of codewords corresponding to the same data, and is aware of all encoding and decoding algorithms, thus leading to much stronger decoding ability than the decoder having access to a single codeword [50]), the sensitive information can be leaked. Unfortunately, there is barely no coding solution to solve the insecure deletion for flash memories.

In a recent paper [47], a new coding scheme was proposed, *Secure Write-Efficient Memories*. The significance of secure WEM is two-fold, on the practical side it is the first coding model combating both the endurance and the insecure deletion; on the theoretical side it extends the current research scope of rewriting codes in a similar way as wiretap

channel coding [73] extends the channel coding model.

9.1.2 Definition of secure WEM

In the secure WEM setting (shown in Figure 9.1), Alice wishes to store messages in a limited lifetime storage medium using a rewriting code, WEM [2], the messages are accessible to Bob through a storage channel, which is assumed noiseless for simplicity, but her transmissions also reach an eavesdropper Eve through a wiretap channel. Alternatively, let M be the message that Alice wishes to store. Based on the message M and the current N cell state vector x_0^{N-1}, the rewriter maps M to an N-bit codeword y_0^{N-1}. This codeword is transmitted through the noiseless storage channel and the wiretap channel resulting y_0^{N-1} and z_0^{N-1}. The decoder estimates y_0^{N-1} to recover the message M.

The goal of secure WEM codes is to design a rewriting coding scheme such that it is possible to store messages cost-effectively and securely. Being cost-effective means for each rewrite the defined rewriting cost, i.e., which is measured by $\varphi(x_0^{N-1}, y_0^{N-1})$ for a defined cost $\varphi(\cdot)$, has to be less than a predefined number to solve the endurance problem. Being secure means the uncertainty of the eavesdropper about the message M after observing the wiretap channel output z_0^{N-1}, i.e., which is measured by $\frac{1}{N}H(M|z_0^{N-1})$ [73], also satisfies a predefined constraint to solve the insecure deletion problem.

The following notations will be used to define secure WEM. For Alice and Bob, let $\mathcal{X}$ be the alphabet of the symbols stored in a cell, and $\mathcal{Z}$ be that for Eve. $\forall x, y \in \mathcal{X}$, let the *rewriting cost* of changing a cell's level from x to y be $\varphi(x, y)$, which may be time or energy taken. Given N cells and $x_0^{N-1}, y_0^{N-1} \in \mathcal{X}^N$, let $\varphi(x_0^{N-1}, y_0^{N-1}) = \frac{1}{N}\sum_{i=0}^{N-1}\varphi(x_i, y_i)$ be the *average rewriting cost* of changing the N cell levels from x_0^{N-1} to y_0^{N-1}.

Let $\mathcal{D} \subseteq \mathbb{N}$ and it denotes the $|\mathcal{D}|$ possible values of the data stored in the N cells. Let the *decoding function* be $\mathbf{D} : \mathcal{X}^N \to \mathcal{D}$, which maps the N cells' levels to the data they represent. Let the *rewriting function* be $\mathbf{R} : \mathcal{X}^N \times \mathcal{D} \to \mathcal{X}^N$, which changes the N cells'

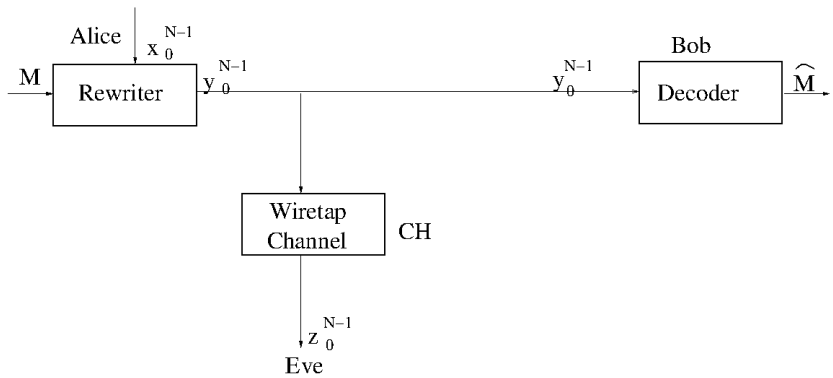


Figure 9.1: The secure WEM model. CH is the wiretap channel. $M, x_0^{N-1}, y_0^{N-1}, z_0^{N-1}$ and $\hat{M}$ are the message to rewrite, the current cell states, the rewrite codeword, the wiretap channel's output and the estimated message, respectively.

levels to represent the new input data.

We present the definition of secure WEM codes in the following.

Definition 64. An $(N, 2^{NR}, R_e, D)$ secure write-efficient memory code with a wiretap channel $\mathbb{P} = (\mathcal{X}, \mathcal{Z}, P_{Z|X})$ and the rewriting cost function $\varphi(\cdot)$ consists of

- $\mathcal{D} = \{0, 1, \cdots, 2^{NR} - 1\}$ and its corresponding codewords $\bigcup_{i=0}^{2^{NR}-1} \mathcal{C}_i$, where $\mathcal{C}_i \subseteq \mathcal{X}^N$ is the set of codewords representing data i. We require $\forall i \neq j, \mathcal{C}_i \bigcap \mathcal{C}_j = \emptyset$;
- $\mathbf{R}(M, x_0^{N-1})$ such that
  - $\varphi(x_0^{N-1}, \mathbf{R}(M, x_0^{N-1})) \leq D$ for any $M \in \mathcal{D}$ and $x_0^{N-1} \in \mathcal{X}^N$;
  - $\frac{1}{N} H(M|z_0^{N-1}) \geq R_e - \epsilon$ for any $M \in \mathcal{D}, z_0^{N-1} \in \mathcal{Z}^n$, $\epsilon > 0$ as $N \to \infty$.
- $\mathbf{D}(y_0^{N-1})$ such that $\mathbf{D}(\mathbf{R}(x_0^{N-1}, M)) = M$ for all $M \in \mathcal{D}$ and $x_0^{N-1} \in \mathcal{X}^N$.

That is, the first condition indicates that each data is represented by a group of codewords, the first requirement of the rewriting function indicates that during each rewrite

the average rewriting cost between the current codeword x_0^{N-1} and the updated codeword y_0^{N-1} is less than a predefined number, the second requirement of the rewriting function indicates that the leaked information of the message at the eavesdropper is limited, and the last one indicates that the decoder knows the rewritting message given a rewriting codeword.

9.1.3 Main results of secure WEM [47]

Previous work of [47] introduces us the model of secure WEM, and presents us some information theory results, for which we recap in the following.

The following notations will be used. Let $\mathcal{P}(\mathcal{X} \times \mathcal{X})$ be the set of joint probability distributions over $\mathcal{X} \times \mathcal{X}$. For a pair of random variables $(X, Y) \in (\mathcal{X}, \mathcal{X})$, let $P_{XY}, P_X, P_{X|Y}$ denote the joint probability distribution, the marginal distribution, and the conditional probability distribution, respectively. $E(\cdot)$ denotes the expectation operator. If X is uniformly distributed over $\{0, 1, \cdots, q-1\}$, denote it by $X \sim U(q)$.

Fixed D, $\varphi(\cdot)$ and $\mathbb{P} = (\mathcal{X}, \mathcal{Z}, P_{Z|X})$, $(R, R_e) \in \mathbb{R}^2$ is *achievable* if there exists an $(N, 2^{NR}, R_e, D)$ codes. The set of all achievable tuples is denoted by $\mathcal{R}^{swem}$, *rewriting-rate-equivocation region*. The *secrecy rewriting capacity* is $C^{swem}(D) \stackrel{def}{=} \sup_R\{R : (R, R) \in \mathcal{R}^{swem}\}$, i.e., the maximal R such that (R, R) is achievable.

The $\mathcal{R}^{swem}$ was obtained in [47] and shown in the following theorem:

Theorem [47] 65. Define $\mathcal{R}(P_{XY}) =$

$$\{(R, R_e) : \begin{array}{l} R \leq H(Y|X) \\ R_e \leq H(Y|Z) \\ R_e \leq R \end{array} \},$$

where $P_{XY} \in \mathcal{P}(D) \stackrel{def}{=} \{P_{XY} : P_X = P_Y, E(\varphi(X, Y)) \leq D\}$, the joint distribution of X, Y, Z factorizes as $P_X P_{Y|X} P_{Z|Y}$, and the $P_{Z|Y}$ is given by $\mathbb{P} = (\mathcal{X}, \mathcal{Z}, P_{Z|Y})$. Then

