Average case reductions for Subset Sum and Decoding of Linear Codes

by

Geneviève Arboit

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Average case reductions for Subset Sum and Decoding of Linear Codes

Geneviève Arboit

Master of Science

Graduate Department of Computer Science

University of Toronto

1999

In a 1996 paper, R. Impagliazzo and M. Naor show two average case reductions for the Subset Sum problem (**SS**). We use similar ideas to obtain stronger and additional such reductions for **SS**. Furthermore, we use modifications of these ideas to obtain similar reductions for the Decoding of Linear Codes problem (**DLC**). The theorems give further evidence that the hardest case for Average case **SS** is when the number of integers is equal to their length. For Average case **DLC**, the theorems give evidence that the hardest case is when the dimension of the code is equal to the channel capacity times the length of the words.

Average case **SS** and **DLC** hardness assumptions can be used to obtain one-way functions, pseudorandom generators, and secure private-key cryptography.

Acknowledgments

I dedicate this work to my parents, who have always supported me in more ways than I can enumerate.

I am thankful to my second reader, Michel Molloy, as well as to Daniel Panario, who pointed me to combinatorics references and techniques, which directly applied to the first parts of Section 5.3, as well as, generally, to subsequently obtained results. I also thank my supervisor, Charles Rackoff for showing me how much more than I thought there is to probability theory.

I thank the following people for some very useful conversations: Micah Adler, Faith Fich, Adreas Goerdt, Valentine Kabanets, Daniele Micciancio, Lucia Moura, François Pitt, Amit Sahai, Steven Tanny.

I thank the following people for their precious morale support: Albert Camus, Love, Kerry Khoo, the Wolf, Daniel Nyborg, Patti, Nataša Przulj, Alex Vasilescu, Ning Wu.

I thank the following people for encouraging me to undertake graduate studies: Syed Ali, Elie Cohen, Joel Hillel, Clement Lam, Daniel Nadeau, Geza Szamosi.

I am grateful to the Natural Sciences and Engineering Research Council for their generous assistance through a NSERC PGS A grant.

Contents

Chapter 1

Introduction

1.1 Computational complexity theory

Computational complexity theory analyses the time efficiency and resource costs, including space and possibly random bits, of problem solving. In particular, modern cryptography is based on a gap between the efficiency of algorithms for legitimate users, and the computational infeasibility for an adversary trying to extract information. Consequently, a necessary condition for the existence of secure encryption schemes is that $\mathcal{NP}$ is not contained in $\mathcal{BPP}$, and thus that $\mathcal{P} \neq \mathcal{NP}$.

However, that $\mathcal{P} \neq \mathcal{NP}$ implies only that there exist encryption schemes that are hard to break in the worst case. It may be the case that an $\mathcal{NP}$-complete problem can not be solved efficiently only on a few artificially difficult instances. In fact, any $\mathcal{NP}$-complete problem could be easy to solve almost always, and particular examples of this can be constructed [Wil84]. It is in this sense that worst case complexity theory gives no information about average instances of a problem.

Consequently, the subject matter of this thesis is related to considering average case versions of some $\mathcal{NP}$-complete problems which are not known to be easy to solve. It is often the case that if a problem is hard on the average, then it can be used to obtain a "one-way function". Informally, these are functions that are efficiently computable, while being hard to invert on average.

Definition 1 [One-way function] *A function $f : \{0,1\}^* \mapsto \{0,1\}^*$ is one-way if the following two conditions hold:*

1. *f is easy to compute: There exists a deterministic polynomial time algorithm, A, so that on input x algorithm A outputs $f(x)$, i.e. $A(x) = f(x)$.*
2. *f is hard to invert: For every probabilistic polynomial-time algorithm, I, and for x uniformly selected in $\{0,1\}^n$*

$$\Pr_x[I(f(x), 1^n) \in f^{-1}(f(x))] < \frac{1}{n^c}$$

   for every constant c and for sufficiently large n.

The first property is intended to be used by "legitimate users", and the second, for the counteraction of potential "adversaries". It can be proven that from one-way functions, we can construct pseudorandom generators, secure private-key cryptography, and secure digital signature schemes [Lub96, Lectures 9-10, 11-12, and 17]. However, it is not known whether one-way functions exist, since their existence implies $\mathcal{P} \neq \mathcal{NP}$.