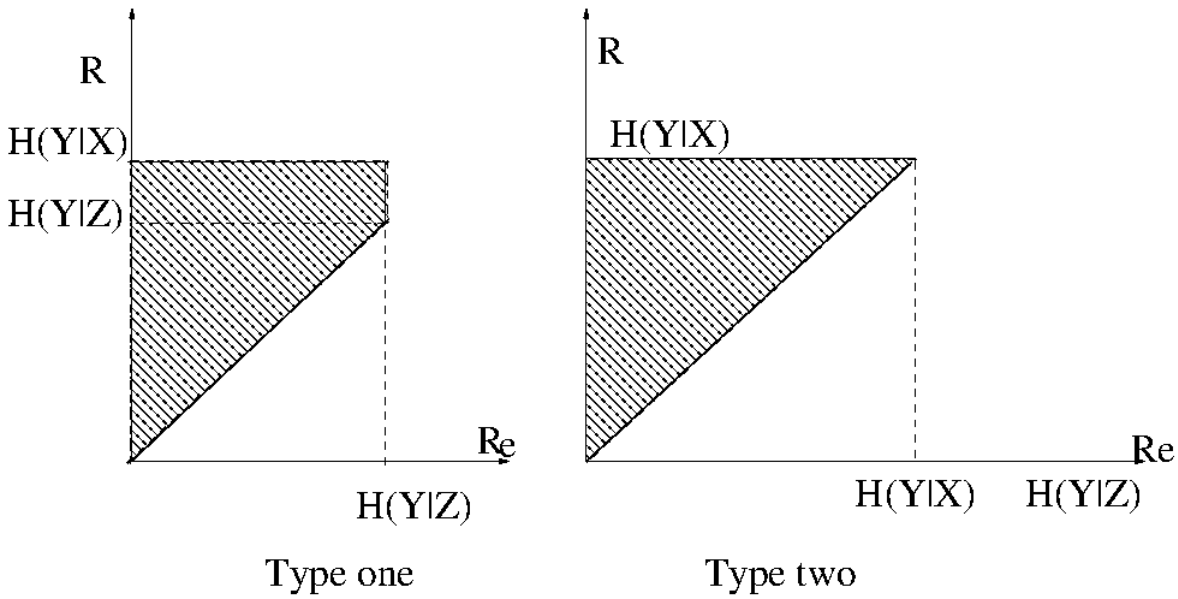


Figure 9.2: Typical shape of $\mathcal{R}(P_{XY})$.

$\mathcal{R}^{swem} = \bigcup_{P_{XY}} \mathcal{R}(P_{XY})$.

The typical shapes of the above achievable region $\mathcal{R}(P_{XY})$ are presented in Figure 9.2: type one is the case where $H(Y|Z) \leq H(Y|X)$ for a given $P_{XY} \in \mathcal{P}(D)$, and type two is the other case.

By specializing Theorem 65 to the case $R = R_e$, we obtain the following result for secrecy rewriting capacity.

Corollary 66. The secrecy rewriting capacity of secure WEM $(N, 2^{NR}, R_e, D)$ code with wiretap channel $\mathbb{P} = (\mathcal{X}, \mathcal{Z}, P_{Z|Y})$ and the rewriting cost function $\varphi(\cdot)$ is:

$$C^{swem}(D) = \max_{P_{XY} \in \mathcal{P}(D)} \{\min\{H(Y|X), H(Y|Z)\}\},$$

where the definition of $\mathcal{P}(D)$ is the same as above.

9.1.4 Contribution and structure of this paper

In this paper, we present an optimal (i.e., achieve the whole rewriting-rate-equivocation region) code construction based on polar codes for secure WEM for a large family of secure WEM. The remaining of this paper is structured as follows: in Section 9.2, we present a brief introduction of polar codes and some useful terms; in Section 9.3, we present a polar code construction for secure WEM, which achieves the whole region of secure WEM; the conclusion and possible future work are obtained in Section 9.4.

9.2 Polar Code Terms and Notations

Polar code [5] was invented by Arikan in 2008, and it is the first theoretically proven capacity approaching code for symmetric channels. Polar code is being hailed as a milestone in coding theory not only for its great success in channel coding but also for its remarkable performances in lossy source coding [42] , wiretap channel coding [4, 19, 53], write-once memories [9], etc. In this part, we present a brief introduction to polar codes so that some terms can be understood later.

Let $W = (\mathcal{X}, \mathcal{Y}, W_{Y|X})$ be a binary-input discrete memoryless channel. Let $G_2^{\otimes n}$ be n-th Kronecker product of $\left(\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right)$. Let $Z(W) = \sum_{y \in \mathcal{Y}} \sqrt{W_{Y|X}(y|0) W_{Y|X}(y|1)}$ be the Bhattacharyya parameter.

Let $N = 2^n$, and the polar code $C_N(F, u_F)$ is $\{x_0^{N-1} = u_0^{N-1} G_2^{\otimes n} : u_{F^c} \in \{0,1\}^{|F^c|}\}$, where $\forall F \subseteq \{0, 1, \cdots, N-1\}$, u_F is the subvector $u_i : i \in F$, and $u_F \in \{0,1\}^{|F|}$. By convention, F is the frozen set and u_F is the frozen set value.

Denote $W_N^{(i)} : \{0,1\} \to \mathcal{Y}^N \times \{0,1\}^i$ the i-th sub-channel with input set $\{0,1\}$, output set $\mathcal{Y}^N \times \{0,1\}^i$, and the transition probability $W_N^{(i)}(y_0^{N-1}, u_0^{i-1}|u_i) \stackrel{def}{=} \frac{1}{2^{N-1}} \sum_{u_{i+1}^{N-1}} W^N(y_0^{N-1}|u_0^{N-1})$, where $W^N(y_0^{N-1}|u_0^{N-1})$ is $\prod_{i=0}^{N-1} W_{Y|X}(y_i|(u_0^{N-1} G_2^{\otimes n})_i)$, and $(u_0^{N-1} G_2^{\otimes n})_i$ denotes the i-th element of $u_0^{N-1} G_2^{\otimes n}$.

Let $\beta < 1/2$ be a fixed positive constant, define a good sub-channel set as $\mathcal{G}_N(W,\beta) = \{i \in \{0,1,\cdots,N-1\} : I(W_N^{(i)}) > \frac{1}{N}2^{-N^\beta}\}$, and define a bad sub-channel set as $\mathcal{B}_N(W,\beta) = \{i \in \{0,1,\cdots,N-1\} : I(W_N^{(i)}) \leq \frac{1}{N}2^{-N^\beta}\}$. By abusing notations, we also define a good sub-channel set as $\mathcal{G}'_N(W,\beta) = \{i \in \{0,1,\cdots,N-1\} : Z(W_N^{(i)}) < 1-(\frac{1}{N}2^{-N^\beta})^2\}$ and define a bad sub-channel set as $\mathcal{B}'_N(W,\beta) = \{i \in \{0,1,\cdots,N-1\} : Z(W_N^{(i)}) \geq 1-(\frac{1}{N}2^{-N^\beta})^2\}$.

Based on [43, Lemma 2.6], $\lim_{N\to\infty} \frac{1}{N}|\mathcal{B}_N(W,\beta)| = \lim_{N\to\infty} \frac{1}{N}|\mathcal{B}'_N(W,\beta)| = 1 - I(W)$, and $\lim_{N\to\infty} \frac{1}{N}|\mathcal{G}_N(W,\beta)| = \lim_{N\to\infty} \frac{1}{N}|\mathcal{G}'_N(W,\beta)| = I(W)$.

9.3 Optimal Code Construction

In this section, we present a polar code construction for a special case of secure WEM and prove that the code construction achieves the whole achievable region. Due to space limitation, we only present the code constructions for type one rewriting-rate-equivocation region of secure WEM.

9.3.1 Symmetric secure WEM

In this subsection, we define symmetric secure WEM, which is a large family of secure WEM, and it is the symmetric secure WEM that our polar code construction is focusing in this paper.

Recall that the rewriting capacity of WEM is $\mathcal{R}(D) = \max_{P_{XY}\in\mathcal{P}(D)} H(Y|X)$ [2]. Analogous to a symmetric channel, a symmetric WEM is such a WEM that its rewriting capacity is achieved when current cell state alphabet (i.e., X) and updated cell state alphabet (i.e., Y) are uniformly distributed. That is, for symmetric WEM its capacity is determined as $\mathcal{R}(D) = \max_{P_{XY}\in\mathcal{P}^s(D)} H(Y|X)$, where $\mathcal{P}^s(D) \stackrel{def}{=} \{P_{XY} : P_X = P_Y, X \sim U(q), E(\varphi(X,Y)) \leq D\}$ and q is the number of states for X.

For a P_{XY} achieving rewriting capacity of a symmetric WEM, it induces a channel $\mathbb{W} = (X, Y, W_{Y|X})$, and we term it WEM *channel*. A symmetric secure WEM is such a

secure WEM model that both the WEM and the wiretap channel are symmetric. Further, we consider the case where the WEM channel is *stochastically degraded* with respect to the wiretap channel, i.e., the type one rewriting-rate-equivocation region of secure WEM. Besides, the code construction presented here focuses on symmetric rewriting cost, i.e., $\varphi(0,1) = \varphi(1,0)$, the Hamming distrance metric.

We present a concrete example of Symmetric secure WEM we are considering in the following:

Example 67. Let the rewriting cost metric be the Hamming distance metric, i.e., $\varphi(0,1) = \varphi(1,0) = 1$ and $\varphi(0,0) = \varphi(1,1) = 0$, in this case the capacity of symmetric WEM is $H(D)$ where $0 \leq D \leq 1/2$ and the WEM channel induced is a Binary Symmetric Channel (BSC) with parameter D. Let the wiretap channel $\mathbb{P} = (\mathcal{X}, \mathcal{Z}, P_{Z|Y})$ be a BSC with flipping rate p ($0 \leq p \leq 1/2$). In this case, the secrecy capacity is $H(p)$ based on Corollary 66. When $D > p$, the WEM channel stochastically degrades with respect to the wiretap channel, and it is one example of symmetric secure WEMs we are focusing in this work.

9.3.2 Code construction

The outline of the code construction is presented in Figure 9.3: Given the WEM channel and the wiretap channel, we divide all sub-channels to three parts, i.e., sub-channels bad for both channels, the sub-channel index set is denoted as set $\mathcal{M} \subseteq \mathbb{N}$, sub-channels good for both channels, the sub-channel index set is denoted as set $\mathcal{M}_2 \subseteq \mathbb{N}$, and remaining sub-channels, the sub-channel index set is denoted as the set $\mathcal{M}_1 \subseteq \mathbb{N}$. The intuition behind the above division is based on the success of polar code on both WEM [48] and wiretap channel [53]: the division of sub-channels for WEM is to divide sub-channels of WEM channel into good sub-channels and bad sub-channels set, data is represented by bit values of bad sub-channel indices, i.e., frozen set value, and bits in the good sub-channel

set are equipped with the successive cancellation encoding [43] to ensure the rewriting constraint is satisfied; one division of sub-channels for wiretap channel coding is to divide sub-channels of wiretap channel into good sub-channels and bad sub-channels set, sensitive information is not represented by bits of good sub-channel bits to avoid information leaking.

Then the polar code with frozen set $\mathcal{M}$, and frozen set value $u_{\mathcal{M}}$ represents data $u_{\mathcal{M}}$. The rewriting function $\mathbf{R}(M, x_0^{N-1})$ is to fill in bits of $\mathcal{M}$ by M, bits of $\mathcal{M}_1$ by random bits, and bits of $\mathcal{M}_2$ by bits determined by successive cancellation encoding. The decoding function $\mathbf{D}(y_0^{N-1})$ is to retrieve the value represented by bits of $\mathcal{M}$.

Formally, let $\mathcal{G}'_N(\mathbb{W}, \beta)$ and $\mathcal{G}_N(\mathbb{P}, \beta)$ denote good sub-channel sets for the WEM channel $\mathbb{W}$ and the wiretap channel $\mathbb{P}$, and let $\mathcal{B}'_N(\mathbb{W}, \beta)$ and $\mathcal{B}_N(\mathbb{P}, \beta)$ denote the bad sub-channels for them, respectively. When $\mathbb{W}$ is stochastically degraded with respect to $\mathbb{P}$, it implies that $\mathcal{B}_N(\mathbb{P}, \beta) \subseteq \mathcal{B}'_N(\mathbb{W}, \beta)$ [43]. Let $\mathcal{M} \stackrel{def}{=} \mathcal{B}'_N(\mathbb{W}, \beta) \bigcap \mathcal{B}_N(\mathbb{P}, \beta) = \mathcal{B}_N(\mathbb{P}, \beta)$, $\mathcal{M}_1 \stackrel{def}{=} \mathcal{B}'_N(\mathbb{W}, \beta) \bigcap \mathcal{G}_N(\mathbb{P}, \beta)$ and $\mathcal{M}_2 \stackrel{def}{=} \mathcal{G}'_N(\mathbb{W}, \beta)$. We know that $\lim\limits_{N\to\infty} \frac{|\mathcal{M}|}{N} = H(Y|Z)$, $\lim\limits_{N\to\infty} \frac{|\mathcal{M}_1|}{N} = H(Y|X) - H(Y|Z)$ and $\lim\limits_{N\to\infty} \frac{|\mathcal{M}_2|}{N} = I(X;Y)$.

The code construction for binary symmetric secure WEM is presented below:

Algorithm 9.3.1 A code construction for binary symmetric secure WEM

1: The $(N, 2^{NR}, R, D)_{ave}$ code is $\mathcal{C} = C_N(\mathcal{M}, u_{\mathcal{M}})$, where $C_N(\mathcal{M}, u_{\mathcal{M}}(M))$ is a polar code with the frozen set $\mathcal{M}$ as above, frozen set value M, the binary representation of M is $u_{\mathcal{M}}(M)$, and $|\mathcal{M}| = NR$.

That is, the $(N, 2^{NR}, R, D)$ code is the *polar code ensemble* of codeword length N and frozen set $\mathcal{M}$ determined above, and each polar code $\mathcal{C}_N(\mathcal{M}, u_{\mathcal{M}}(M))$ $(0 \leq M \leq 2^{NR} - 1)$ of the ensemble represents the data with the binary representation $u_{\mathcal{M}}(M)$.

The rewriting operation $y_0^{N-1} = \mathbf{R}(M, x_0^{N-1})$ is below, where m_1 is a random bit,

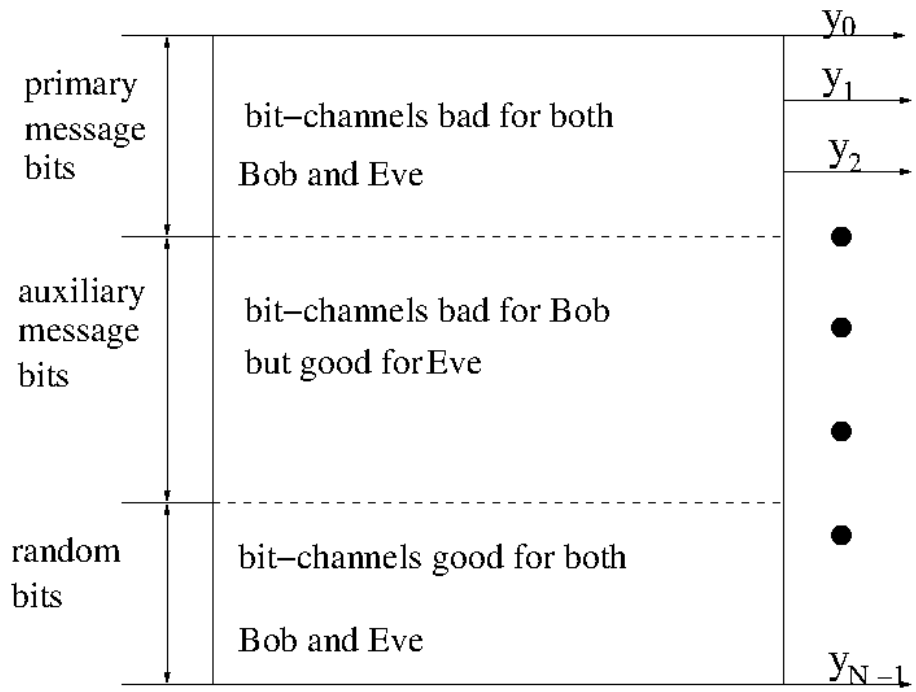


Figure 9.3: Illustration of the polar code construction for symmetric secure WEM achieving the capacity, where the output y_0^{N-1} is permuted in such a way that sub-channels are positioned as above.

$u_{\mathcal{M}}(M)_j$ is the jth bit of the binary representation of M, $f(\cdot) : \{0, 1, ..., |\mathcal{M}| - 1\} \rightarrow \mathcal{M}$ is a one-to-one mapping, and $W(y|x)$ is determined by the WEM channel $\mathbb{W} = (X, Y, W_{Y|X})$.

That is, u_0^{N-1} is assembled by rewriting message M, auxiliary random message M_1 (which is to make sure the security constraint is satisfied), and random message determined by SC encoding (which is to make sure the rewriting cost constraint is satisfied).

The decoding function $u_{\mathcal{M}}(M) = \mathbf{D}(y_0^{N-1})$ is below:

That is, $\mathbf{D}(y_0^{N-1})$ is to retrieve the value represented by bits of $\mathcal{M}$.

Algorithm 9.3.2 The rewriting operation $y_0^{N-1} = \mathbf{R}(M, x_0^{N-1})$.

1: Let $v_0^{N-1} = x_0^{N-1} + g_0^{N-1}$, where g_0^{N-1} is a common and uniformly distributed message, and $+$ is over GF(2).

2: Apply SC (Successive Cancellation) encoding [43] to $(v_0^{N-1})_{\mathcal{M}_2}$, and this results in a vector $u_0^{N-1} = \hat{U}(v_0^{N-1}, u_{\mathcal{M}}(M))$, that is, $u_j =$

$$\begin{cases} u_{\mathcal{M}}(M)_{f(j)} & \text{if } j \in \mathcal{M} \\ m_1 & \text{if } j \in \mathcal{M}_1, m_1 \text{ is randomly chosen,} \\ m & \text{with probability } \frac{W_N^{(i)}(u_0^{j-1}, v_0^{N-1}|m)}{\sum_{m'} W_N^{(i)}(u_0^{j-1}, v_0^{N-1}|m')}, \end{cases}$$

and $\hat{y}_0^{N-1} = u_0^{N-1} G_2^{\otimes n}$.

3: $y_0^{N-1} = \hat{y}_0^{N-1} + g_0^{N-1}$.

Algorithm 9.3.3 The decoding operation $u_{\mathcal{M}}(M) = \mathbf{D}(y_0^{N-1})$.

1: $\hat{y}_0^{N-1} = y_0^{N-1} + g_0^{N-1}$.

2: $u_{\mathcal{M}}(M) = (\hat{y}_0^{N-1}(G_2^{\otimes n})^{-1})_{\mathcal{M}}$.

9.3.3 Theoretical analysis of the code construction

In this part, we present the theoretical analysis showing that the presented code construction is optimal. We start with calculating the probability of a random selected vector, which is used to prove that the induced channel is symmetric, then with the symmetric channel we proceed to prove the rewriting cost constraint as well as the security constraint are satsified, and finally based on those we prove that the proposed code construction is optimal.