There exists three types of conjectures on how one-way functions may be obtained. The first type is number theoretic. An assumption on the intractability of factorization, and another, on that of discrete logarithm, can be used to obtain one-way functions. While these assumptions are most commonly used to obtain secure public-key cryptography, the known secure private-key cryptography schemes that they yield are too inefficient to be used in practice. The second way is to conjecture that given algorithms directly produce pseudo-random generators. A one-way function can be derived from the assumption that the encryption algorithm named "Digital Encryption Standard" (DES) is a pseudo-random generator. This derived one-way function is used in the UNIX password scheme.

Finally, we try to obtain one-way functions from hardness assumptions on average case versions of $\mathcal{NP}$-complete problems. This thesis concentrates on this third way, and in particular on the average case Subset Sum (**SS**) problem and average case Decoding of Linear Codes (**DLC**) problem. The conjecture that these functions (with certain choices of parameters) are one-way is supported by the failure of extensive research to produce efficient adversary algorithms [Gol98, Section 2.2.4].

1.2 Subset Sum problem

Informally, the Subset Sum (**SS**) problem is the following. A set of n integers, each of m bits in length, and a sum, modulo 2^m, of some of these integers are given. The problem is to find a selection of the integers that adds to the given sum, modulo 2^m.

An $n \times m$ matrix A represents the n integers of m bits each. Each row A_i represents one of the m-bit integers, also called "weights". The given sum α is m-bit long, and the set, consisting of the original selection of integers, is represented by its n-bit long incidence vector x. The Subset Sum problem, parameterized by the number of integers n and their length, m, is denoted by $\mathbf{SS}(n, m)$.

Definition 2 **[Scalar product for $\mathbf{SS}(n, m)$]** *Let $x \in \{0,1\}^n$ be a $1 \times n$ row vector and $A \in \{0,1\}^{nm}$ be an $n \times m$ matrix $A \in \{0,1\}^{nm}$. Then the scalar product of x and A is*

$$x \odot A = \sum_{i=1}^{n} x_i A_i \bmod 2^m$$

1.2.1 Worst case SS

The definitions of the two worst case versions of the **SS** problem follow, as well as what is known on the hardness of their solution.

Definition 3 [Decisional SS(n, m)] *Given $A_1, ..., A_n \in \{0,1\}^m$ and $\alpha \in \{0,1\}^m$, determine whether there is an $x \in \{0,1\}^n$ such that $\alpha = x \odot A$.*

Definition 3 is not identical to the definition of the Subset Sum problem given in [GJ79, $\mathcal{NP}$-complete problem SP13], but the two definitions are equivalent (the latter is clearly reducible to the former, while the former is also reducible to the latter, with the help of linear time guessing of some extra carry bits). Worst case Decisional **SS** is one of the original problems that was proven to be $\mathcal{NP}$-complete by Karp [Kar72].

Theorem 4 [GJ79, $\mathcal{NP}$-complete problem SP13] *The Decisional* **SS** *problem is $\mathcal{NP}$-complete.*

Definition 5 [Functional SS(n, m)] *Given $A_1, ..., A_n \in \{0,1\}^m$ and $\alpha \in \{0,1\}^m$, find, if such exists, an $x \in \{0,1\}^n$ such that $\alpha = x \odot A$.*

Since Decisional **SS** is $\mathcal{NP}$-complete, Functional **SS** is reducible to it. Clearly, Functional **SS** is harder, so it is also $\mathcal{NP}$-hard.

1.2.2 Average case SS

Average case **SS** is the first of the two problems in whose complexity we are interested. We will analyze average case reductions for **SS**. Our aim is to obtain average case reductions between versions of **SS** with a choice of parameters, and **SS** with another choice of parameters.

The definition of the average case version of the problem follows below, as well as a conjecture on the hardness of its solution. The following notation is used when a random variable uniformly samples a given space. Let x be a random variable. Then $x \in_U S$ means that x is chosen uniformly within the domain S.

Definition 6 [Average case SS(n, m)] *Let $A_1, ..., A_n \in_U \{0,1\}^m$ and $x \in_U \{0,1\}^n$, and then let $\alpha = x \odot A$. Given $A_1, ..., A_n$ and α, find $x' \in \{0,1\}^n$ satisfying $\alpha = x' \odot A$.*

In order to conjecture that Average case $\mathbf{SS}(n, m)$ is hard, the statement of the problem needs to be dependent on only one parameter, since the conjecture should be of the form "For every polynomial time probabilistic algorithm, the probability of solving $\mathbf{SS}(n, m)$ is small as a function of n". So let the parameter m be a parameter function that depends on n. In other words, the value of m is denoted by $g(n)$.

Conjecture 7 [Hardness for Average case SS w.r.t. to the parameterization $m = g(n)$] *Let I be a probabilistic polynomial time algorithm of the following form. Let $n \in \mathbb{N}^+$ and let $m = g(n)$. Algorithm I that takes as input a pair consisting of an $n \times m$ matrix and an m-bit row vector, and returns as output an n-bit row vector. Define the probability $p(n)$ via the following experiment.*

Let $A_1, ..., A_n$ be m-bit integers, chosen uniformly and independently, and let x be an n-bit row vector, chosen uniformly and independently. Then let $\alpha = x \odot A$ and define $p(n) = \Pr[I(A, \alpha) \odot A = \alpha]$.

Then for every constant c and for sufficiently large n, it holds that $p(n) < \frac{1}{n^c}$.

It is easy to see that this conjecture is equivalent to saying that the following function is one-way (as defined by Definition 1), assuming that the same parameterization $m = g(n)$ is used.

Definition 8 [Candidate one-way function from SS w.r.t. to the parameterization $m = g(n)$] *Let $n \in \mathbb{N}^+$ and let $m = g(n)$. Let A be an $n \times m$ bit matrix and x an n-bit row vector. Define the function $f_{SS,g}$ as*

$$f_{SS,g}(A, x) = (A, x \odot A)$$

mapping $nm + n$ bits to $nm + m$ bits.

Solved instances The general Average case $\mathbf{SS}(n, m)$ problem is solvable for special dimensions, which give the problem a special structure.

The problem is said to have "low density", if $m > n$. Informally, if $m \in \Omega(n^2)$, there is an polynomial time algorithm solving Average case $\mathbf{SS}(n, m)$ with high probability [CJL+92]. It is an improvement over the algorithms in [LO85] and [Bri83]. All such algorithms use a reduction of **SS** into a shortest vector in a lattice approximation problem.

The problem is said to have "high density", if $m < n$. For very small m, that is, informally for $m \in \mathcal{O}(\log n)$, there is a dynamic programming algorithm for Worst case $\mathbf{SS}(n, m)$ that runs in time $\mathcal{O}(n^2)$ [TS86]. If $m < \frac{1}{2}\log(n)$, there are even more efficient algorithms [GM91]. For $m = n$, the best known attack takes time $\mathcal{O}(2^{\frac{n}{2}})$ and space $\mathcal{O}(2^{\frac{n}{4}})$ [SS79].

Past results and this thesis One of the few places in which results on average case **SS** are found is [IN96]. Impagliazzo and Naor show that, supposing that for some $c > 1$ and for $n \in \mathbb{N}^+$ sufficiently large, $g(n) \geq cn$, if $f_{SS,g}$ is one-way, then $f_{SS,g}$ is also a pseudorandom number generator [Theorem 2.2]; supposing that for some $c < 1$ and for $n \in \mathbb{N}^+$ sufficiently large, $g(n) \leq cn$, if $f_{SS,g}$ is one-way, then $f_{SS,g}$ is also a family of universal one-way hash functions [Theorem 3.1].

They are also interested in average case reductions between different $\mathbf{SS}(n, m)$ problems. The subject matter of this thesis is such average case reductions. We will ask: *if a solution can be found with some probability, for given dimensions of the problem, then with what probability can a solution be found for other dimensions?* Impagliazzo and Naor show the following [IN96, Proposition 1.2].

1. Suppose $g(n) \leq g'(n)$ for all n, and $cn \leq g(n)$ for some $c > 1$ and n sufficiently large. Then hardness w.r.t. to the parameterization $m' = g'(n)$ implies hardness w.r.t. to the parameterization $m = g(n)$.

2. Suppose $g(n) \geq g'(n)$ for all n, and $cn \geq g(n)$ for some $c < 1$ and n sufficiently large. Then hardness w.r.t. to the parameterization $m' = g'(n)$ implies hardness w.r.t. to the parameterization $m = g(n)$

Rackoff stated a stronger form of these two theorems [Rac98], which follow.

1. Suppose $n \leq g(n) \leq g'(n)$ for all n. Then hardness w.r.t. to the parameterization $m' = g'(n)$ implies hardness w.r.t. to the parameterization $m = g(n)$.