The probability of a random vector being selected Let $\mathcal{R} = \mathcal{M}_1 \bigcup \mathcal{M}_2$, and let $e_{\mathcal{R}}$ denote the random bits determined by the above algorithm, and in this part we focus on the probability $e_{\mathcal{R}}$ is selected given the rewriting data M, $P(e_{\mathcal{R}|M})$.

Let e_0^{N-1} denote a vector by assembling a rewriting message M and $e_{\mathcal{R}}$, and we know

that

$$P(e_0^{N-1}|v_0^{N-1}) = \prod_i P_{E_i|E_0^{i-1},V_0^{N-1}}(e_i|e_0^{i-1}, v_0^{N-1}),$$

where v_0^{N-1} is the random vector determined in our rewriting function, and $P_{E_i|E_0^{i-1},V_0^{N-1}}(e_i|e_0^{i-1}, v_0^{N-1}) = \frac{W_N^{(i)}(e_0^{i-1},v_0^{N-1}|e_i)}{\sum_{e_i'} W_N^{(i)}(e_0^{i-1},v_0^{N-1}|e_i')}$ if $i \in \mathcal{M}_2$, $\frac{1}{2}$ if $i \in \mathcal{M}_1$ and 1 otherwise.

The following lemma presents us the condition under which $\hat{U}(v_0^{N-1}, u_{\mathcal{M}}(M_1))_{\mathcal{M}^c} = \hat{U}(w_0^{N-1}, u_{\mathcal{M}}(M_2))_{\mathcal{M}^c}$. Note that in the following $+$ is over GF(2).

Lemma 68. Let $M_1, M_2 \in \mathcal{M}$, $u_{\mathcal{M}}(M_1), u_{\mathcal{M}}(M_2) \in \{0,1\}^{|\mathcal{M}|}$, let $v_0^{N-1}, w_0^{N-1} \in \{0,1\}^N$ such that $v_0^{N-1}+w_0^{N-1} = x_0^{N-1}G_2^{\otimes n}$ where $(x_0^{N-1})_{\mathcal{M}} = u_{\mathcal{M}}(M_1)+u_{\mathcal{M}}(M_2)$ and $(x_0^{N-1})_{\mathcal{M}^c}$ is the zero vector, then under the coupling through a common source of randomness, $\hat{U}(v_0^{N-1}, u_{\mathcal{M}}(M_1))_{\mathcal{M}^c} = \hat{U}(w_0^{N-1}, u_{\mathcal{M}}(M_2))_{\mathcal{M}^c}$.

Proof. Let e_0^{N-1} and f_0^{N-1} be the result of $\hat{U}(v_0^{N-1}, u_{\mathcal{M}}(M_1))$ and $\hat{U}(w_0^{N-1}, u_{\mathcal{M}}(M_2))$. We prove that $e_i = f_i + ((v_0^{N-1} + w_0^{N-1})(G_2^{\otimes n})^{-1})_i$ for $0 \le i \le N-1$ by induction. This holds true for $i = 0$.

Now suppose this also holds true for $i-1$, and now consider the case for i. As $e_i = f_i + ((v_0^{N-1} + w_0^{N-1})(G_2^{\otimes n})^{-1})_i$ holds true for $i \in \mathcal{M}$, we only consider $i \in \mathcal{M}^c$.

First consider $i \in \mathcal{M}_1$, since they have access to the same random source, clearly $e_i = f_i + ((v_0^{N-1} + w_0^{N-1})(G_2^{\otimes n})^{-1})_i$.

Second consider $i \in \mathcal{M}_2$, and it is proved using a skill similar to [42, Lemma 8].

$$
\begin{aligned}
\frac{W_N^{(i)}(v_0^{N-1}, e_0^{i-1}|1)}{W_N^{(i)}(v_0^{N-1}, e_0^{i-1}|0)} &= \frac{\sum_{e_{i+1}^{N-1}} W^N(v_0^{N-1}|e_0^{i-1}1e_{i+1}^{N-1})}{\sum_{e_{i+1}^{N-1}} W^N(v_0^{N-1}|e_0^{i-1}0e_{i+1}^{N-1})}, \\
&= \frac{\sum_{e_{i+1}^{N-1}} W^N(w_0^{N-1}|e_0^{i-1}1e_{i+1}^{N-1} + (v_0^{N-1} + w_0^{N-1})(G_2^{\otimes n})^{-1})}{\sum_{e_{i+1}^{N-1}} W^N(w_0^{N-1}|e_0^{i-1}0e_{i+1}^{N-1} + (v_0^{N-1} + w_0^{N-1})(G_2^{\otimes n})^{-1})}, \\
&= \frac{\sum_{e_{i+1}^{N-1}} W^N(w_0^{N-1}|f_0^{i-1}1e_{i+1}^{N-1})}{\sum_{e_{i+1}^{N-1}} W^N(w_0^{N-1}|f_0^{i-1}0e_{i+1}^{N-1})}, \\
&= \frac{W_N^{(i)}(w_0^{N-1}, f_0^{i-1}|1)}{W_N^{(i)}(w_0^{N-1}, f_0^{i-1}|0)},
\end{aligned}
$$

where the third equation is due to the assumption that $((v_0^{N-1} + w_0^{N-1})(G_2^{\otimes n})^{-1})_{\mathcal{M}^c}$ is the zero vector and the assumption $e_j = f_j + ((v_0^{N-1} + w_0^{N-1})(G_2^{\otimes n})^{-1})_j$ for $j \leq i-1$.

Thus $\hat{U}(v_0^{N-1}, u_{\mathcal{M}}(M_1))_i = \hat{U}(w_0^{N-1}, u_{\mathcal{M}}(M_2))_i$ when they have access to the same random source. Thus we conclude $e_i = f_i + ((v_0^{N-1} + w_0^{N-1})(G_2^{\otimes n})^{-1})_i$, and $\hat{U}(v_0^{N-1}, u_{\mathcal{M}}(M_1))_{\mathcal{M}^c} = \hat{U}(w_0^{N-1}, u_{\mathcal{M}}(M_2))_{\mathcal{M}^c}$. □

Let $P(e_{\mathcal{R}}|M)$ denote the average probability (over v_0^{N-1}) that $e_{\mathcal{R}}$ is chosen given M, and

$$
\begin{aligned}
P(e_{\mathcal{R}}|M) &= \sum_{v_0^{N-1}} P(v_0^{N-1}) P(e_0^{N-1}|v_0^{N-1}), \\
&= \sum_{v_0^{N-1}} \frac{1}{2^N} P(e_0^{N-1}|v_0^{N-1}),
\end{aligned}
$$

as v_0^{N-1} is uniformly distributed.

The following theorem presents us on average the probability that $e_{\mathcal{R}}$ is chosen given

M is the same for any M.

Theorem 69. $P(e_{\mathcal{R}}|M)$ is independent of M, i.e., $P(e_{\mathcal{R}}|M_1) = P(e_{\mathcal{R}}|M_2)$ for any M_1, M_2.

Proof. The correctness holds by the fact that for each v_0^{N-1} there is a unique w_0^{N-1} such that $e_{\mathcal{R}} = \hat{U}(v_0^{N-1}, u_{\mathcal{M}}(M_1))_{\mathcal{M}^c} = \hat{U}(w_0^{N-1}, u_{\mathcal{M}}(M_2))_{\mathcal{M}^c}$ based on the previous lemma. □

As $P(e_{\mathcal{R}}|M)$ is independent of M, hereafter we will omit M and write $P(e_{\mathcal{R}}|M)$ as $P(e_{\mathcal{R}})$.

The induced channel is symmetric The induced channel is pictorially presented in Figure 9.4, where the input is $N-r$ bits $u_{\mathcal{M}}$, representing the rewriting data, and the output of the channel is z_0^{N-1}, the output of y_0^{N-1} through the wiretap channel. Let (v_0^{N-r-1}, e_0^{r-1}) denote the vector u_0^{N-1} with $u_{\mathcal{R}} = v_0^{N-r-1}$ and $u_{\mathcal{R}^c} = e_0^{r-1}$.

For this channel its channel transition probability is $\mathcal{Q}(z_0^{N-1}|u_0^{N-r-1}) =$

$$\sum_{e_0^{r-1}} P(e_0^{r-1}) \prod_{i=0}^{N-1} P(z_i|((u_0^{N-r-1}, e_0^{r-1})G_2^{\otimes n})_i),$$

where $P(e_0^{r-1})$ denotes the probability e_0^{r-1} is selected given the rewriting data vector u_0^{N-r-1}, it is determined as the previous part, and $P(z|x)$ is determined by the wiretap channel $\mathbb{P} = (\mathcal{X}, \mathcal{Z}, P_{Z|X})$. For convenience, we denote our channel as $\mathbb{Q}(\mathbb{P}, \mathcal{R}) = (\mathcal{X}^{N-r}, \mathcal{Z}^N, \mathcal{Q}_{Z^N|U^{N-r}})$. where $\mathcal{X} = \{0, 1\}$.