2. Suppose $n \geq g(n) \geq g'(n)$ for all n. Then hardness w.r.t. to the parameterization $m' = g'(n)$ implies hardness w.r.t. to the parameterization $m = g(n)$.

Instead of parameterizing m as a function $g(n)$ of n, we can consider the parameterization of n as a function $h(m)$ of m. Just as we can conjecture the Hardness for Average case **SS** w.r.t. to the parameterization $m = g(n)$, we can conjecture Hardness for Average case **SS** w.r.t. to the parameterization $n = h(m)$, and similarly obtain an equivalent candidate one-way function. The relationship between these two types of parameterization will be discussed in Section 4.1.1.

Correspondingly, Rackoff stated these additional two theorems [Rac98], which follow.

1. Suppose $h'(m) \leq h(m) \leq m$ for all m. Then hardness w.r.t. to the parameterization $n' = h'(m)$ implies hardness w.r.t. to the parameterization $n = h(m)$.

2. Suppose $h'(m) \geq h(m) \geq m$ for all m. Then hardness w.r.t. to the parameterization $n' = h'(m)$ implies hardness w.r.t. to the parameterization $n = h(m)$.

We will prove all four above stated theorems in Chapter 2. Each of the theorems is proven by a reduction, which are, with abusive notation, the following.

1. If $n \leq m \leq m'$, then $\mathbf{SS}(n, m') \propto_{av} \mathbf{SS}(n, m)$.

2. If $n \geq m \geq m'$, then $\mathbf{SS}(n, m') \propto_{av} \mathbf{SS}(n, m)$.

3. If $n' \leq n \leq m$, then $\mathbf{SS}(n', m) \propto_{av} \mathbf{SS}(n, m)$.

4. If $n' \geq n \geq m$, then $\mathbf{SS}(n', m) \propto_{av} \mathbf{SS}(n, m)$.

The meaning of this notation has to be explained, since the statements are asymptotic, although they do not appear to be. The simplest way is to give an explanation by example.

We mean by (1) that there is a probabilistic oracle algorithm J with the following properties. Algorithm J has for inputs n, m and m' such that $n \leq m \leq m'$, and a deterministic oracle I for (average case) $\mathbf{SS}(n,m)$. We define p to be the probability that the correct solution to $\mathbf{SS}(n,m)$ is returned, when the inputs are distributed "properly" for the Average case $\mathbf{SS}(n,m)$. That is, for A, an $n \times m$ bit matrix chosen uniformly and independently, and for an n-bit row vector x chosen uniformly and independently, let $\alpha = x \odot A$. Then p is the probability that $I(A,\alpha)$ returns an n-bit vector such that $I(A,\alpha) \odot A = \alpha$.

Algorithm J uses oracle I in attempting to solve instances of (average case) $\mathbf{SS}(n,m')$. Algorithm J runs in time polynomial in $m' = \max(n,m,m',n')$. Let p' be the probability that J solves "properly" distributed instances of $\mathbf{SS}(n,m')$. That is, for for B, an $n \times m'$ bit matrix chosen uniformly and independently, and for an n-bit row vector y chosen uniformly and independently, let $\beta = y \odot B$. Then p' is the probability that $J(B,\beta,n,m,m',I)$ returns an n-bit vector such that $J(B,\beta,n,m,m',I) \odot B = \beta$. We will show that, for $m' = \max(n,m,m',n')$ sufficiently large, $p' \geq poly\left(\frac{p^2}{\max(n,m,m',n')}\right)$.

The four theorems above imply two simpler statements, which follow.

1. Hardness w.r.t. to the parameterization $m' = g'(n)$ implies hardness w.r.t. to the parameterization $m = n$.

2. Hardness w.r.t. to the parameterization $n' = h'(m)$ implies hardness w.r.t. to the parameterization $n = m$.

Statement (1) is implied by the theorems of [IN96, Proposition 1.2]. The reason why these statements are not equivalent to one another is subtle and will be discussed in Section 4.1.1. In particular, one might think that if $g = \lfloor h^{-1} \rfloor$ or $h = \lfloor g^{-1} \rfloor$, then (1) and (2) are equivalent, but they are not.

The above statements (1) and (2) can be summarized by saying that if Average case **SS** is hard for some dimensions $(n, g(n))$, then it is hard for (n, n); if Average case **SS** is hard for some dimensions $(h(m), m)$, then it is hard for (m, m). One might think that this means that $\mathbf{SS}(n, m)$ is hardest when $n = m$, i.e. when the number of integers is equal to their length. However, what this means is not clear and is actually misleading.