Note that our definition of $\mathbb{Q}(\mathbb{P}, \mathcal{R})$ shares some similarities with the induced channel in [53, Subsection C of Section VI], that is, the inputs of both channels are data communicated to decoders, and the outputs of them are both noisy codewords through the wiretap channel. However, the channels differ in their transition probabilities, which stems from how the random bits are determined, i.e., for channel in [53], the random bits are chosen

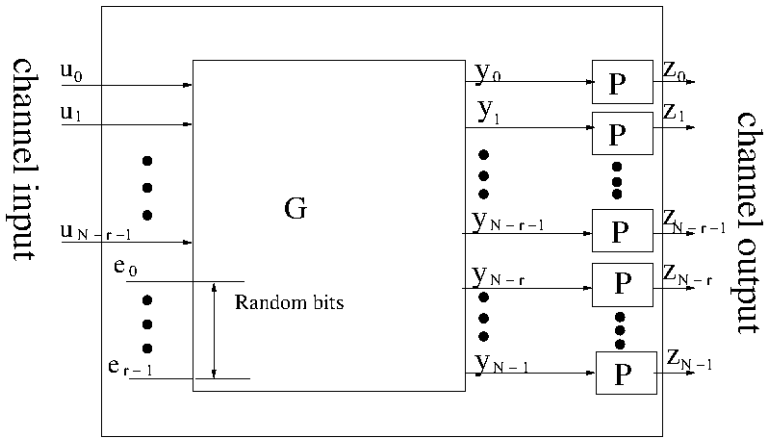


Figure 9.4: Illustration of the induced channel, where the output y_0^{N-1} is permuted the same way as before such that sub-channels are positioned as the above figure; and where the channel inputs are u_0^{N-r-1} (i.e., rewriting data) and the channel outputs are z_0^{N-1} (i.e., noisy codeword of y_0^{N-1} though wiretap channel).

independently and uniformly, and for our channel, the random bits are partially determined by successive cancellation encoding and are partially determined independently and uniformly.

We now present the main result in the following theorem, which presents us $\mathbb{Q}(\mathbb{P}, \mathcal{R})$ is symmetric.

Theorem 70. $\mathbb{Q}(\mathbb{P}, \mathcal{R})$ *is symmetric.*

Proof. Given a channel $(\mathcal{X}, \mathcal{Y}, W_{Y|X})$, we first recall the definition of symmetric channel from group theory. *A group action* of an abelian group $\mathcal{A}$ on a set $\mathscr{Y}$ is a function $\mathcal{A} \times \mathscr{Y} \to \mathscr{Y}$, denoted $(a, y) \to a \cdot y$, with the following properties:

- $0 \cdot y = y$ for all $y \in \mathscr{Y}$, where 0 is the unit of $\mathcal{A}$;
- $(a + b) \cdot y = a \cdot (b \cdot y)$ for all $a, b \in \mathcal{A}$ and all $y \in \mathscr{Y}$, where $+$ denotes the group operation for $\mathcal{A}$.

The following result from [53, Theorem 11] states a necessary condition under which the channel is symmetric.

> Let $(\mathcal{X}, \mathcal{Y}, W_{Y|X})$ be a discrete memoryless channel, and suppose that $\mathcal{X}$ is an abelian group under the binary operation $+$. Further, suppose that there exists a group action $\cdot$ of $\mathcal{X}$ on $\mathcal{Y}$ such that
>
> $$W(y|a+x) = W(a.y|x)$$
>
> for all $a, x \in \mathcal{X}$ and all $y \in \mathcal{Y}$. Then $(\mathcal{X}, \mathcal{Y}, W_{Y|X})$ is a symmetric channel.

For $\mathbb{Q}(\mathbb{P}, \mathcal{R}) = (\mathcal{X}^{N-r}, \mathcal{Z}^N, \mathcal{Q}_{Z^N|U^{N-r}})$, we first explore an action of $\mathcal{X}^{N-r}$, denoted as $\cdot$, such that $(\mathcal{X}^{N-r}, \cdot)$ is an abelian group, and we then explore a group action, denoted as $\circ$ of the abelian group $\mathcal{X}^{N-r}$ on $\mathcal{Z}^N$, such that $\mathbb{Q}(\mathbb{P}, \mathcal{R})$ is symmetrical based on the above cited result

Let π_1 be a permutation on $\mathcal{Z}$ and it is an involution, that is $\pi_1 = \pi_1^{-1}$. Let π_0 be the identity permutation on $\mathcal{Z}$. Following Arikan [5], let the group action of the additive group of $\mathcal{X} = \{0, 1\}$ on the set $\mathcal{Z}$ be $x \cdot z = \pi_x(z)$ for all $x \in \mathcal{X}$ and $z \in \mathcal{Z}$. The group action has the property $(x + y) \cdot z = x \cdot (y \cdot z)$ and $(x \cdot y) \cdot z = x \cdot (y \cdot z)$ which can be verified based on enumeration. Therefore, the additive group $\mathcal{X}$ with the operation $\cdot$ is an abelian group.

Similarly, let $x_0^{N-1} \cdot z_0^{N-1} = (x_0 \cdot z_0, \cdots, x_{N-1} \cdot z_{N-1})$ for all $x_0^{N-1} \in \mathcal{X}^N$ and $z_0^{N-1} \in \mathcal{Z}^N$. The action has the property $(x_0^{N-1} + y_0^{N-1}) \cdot z_0^{N-1} = x_0^{N-1} \cdot (y_0^{N-1} \cdot z_0^{N-1})$ based on the property $(x+y) \cdot z = x \cdot (y \cdot z)$, and $(x_0^{N-1} \cdot y_0^{N-1}) \cdot z_0^{N-1} = x_0^{N-1} \cdot (y_0^{N-1} \cdot z_0^{N-1})$ based on the property $(x \cdot y) \cdot z = x \cdot (y \cdot z)$. Therefore, the additive group $\mathcal{X}^N$ with the operation $\cdot$ is an abelian group.

Define $\circ$ as $x_0^{N-r-1} \circ z_0^{N-1} \stackrel{def}{=} (x_0^{N-r-1}, 0_0^{r-1}) G_2^{\otimes n} \cdot z_0^{N-1}$. We can verify that the defined action is a *group action* as it satisfies the following two requirements.

- $0_0^{N-r-1} \circ z_0^{N-1} = z_0^{N-1}$;

- $(x_0^{N-r-1} + y_0^{N-r-1}) \circ z_0^{N-1} = x_0^{N-r-1} \circ (y_0^{N-r-1} \circ z_0^{N-1})$,

where the second item is due to $(x_0^{N-r-1} + y_0^{N-r-1}) \circ z_0^{N-1}$

$$\begin{aligned}
&= ((x_0^{N-r-1}, 0_0^{r-1}) + (y_0^{N-r-1}, 0_0^{r-1}))G_2^{\otimes n} \cdot z_0^{N-1} \\
&= (x_0^{N-r-1}, 0_0^{r-1})G_2^{\otimes n} \cdot ((y_0^{N-r-1}, 0_0^{r-1})G_2^{\otimes n} \cdot z_0^{N-1}) \\
&= x_0^{N-r-1} \circ (y_0^{N-r-1} \circ z_0^{N-1}),
\end{aligned}$$

where the second equation is based on the property $(x_0^{N-1} + y_0^{N-1}) \cdot z_0^{N-1} = x_0^{N-1} \cdot (y_0^{N-1} \cdot z_0^{N-1})$.

We proceed the proof as follows:

$\mathcal{Q}(z_0^{N-1}|a_0^{N-r-1} + x_0^{N-r-1})$

$$\begin{aligned}
&= \sum_{e_0^{r-1}} P(e_0^{r-1}) \prod_i P(z_0^{N-1}|(((a_0^{N-r-1}, 0_0^{r-1}) + (x_0^{N-r-1}, e_0^{r-1}))G_2^{\otimes n}))_i, && (9.1) \\
&= \sum_{e_0^{r-1}} P(e_0^{r-1}) \prod_i P((a_0^{N-r-1}, 0_0^{r-1})G_2^{\otimes n} \cdot z_0^{N-1}|(x_0^{N-r-1}, e_0^{r-1})G_2^{\otimes n}))_i, && (9.2) \\
&= \sum_{e_0^{r-1}} P(e_0^{r-1}) \prod_i P(a_0^{N-r-1} \circ z_0^{N-1}|(x_0^{N-r-1}, e_0^{r-1})G_2^{\otimes n}))_i, && (9.3) \\
&= \mathcal{Q}(a_0^{N-r} \circ z_0^{N-1}|x_0^{N-r-1}),
\end{aligned}$$

where

(9.1) follows from the definition of $\mathcal{Q}(z_0^{N-1}|u_0^{N-r-1})$;

(9.2) follows from [5, Proposition 12]. i.e., $P^N(z_0^{N-1}|(a_0^{N-1}+x_0^{N-1})G_2^{\otimes n}) = P^N(a_0^{N-1}G_2^{\otimes n} \cdot z_0^{N-1}|x_0^{N-1}G_2^{\otimes n})$ and $P^N(z_0^{N-1}|x_0^{N-1}) = \prod_{i=0}^{N-1} P(z_i|x_i)$;

(9.3) follows from our definition of the operation $\circ$, and also from Theorem 69.

$\square$

Rewriting cost constraint and security constraint We first focus on the rewriting cost constraint. From [48, Theorem 9], we know that as long as $\mathcal{M}_2 \subseteq \mathcal{G}'_N(\mathbb{W}, \beta)$, with high probability $\varphi(x_0^{N-1}, y_0^{N-1}) \leq D$ for arbitrary x_0^{N-1}, y_0^{N-1}, i.e., $Pr(\varphi(x_0^{N-1}, y_0^{N-1}) \geq D + \sigma) < 2^{-N^\beta}$ for $\sigma > 0$. Therefore based on our selection of $\mathcal{M}_2$, which is $\mathcal{M}_2 = \mathcal{G}'_N(\mathbb{W}, \beta)$, the rewriting cost constraint is satisfied with high probability.

We next focus on the security constraint, and we apply a skill similar to [19, 53]. $I(M; z_0^{N-1})$

$$\begin{aligned}
&\leq I(\hat{u}_M; \hat{z}_0^{N-1}), && (9.4)\\
&= I(\bar{u}_M; \bar{z}_0^{N-1}), && (9.5)\\
&= \sum_{i=0}^{|\mathcal{M}|} I(\bar{u}_i; \bar{z}_0^{N-1} | \bar{u}_0, \cdots, \bar{u}_{i-1}), && (9.6)\\
&= \sum_{i=0}^{|\mathcal{M}|} I(\bar{u}_i; \bar{z}_0^{N-1} \bar{u}_0^{i-1}), && (9.7)\\
&= \sum_{i=0}^{|\mathcal{M}|} C(P_N^{(i)}), && (9.8)
\end{aligned}$$

where

(9.4) follows from the channel $\mathbb{Q}(\mathbb{P}, \mathcal{R})$ is symmetric, and $\hat{u}_M$ and $\hat{z}_0^{N-1}$ denote versions of u_M and z_0^{N-1} when u_i and z_i are uniformly and independently distributed;

(9.5) is due to the permutation such that u_0^{N-1} is arranged as Figure 9.3;

(9.6) is due to the chain rule of mutual information;

(9.7) is due to $\bar{u}_i$ is independent of each other;

(9.8) is due to $\mathbb{P}_N^{(i)}$ is i-th virtual bit channel induced by the wiretap channel $\mathbb{P} = (\mathcal{X}, \mathcal{Z}, P_{Z|Y})$ (refer to Section 9.2 for its definition).

Based on our selection of $\mathcal{M}$, which is $\mathcal{B}_N(\mathbb{P}, \beta)$, we know that $C(\mathbb{P}_N^{(i)}) \leq 2^{-N^\beta}$ and further obtain $\frac{I(M;z_0^{N-1})}{N} \leq \frac{|\mathcal{B}_N(\mathbb{P},\beta)|}{N} 2^{-N^\beta}$, which is approaching 0 as $N \to \infty$.

Therefore, we can conclude that the security constraint is satisfied since $\frac{1}{N} H(M|z_0^{N-1}) = \frac{1}{N} H(M) - \frac{1}{N} I(M; z_0^{N-1}) \to R$ as $N \to \infty$.

Capacity approaching property When the WEM channel is stochastically degraded with respect to the wiretap channel, the secrecy capacity is $H(Y|Z)$ [47], as $\lim_{N\to\infty} \frac{|\mathcal{M}|}{N} = H(Y|Z)$, our code construction is achieving the secrecy capacity asymptotically.

Theoretical performance conclusion Thus based on analysis from $a) \sim d)$, we have the following conclusion for theoretical performance of our proposed code construction:

Theorem 71. For any symmetric secure WEM, when the WEM channel is stochastically degraded with respect to the wiretap channel, there exists a polar code achieving the secrecy capacity.

9.3.4 Optimal code construction achieving the whole region

In this subsection, we extend the above code construction to achieve the whole rewriting-rate-equivocation region.

Given a $\forall (R, R_e) \in$

$$\{(R, R_e) : \begin{array}{ll} R & \leq H(Y|X) \\ R_e & \leq H(Y|Z) \\ R_e & \leq R \\ H(Y|Z) & \leq H(Y|X) \end{array} \}, \tag{9.9}$$

for a $P_{XY} \in \mathcal{P}^s(D)$, we know that based on the code construction in the previous sub-

section, we can construct a code construction for $(N, 2^{NR_e}, R_e, D)$ symmetric secure WEM, and partition the set $\{0, 1, \cdots, N-1\}$ into $\mathcal{B}'_N(\mathbb{W}, \beta) \bigcap \mathcal{B}_N(\mathbb{P}, \beta) = \mathcal{B}_N(\mathbb{P}, \beta)$, $\mathcal{B}'_N(\mathbb{W}, \beta) \bigcap \mathcal{G}_N(\mathbb{P}, \beta)$ and $\mathcal{G}'_N(\mathbb{W}$. We know that $R_e = \frac{|\mathcal{B}'_N(\mathbb{W},\beta) \bigcap \mathcal{B}_N(\mathbb{P},\beta)|}{N}$. Our code construction for an $(N, 2^{NR}, R_e, D)$ symmetric secure WEM is as follows:

- let $\mathcal{M}^1 = \mathcal{B}'_N(\mathbb{W}, \beta) \bigcap \mathcal{B}_N(\mathbb{P}, \beta)$ of size NR_e;
- let $\mathcal{M}^2 \subseteq \mathcal{B}'_N(\mathbb{W}, \beta) \bigcap \mathcal{G}_N(\mathbb{P}, \beta)$ of size $N(R - R_e)$ whose elements have lowest $I(\mathbb{W}_N^{(i)})$;
- let $\mathcal{M} = \mathcal{M}^1 \bigcup \mathcal{M}^2$;
- let $\mathcal{M}_1 = \mathcal{B}'_N(\mathbb{W}, \beta) \bigcap \mathcal{G}_N(\mathbb{P}, \beta) - \mathcal{M}^2$;
- let $\mathcal{M}_2 = \mathcal{G}'_N(\mathbb{W}, \beta)$;
- the $(N, 2^{NR}, R_e, D)_{ave}$ code is $\mathcal{C} = \bigcup_M C_N(\mathcal{M}, u_{\mathcal{M}}(M))$, where $C_N(\mathcal{M}, u_{\mathcal{M}}(M))$ is a polar code with the frozen set $\mathcal{M}$ and frozen set value M with its binary representation $u_{\mathcal{M}}(M)$.

That is, comparing with the previous code construction, the only difference is that bits of $\mathcal{B}'_N(\mathbb{W}, \beta) \bigcap \mathcal{G}_N(\mathbb{P}, \beta)$ in this case also represent user information, i.e., in Figure 9.3, some auxiliary message bits carry information.

The rewriting function and the decoding function are the same as previous ones. We conclude its performance in the following theorem.

Theorem 72. For any symmetric secure WEM code (R, R_e) satisfying (9.9), when the WEM channel is stochastically degraded with respect to the wiretap channel, there exists a polar code achieving the whole region.

Proof. We present the sketch proof as follows. We first focus on the rewriting cost constraint: since $\mathcal{M}_2 \subseteq \mathcal{G}'_N(\mathbb{W}, \beta)$ (the same as the previous subsection), similarly based on [48, Lemma 7] or [42, Theorem 1] we obtain the average rewriting cost $\bar{D} \leq D + O(2^{-N^\beta})$.

Next we focus on the security constraint: with similar arguments of $a) \sim c)$ of the previous subsection, we can prove that the channel $\mathbb{Q}(\mathbb{P}, \mathcal{R})$ is still symmetric in this case; similarly, we obtain

$$
\begin{aligned}
I(M; z_0^{N-1}) & \\
&\leq \sum_{i=0}^{|\mathcal{M}^1 \bigcup \mathcal{M}^2|} C(\mathbb{P}_N^{(i)}), && (9.10)\\
&\leq \sum_{i=0}^{|\mathcal{M}^2|} C(\mathbb{P}_N^{(i)}) + \frac{|\mathcal{B}_N(\mathbb{P}, \beta)|}{N} 2^{-N^\beta}, && (9.11)\\
&\leq N(R - R_e) + \epsilon, && (9.12)
\end{aligned}
$$

where

(9.10) follows from the similar arguments of $d)$ in the previous subsection;

(9.11) is due to the selection of $\mathcal{M}^1$ and the definition of $\mathcal{B}_N(\mathbb{P}, \beta)$;

(9.12) is due to the selection of $\mathcal{M}^2$ and the definition of $\mathcal{G}_N(\mathbb{P}, \beta)$.

Thus we further obtain $\frac{1}{N} H(M|z_0^{N-1}) \geq R_e + \epsilon$ as desired. □

9.4 Conclusion

In this work, we present a construction based on polar codes has been proposed for symmetric secure WEM and prove that it is achieving its capacity.

We list some possible future work here:

- In this work we assume that the main channel is perfect for simplicity, and it is interesting to explore the case when both channels are noisy.

- Another possible future work is to explore other code construction based on other codes, e.g., LDPC codes.

- It is interesting to explore the similarity between secure WEM and wiretap channel codes [73]: wiretap channel codes expand channel codes by a security constraint to ensure messages can be communicated both *reliably* and *securely*, and similarly secure WEM codes expand WEM codes also by a security constaint to guarantee that messages can be communicated both *cost-effectively* and *securely*. We are interested in further connection between them.

10. CONCLUDING REMARKS AND FUTURE WORK

Storage systems are experiencing a historical paradigm shift from hard disk to non-volatile memories. Those changes are due to their advantages such as higher density, scaling size and its non-volatility, etc.