All we have is that certain reductions where the probability of solution is never decreased "too much" by using solutions for instances where $m = n$. One might think that, while $\mathbf{SS}(n, n)$ is hard, the larger n is, the harder it gets, but this is not clear. For example, we do not know how to compare $\mathbf{SS}(n^2, n^2)$ and $\mathbf{SS}(n, n)$. This open question will be discussed in Section 4.1.2.

1.3 Decoding of Linear Codes problem

The purpose of error-correcting codes is to correct errors in messages transmitted on noisy communication channels. We are concerned with binary symmetric channels, where the probability that the channel makes a mistake, i.e. that a bit is flipped, is a constant ε. An additional assumption is made: the errors, for each bit, are independent [1]. The transmitting end attempts to protect messages by adding some redundancy to them. This allows the receiving end to make some corrections.

From now on, we consider ε to be a fixed constant fraction such that $0 < \varepsilon < \frac{1}{2}$. If $\varepsilon > \frac{1}{2}$, then interchanching the names of the received symbols changes this to a binary channel with $\varepsilon < \frac{1}{2}$ [MS77, Section 1.1, Problem (3)]. If $\varepsilon = \frac{1}{2}$, then no communication is possible. This is a special case of Shannon's bound (further discussed in Section 1.3.2) where the capacity of the channel is zero, which occurs when the error probability $\varepsilon = \frac{1}{2}$. Informally, the rate at which a message can be sent through a channel involves the

[1] In concrete engineering situations, the errors often are, on the contrary, correlated.

expansion of the message by a factor 1/(capacity of a channel with error probability ε). So if the capacity is zero, then messages are not transmittable. If $\varepsilon = 0$, then all the results we later present still hold, though they are not interesting because decoding can trivially be done efficiently via linear algebra. The question of not requiring ε to be constant will be addressed in Section 4.2.2.

One common, simple, and efficient method for encoding and decoding, is that of "linear codes" [MS77], which are determined by $n \times m$ bit matrices A. We will refer to any such A as a linear code [2] of word length m and dimension n. The ratio $\frac{n}{m}$ is the rate of the code. For a fixed $n \times m$ bit A, an n-bit message x will be transmitted as the m-bit codeword xA. Some bits of the received word are flipped, which correspond to the one-entries of an m-bit row vector e. The received word is equal to $xA + e$.

Informally, the Decoding of Linear Codes (**DLC**) problem is the following. We are given an $n \times m$ bit matrix A, and a bitwise sum of some its rows, with at most a constant fraction ε of its bits flipped. The problem is to find a selection of A's rows that bitwise adds to the given sum, with at most a fraction ε of the sum's bits flipped. The Decoding of Linear Codes problem, parameterized by the dimensions of the matrices A's considered, n and m, is denoted by **DLC**(n, m).

Definition 9 [Scalar product for DLC(n, m)] *Let $x \in \{0,1\}^n$ be a $1 \times n$ row vector and $A \in \{0,1\}^{nm}$ be an $n \times m$ bit matrix. Then the scalar product of x and A is*

$$xA = \left(\sum_{j=1}^{n} x_j A_{j1} \bmod 2, ..., \sum_{j=1}^{n} x_j A_{jm} \bmod 2 \right)$$

[2] To take n to be the number of rows and m to be the number of columns follows the conventions established for **SS**(n, m) in Section 1.2, which follow the conventions in the literature cited there. However, most error-correcting code references use k and n instead of n and m, respectively.

1.3.1 Worst case DLC

The definitions of the two worst case versions of the the problem follow, as well as what is known on the hardness of their solution. The Hamming weight function $w(e)$ denotes the number of one's in a bit vector e.

Definition 10 [Decisional DLC(n, m)] *Given $A_1, ..., A_n \in \{0,1\}^m$ and $\alpha \in \{0,1\}^m$, determine whether there is an $x \in \{0,1\}^n$ and an $e \in \{0,1\}^m$ such that $\alpha = xA + e$ and $w(e) \leq \lfloor \varepsilon m \rfloor$.*

Definition 10 is equivalent, although not identical, to the definition of the Decoding of Linear Codes problem given in [GJ79, $\mathcal{NP}$-complete problem MS7]. This last definition utilizes duality properties of linear codes [MS77, Section 1.8]. Worst case Decisional **DLC** was proven to be $\mathcal{NP}$-complete in [BMv78].