On the other hand, there are serious challenges (such as endurance, reliability and security) to solve due to flash memories characteristics, such as block erasure, unique noise, out-of-place update and etc.

In this dissertation, as a step in this direction, we first study noise modeling and capacity analysis for NAND flash memories, which gains us some insight on how flash memories are working and their unique noise; Second, based on the characteristics of content-replication in flash memories, we propose a joint decoder to enhance the flash memories reliability; Third, we explore data representation schemes in flash memories and optimal rewriting code constructions in order to solve the endurance problem; Fourth, in order to make our rewriting code more practical, we study noisy write-efficient memories and WOM codes against inter-cell interference in NAND memories; Finally, motivated by the secure deletion problem in flash memories, we study coding schemes to solve both the endurance and the security issues in flash memories.

My prior work has established an effective framework for flash memory storage system as well as some potentially challenging research directions that I plan to address in the near future. Examples of those topics include designing code model to address the endurance, the reliability and the security simultaneously, and designing more practical rewriting codes to make them appealing for industry. In the long term, I intend to have a flexible and adaptable research agenda to tackle emerging challenges.

Examples of research directions in the next one year are summarized below:

Secure Noisy Write-Efficient Memory As shown in prior work, optimal codes have been obtained for rewriting code, enhanced rewriting codes with functions either error correcting or avoid information leaking have also explored. It is natural to explore the rewriting code model having both functionalists. To that goal, the first work is to have a theoretical study to explore the possibility of this code model and to investigate its best performance, and the next work is to explore efficient code constructions.

Making rewriting code possible in product Research work in rewriting code has been around ten years, and various efficient and even optimal code construction schemes have been proposed. However, there is still a long way to go before we can put rewriting code into products, and reasons include the rate of rewriting code is too low, and the write latency is too large. Challenging as they are, meanwhile they provide me further research opportunities to make rewriting code more practical.

Examples of research directions in the next three years are summarized below:

Efficient RAIN for flash Memory Besides error correction codes, another scheme to protect data in flash memory is traditional Redundant Array and Independent Disk (RAID) method where some industries are employing in their products. However, as its name suggests, RAID was designed specifically for hard disk not for flash memories, therefore methods like Redundant Array and Independent NAND flash memories (RAIN) are necessary and urgent. There is plenty of research work can be done, for example the first work is to understand why RAID is not suitable for flash memories by doing experiments, and after that more work can be done to design an efficient RAIN.

Integrating machine learing in decoding algorithms Efficient soft decoding error correction codes like LDPC have been employed in flash memories and show great advantages over traditional hard decoding algorithms, however, one great challenge is the difficulty of obtaining soft information, which is in essence to estimate a probability density function. As is well known that machine learning is an efficient tool to extract desired

information from large amount of data and currently is used in various areas, it will be helpful and interesting to integrating machine learning in decoding algorithms. However, as many machine learning algorithms are computation intensive, thus the challenging is to design an efficient machine learning algorithm suitable for current flash memories.

To conclude this dissertation, we firmly believe that flash memories represent a significant change and they bring us lots of challenges and endless opportunites for industry and academia.

REFERENCES

[1] A. V. Kuznetsov and A. J. H. Vinck. On the general defective channel with informed encoder and capacities of some constrained memories. *IEEE Transaction on Information Theory*, 40(6):1866–1871, November 1994.

[2] R. Ahlswede and Z. Zhang. Coding for write-efficient memory. *Information and Computation*, 83(1):80–97, October 1989.

[3] R. Ahlswede and Z. Zhang. On multiuser write-efficient memories. *IEEE Transactions on Information Theory*, 40(3):674–686, May 1994.

[4] Mattias Andersson. *Coding for the Wiretap Channal*. PhD thesis, KTH, 2011.

[5] E. Arikan. Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Transactions on Information Theory*, 55(7):3051–3073, July 2009.

[6] Mihir Bellare, Stefano Tessaro, and Alexander Vardy. Semantic security for the wiretap channel. In *International Cryptology Conference (CRYPTO)*, pages 294–311, 2012.

[7] A. Berman and Y. Birk. Constrained flash memory programming. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, pages 2128–2132, St. Petersburg, Russia, August 2011.

[8] M.Bloch Bloch and J. Barros. *Physical-Layer Security: From Information Theory to Security Engineering*. Cambridge University Press, 2011.

[9] David Burshtein and Alona Strugatski. Polar Write Once Memory Codes. *IEEE Transacation on Information Theory*, 59(8):5088–5101, August 2013.

[10] Yu Cai, Erich F. Haratsch, Onur Mutlu, and Ken Mai. Error patterns in MLC NAND flash memory: measurement, characterization, and analysis. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '12, pages 521–526, Dresden, Germany, 2012.

[11] Yu Cai, Erich F. Haratsch, Onur Mutlu, and Ken Mai. Threshold voltage distribution in MLC NAND flash memory: characterization, analysis, and modeling. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '13, pages 1285–1290, Grenoble, France, 2013.

[12] Yu Cai, Onur Mutlu, Erich F. Haratsch, and Ken Mai. Program interference in mlc nand flash memory: Characterization, modeling, and mitigation. In *ICCD*, pages 123–130, 2013.

[13] P. Cappelletti, C. Golla, P. Olivo, and E. Zanoni. *Flash memories*. Kluwer, Norwell, MA, 1999.

[14] Yuval Cassuto. Not just for errors: codes for fast and secure flash storage. In *Globecom 2010, Workshop on the Application of Communication Theory to Emerging Memory Technologies*, pages 1871–1875, 2010.

[15] T. M. Cover. Enumerative source coding. *IEEE Transaction on Information Theory*, 19(1):73–77, January 1973.

[16] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, New York, 1991.

[17] G. Crisenza, C. Clementi, G. Ghidini, and M. Tosi. Floating Gate Memroies Reliability. *in Proc. ESREF*, 8:177–187, 1992.

[18] Imre Criszar and Janos Korner. Broadcast channels with confidential messages. *IEEE Transaction on Information Theory*, 24(3):339–348, May 1978.

[19] Eren Şaşoğlu and Alexander Vardy. A New Polar Coding Scheme for Strong Security on Wiretap Channels. In *Proc. IEEE International Symposium on Information Theory(ISIT)*, pages 1117–1121, June 2013.

[20] Imre Csiszar and Janos Korner. *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Academic Press, Inc., Orlando, FL, USA, 1982.

[21] Peter Desnoyers. Analytic modeling of ssd write performance. In *International Systems and Storage conference (SYSTOR 2012)*, June 2012.

[22] G. Dong, S. Li, and T. Zhang. Using data post-compensation and pre-distortion to tolerate cell-to-cell interference in mlc nand flash memory. *IEEE Transactions on Circuits and Systems I*, 57(10):2718–2728, 2010.

[23] Guiqiang Dong, Ningde Xie, and Tong Zhang. On the Use of Soft-Decision Error-Correction Codes in NAND Flash Memory. *IEEE Trans. on Circuits and Systems*, 58(2):429–439, February 2011.

[24] Guiqiang Dong, Ningde Xie, and Tong Zhang. Enabling NAND Flash Memory Use Soft-Decision Error Correction Codes at Minimal Read Latency Overhead. *IEEE Trans. on Circuits and Systems*, 60(9):2412–2421, 2013.

[25] A. Fiat and A. Shamir. Generalized write-once memories. *IEEE Trans.Inf. Theory*, 30(3):34–42, May 1984.

[26] F. Fu and A. J. Han Vinck. On the Capacity of Generalized Write-Once Memory with State Transitions Described by an Arbitrary Directed Acyclic Graph. *IEEE Trans on Inf Theory*, 45(1):308–313, Septemper 1999.

[27] F. Fu and R. W. Yeung. On the capacity and error-correcting codes of write-efficient memories. *IEEE Transcations on Information Theory*, 46(7):2299–2314, November 2000.

[28] Minyue Fu. On gaussian approximation for density evolution of low-density parity-check codes. In *Int. Conf. Communications*, Istanbul, May 2006.

[29] E. En Gad, A. Jiang, and J. Bruck. Trade-offs between instantaneous and total capacity in multi-cell flash memories. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, pages 990–994, Cambridge, MA, U.S.A, June 2012.

[30] Eyal En Gad, Wentao Huang, Yue Li, and Jehoshua Bruck. Rewriting flash memories by message passing. In *Proc. IEEE International Symposium on Information Theory(ISIT)*, Hongkong, June 2015.

[31] Eran Gal and Sivan Toledo. Algorithms and data structures for flash memories. *ACM Computing Surveys*, 37:138–163, 2005.

[32] S. I. Gel'fand and M. S. Pinsker. Coding for Channel with Random Parameters. *Problems of Control Theory*, 9(1):19–31, 1980.

[33] C. D. Heegard. On the capacity of permanent memory. *IEEE Trans.Inf. Theory*, 31(1):34–42, Januray 1985.

[34] D. Ielmini, S. Lavizzari, D. Sharma, and A. Lacaita. Physical interpretation, modeling and impact on phase change memory (PCM) reliability of resistance drift due

to chalcogenide structural relaxation. *IEEE International Electron Devices Meeting, 2007. IEDM 2007*, pages 939–942, 2007.

[35] A. Jiang, V. Bohossian, and J. Bruck. Rewriting codes for joint information storage in flash memories. *IEEE Transactions on Information Theory*, 56(10):5300–5313, October 2010.

[36] A. Jiang, R. Mateescu, M. Schwartz, and J. Bruck. Rank modulation for flash memories. *IEEE Transactions on Information Theory*, 55(6):2659–2673, June 2009.

[37] Anxiao Jiang, Yue Li, Eyal En Gad, Michael Lanberg, and Jehoshua Bruck. Joint rewriting and error correction in write-once memories. In *Proc. IEEE International Symposium on Information Theory(ISIT)*, pages 1067–1071, Istanbul, Turkey, June 2013.

[38] Anxiao Jiang and Yue Wang. Rank Modulation with Multiplicity. In *Proc. IEEE Workshop on Application of Communication Theory to Emerging Memory Technologies*, pages 1928–1932, December 2010.

[39] Srdjan Capkun Joel Reardon and David Basin. SoK: Secure Data Deletion. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2013.

[40] J. Justesen and T. Høholdt. Maxentropic markov chain. *IEEE Transactions on Information Theory*, 30:665–666, July 1984.

[41] M. Karzand and E. Teletar. Polar codes for q-ary source coding. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, pages 909–912, Austin, Texas, June 2010.

[42] Satis Bahu Korada and Rudiger Urbanke. Polar codes are optimal for lossy source coding. *IEEE Trans. Inf. Theory*, 56(4):1751–1768, April 2010.

[43] Satish Babu Korada. *Polar codes for channel and source coding*. PhD thesis, EPFL, 2010.

[44] L. A. Lastras-Montano, M. Franceschini, T. Mittelholzer, J. Karidis, and M. Wegman. On the lifetime of multilevel memories. In *Proc. IEEE International Symposium on Information Theory (ISIT2009)*, pages 1224–1228, Coex, Seoul, Korea, 2009.

[45] J.-D. Lee, S.-H. Hur, and J.-D. Choi. Effects of floating-gate interference on nand flash memory cell operation. *IEEE Electron. Device Lett*, 23(5):264–266, May 2002.

[46] Qing Li. Compressed Rank Modulation. In *Proc. 50th Annual Allerton Conference on Communication, Control and Computing (Allerton 2012)*, pages 185–192, Monticello, IL, October 2012.

[47] Qing Li and Andrew Jiang. Coding for secure write-efficient memories. In *Proc. 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton 2014)*, pages 505–512, Monticello, IL, October 2014.

[48] Qing Li and Anxiao Jiang. Polar codes are optimal for Write-efficient Memories. In *Proc. 51th Annual Allerton Conference on Communication, Control and Computing (Allerton 2013)*, pages 660–667, Monticello, IL, October 2013.

[49] Qing Li, Anxiao Jiang, and Erich F. Haratsch. Noise Modeling and Capacity Analysis for NAND Flash Memories. In *Proc. IEEE International Symposium on Information Theory(ISIT)*, pages 2262–2266, Honolulu, HI, June 2014.

[50] Qing Li, Anxiao Jiang, and Erich F. Haratsch. Joint Decoder of Content-Replication Codes for NAND Flash Memories. In *Proc. Non-volatile Memory Workshop*, San Diego, CA, March 2015.

[51] Yingbin Liang, H. Vincent Poor, and Shlomo Shamai (Shitz). Information theoretic security. *Found. Trends Commun. Inf. Theory*, 5(4):355–580, Aprail 2009.

[52] D. J. C. MacKay. Fountain codes. *IEE Proc.-Commun*, 152:1062–1068, 2005.

[53] H. Mahdavifar and A. Vardy. Achieving the Secrecy Capacity of Wiretap Channels Using Polar Codes. *IEEE Transactions on Information Theory*, 57(10):6428–6443, October 2011.

[54] Brian Marcus, Ronny Roth, and Paul Siegel. Introduction to coding for constrainted systems. available from http://www.math.ubc.ca/ marcus/Handbook/.

[55] Saher Odeh and Yuval Cassuto. NAND Flash Architectures Reducing Writes Amplification Through Multi-Write Codes. In *IEEE 30th Symposium on Mass Storage Systems on Technologies (MSST)*, 2014.

[56] K.-T. Park, M. Kang, D. Kim, S.-W Hwang, B.Y. Choi, Y.-T. Lee, C. Kim, and K. Kim. A zeroing cell-to-cell interference page architecture with temporary lsb storing and parallel msb program scheme for mlc nand flash memories. *IEEE J. Solid-State Circuits*, 40(4):919–928, April 2008.

[57] W.Y. Park and A. Barg. Polar codes for q-ary channels, $q = 2^r$. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, pages 2142–2146, Boston, MA, June 2012.

[58] Joel Reardon, Srdjan Capkun, and David Basin. Data node encrypted file system: Efficient secure deletion for flash memory. In *Proceedings of the 21st USENIX Conference on Security Symposium*, Security'12, pages 333–348, Berkeley, CA, USA, 2012.

[59] Thomas J. Richardson, M. Amin Shokrollahi, and Rudiger L. Urbanke. Design of capacity-approaching irregular low-density parity check codes. *IEEE Transaction on Information Theory*, 47(2):619–637, February 2001.

[60] Thomas J. Richardson and Rudiger L. Urbanke. The capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding. *IEEE Transaction on Information Theory*, 47(2):599–618, February 2001.

[61] R. L. Rivest and A. Shamir. How to reuse a write-once memory. *Informaton and Control*, 55:1–19, 1982.

[62] M. Rosenblum and J. K. Ousterhout. The design and implementation of a log-structured file system. *ACM Trans. Comput. Syst.*, 10(1):26–52, feb 1992.

[63] Thomas J. Richardson Sae-Yong Chung and Rudiger L. Urbanke. Analysis of sum-product decoding of low-density parity-check codes using a gaussian approximation. *IEEE Transcations on Information Theory*, 47(2):657–670, Febuary 2001.

[64] Abdallah M. Saleh, Jun J. Serrano, and Janak H. Patel. Reliability of scrubbing recover-techniques for memory systems. *IEEE Transcations on reliability*, 39(3):114–122, April 1990.

[65] Amir Shpilka. Capacity-Achieving Multiwrite WOM Codes. *IEEE Transcation on Information Theory*, 60(3):1481–1487, March 2014.

[66] David Spleian and Jack Leil Wolf. In *The bell system technical journal*, pages 1037–1076, September 1973.

[67] I. Tal and A. Vardy. List decoding of polar codes. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, pages 1–5, Saint Petersburg, Russia, Auguest 2011.

[68] Ido Tal and Alexander Vardy. How to construct polar codes. *IEEE Transactions on Information Theory*, 59(10):6562–6582, October 2013.

[69] Jiadong Wang, Thomas Courtade, Hari Shankar, and Richard D. Wesel. Soft Information for LDPC Decoding in Flash: Mutual-Information Optimized Quantization. In *Globecom 2011*, Houston, Texas, USA, December 2011.

[70] Michael Wei, Laura M. Grupp, Frederick E. Spada, and Steven Swanson. Reliably erasing data from flash-based solid state drives. In *Proceedings of the 9th USENIX conference on File and stroage technologies (FAST'11)*, San Jose, California, 2011.

[71] Y. Wu. Low complexity codes for writing write-once memory twice. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, pages 1928–1932, Austin, Texas, June 2010.

[72] Y. Wu and A. Jiang. Position modulation code for rewriting write-once memories. *IEEE Transactions on Information Theory*, 57(6):3962–3697, June 2011.

[73] A. D. Wyner. The Wire-tap Channel. *Bell Systems Technical Journal*, 54(8):1355–1387, January 1975.

[74] Ningde Xie, Guiqiang Dong, and Tong Zhang. Using lossless data compression in data storage systems: Not for saving space. *IEEE Transactions on Computers*, 60(3):335–345, 2011.

[75] Wei Xu and Tong Zhang. A time-aware fault tolerance scheme to improve reliability of multilevel phase-change memory in the presence of significant resistance drift. *IEEE Trans. VLSI Syst.*, 19(8):1357–1367, 2011.

[76] Eitan Yaakobi, Scott Kayser, Paul H. Siegel, Alexander Vardy, and Jack K. Wolf. Codes for Write-Once Memories. *IEEE Transactions on Information Theory*, 58(9):5985–5999, September 2012.

[77] Eitan Yaakobi, Paul H. Siegel, Alexander Vardy, and Jack K. Wolf. On Codes that Correct Asymmetric Errors with Graded Magnitude Distribution. In *Proc. IEEE International Symposium on Information Theory*, pages 1021–1025, St. Petersburg, Russia, August 2011.

[78] Eitan Yaakobi, Alexander Yucovich, Gal Mor, and Gala Yadgar. When Do WOM Codes Improve the Erasure Factor in Flash Memories. In *Proc. IEEE International Symposium on Information Theory(ISIT)*, Hongkong, June 2015.

[79] Gala Yadgar, Eitan Yaakobi, and Assaf Schuster. Write Once, Get 50% Free, Saving SSD Erase Costs Using WOM Codes. In *Proceedings of the 10th USENIX conference on File and Storage Technologies (FAST'15)*, pages 257–271, 2015.

[80] Hongchao Zhou, Anxiao Jiang, and Jehoshua Bruck. Error-Correcting Schemes with Dynamic Thresholds in Nonvolatile Memories. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, pages 2143–2147, St. Petersburg, Russia, August 2011.