Theorem 11 [GJ79, $\mathcal{NP}$-complete problem MS7] *The Decisional* **DLC** *problem is $\mathcal{NP}$-complete.*

Definition 12 [Functional DLC(n, m)] *Given $A_1, ..., A_n \in \{0,1\}^m$ and $\alpha \in \{0,1\}^m$, find, if such exist, an $x \in \{0,1\}^n$ and an $e \in \{0,1\}^m$ such that $\alpha = xA + e$ and $w(e) \leq \lfloor \varepsilon m \rfloor$.*

Since Decisional **DLC** is $\mathcal{NP}$-complete, Functional **DLC** is reducible to it. Clearly, Functional **DLC** is harder, so it is also $\mathcal{NP}$-hard.

Good codes The notion of "minimum distance of a code" permits to classify codes as good or bad. Because an error vector of lesser weight is more likely than another with greater weight [MS77, Section 1.3, Problem (7)], when the receiving end decodes a transmitted codeword, it will always pick the error vector which has the least weight possible [3]. This is called "nearest neighbor decoding" or "maximum likelihood decoding"

[3]This strategy is best assuming that a transmission is done only once. Schemes using some error detection and retransmission are different.

[MS77, Section 1.3, p.11].

Definition 13 [Minimum distance of a code] *Let an $n \times m$ bit matrix A be a code. Then its minimum distance d is the minimum Hamming distance between its codewords.*

$$d = \min_{x \neq x'} w(xA - x'A)$$

If A has linearly dependent rows, i.e. A does not have full rank, then for $x \neq x'$, it is possible that $xA = x'A$, so that the minimum distance of A is zero. Note that in the literature on error-correcting codes, A always has full rank. Definition 13 is equivalent to the following [MS77, Section 1.3, Theorem 1].

Definition 14 [Equivalent minimum distance of a code] *Let an $n \times m$ bit matrix A be a code. Then its minimum distance is the minimum Hamming weight of any nonzero codeword.*

$$d = \min_{x \neq 0} w(xA)$$

If A has minimum distance d, then it is possible to decode uniquely for up to $\lfloor \frac{d-1}{2} \rfloor$ errors [MS77, Section 1.3, Theorem 2]. If there are more errors, then the received word may or may not be closer to some other codeword than to the correct one. If it is, the decoder will be deceived into outputting the wrong codeword, yielding a "decoding error". For a code to be good in practice, it is required that this rarely happens [MS77, Section 1.3, p.11].

1.3.2 Average case DLC

Average case **DLC** is the second of the two problems in whose complexity we are interested. "One of the most outstanding open problems in the area of error correcting codes is that of presenting efficient decoding algorithms for random linear codes. Of particular interest are random linear codes with constant information rate which can correct a constant fraction of errors." [Gol98, Section 2.2.4] We will analyze average case reductions for **DCL**. Our aim is to obtain average case reductions between versions of **DLC** with a choice of parameters, and **DLC** with another choice of parameters.

Bounds for the achievable rate

The concept of entropy was defined by Shannon in [Sha48], where two fundamental theorems were first published. Shannon's First Theorem is concerned with the sending of information through noiseless channels. Shannon's Second Theorem is concerned with the sending of information through noisy channels. It contains two parts, which we refer to as an upper and a lower bound. Informally, the upper bound guarantees that most (not necessarily linear) codes with rate bounded above by capacity have arbitrarily small decoding error probability. Informally, the lower bound implies that codes with rate bounded below by capacity have a best decoding error probability that is bounded away from 0, as as the word length m increases. The birth of information theory can be dated from Shannon's publication of this fundamental paper, which can also be found as the second part of the book [SW64].

Rather than stating Shannon's Second Theorem, we will state stronger versions of the above mentioned upper and lower bounds as Theorems 18 and 19, respectively. Informally, Theorem 18 guarantees that most *linear* codes with rate bounded above by capacity have arbitrarily small decoding error probability. Informally, Theorem 19 states that codes with rate bounded below by capacity have a best decoding error probability that approaches 1, as the word length m increases. This stronger form of the lower bound was proven in [Wol60].

Shannon also pioneered the use of entropy in cryptography [Sha49].

Definition 15 [Entropy] *The entropy function is* $H(\varepsilon) = -\varepsilon \log(\varepsilon) - (1-\varepsilon)\log(1-\varepsilon)$, *where the logarithms are taken in base* 2.

Definition 16 **[Capacity]** *The capacity function is* $C(\varepsilon) = 1 - H(\varepsilon)$.

The following theorems show that random linear codes are likely to be good codes, as defined at the end of Section 1.3.1. Nevertheless, that a code is good does not imply the existence of an *efficient* decoding algorithm.

Theorem 17 **[An upper bound: Generalized Gilbert-Varshamov Bound]** *Let* $R < C(\varepsilon)$. *Then "almost all"* $n \times m$ *linear codes of rate* $\leq R$ *have minimum distance* $\geq \varepsilon m$, *in the following sense. The fraction of* $n \times m$ *linear codes of rate* $\leq R$ *that have smaller minimum distance tends to 0 as* m *tends to* ∞.

The proof of this theorem can be found in [TV91, p. 77].

The capacity of the channel $C(\varepsilon)$ will play an important role in our results. In fact, the statement that m is about equal to $\frac{n}{C(\varepsilon)}$ will play the same role as the expression $m = n$ does in the results for Average case **SS**, as stated at the end of Section 1.2.2.

As mentioned at the end of the previous section, it is possible to achieve perfect maximum likelihood decoding with a code of minimum distance $2\varepsilon m + 1$ when no more than εm errors occur, which is the expected number of errors when the probability that the channel makes an error is ε.

Nevertheless, Theorem 18 below guarantees that most linear codes of rate bounded above by $C(\varepsilon)$, can be decoded arbitrarily close to perfectly, for a channel with bit error probability ε. Informally, this says that most such codes act as if they had minimum distance $\geq 2\varepsilon m + 1$, on average.

Theorem 18 **[An upper bound: Coding Theorem for linear codes]** *Let* $R < C(\varepsilon)$. *Then "almost all"* $n \times m$ *linear codes of rate* $\leq R$ *have a small probability of decoding error, in the following sense.*

Consider a random linear code and a binary symmetric channel of capacity $C(\varepsilon)$. *Assume maximum likelihood decoding is used. Consider a random encoded message and a random error vector. Then the probability (over the choice of code, message, and error)* *that the wrong codeword is decoded tends to 0 as m tends to ∞.*

A statement and an explanation of Theorem 18 can be found in [MS77, Section 1.6], and a proof for a random linear code with the binary symmetric channel in [Gal68, Section 6.2].

Theorem 19 [A lower bound: Impossibility of reliable coding] *Let $R > C(\varepsilon)$. In every binary symmetric channel of capacity $C(\varepsilon)$, whenever a (not necessarily linear) code with codewords of length m has rate $\geq R$, then the code tends to be totally unreliable for any decoding strategy, i.e. its error probability tends to 1, with increasing m.*

Theorem 19 was proven by Wolfowitz [Wol60].

Average case DCP

The definition of the average case version of the **DCP** problem follows, as well as a conjecture on the hardness of its solution. The notation $\in_U$ is used as defined in Section 1.2.2.

Definition 20 [Average case DLC(n, m)] *Let $A_1, ..., A_n \in_U \{0,1\}^m$, $x \in_U \{0,1\}^n$, and $e \in_U \{s \in \{0,1\}^m \;\&\; w(s) \leq \lfloor \varepsilon m \rfloor\}$, and then let $\alpha = xA + e$. Given $A_1, ..., A_n$, and α, find $x' \in \{0,1\}^n$ and $e' \in \{0,1\}^m$ satisfying $\alpha = x'A + e'$ and $w(e') \leq \lfloor \varepsilon m \rfloor$.*

As in Section 1.2.2, in order to conjecture that Average case **DLC**(n, m) is hard, the statement of the problem needs to be dependent on only one parameter, since the conjecture should be of the form "For every polynomial time probabilistic algorithm, the probability of solving **DLC**(n, m) is small as a function of n". So let the value of the parameter m be denoted by $g(n)$.

Conjecture 21 [Hardness for Average case DLC w.r.t. to the parameterization $m = g(n)$] *Let I be a probabilistic polynomial time algorithm of the following form. Let $n \in \mathbb{N}^+$ and let $m = g(n)$. Algorithm I that takes as input a pair consisting of an $n \times m$ matrix and an m-bit row vector, and returns as output a pair consisting of an*

n-bit row vector and an m-bit row vector. Define the probability $p(n)$ via the following experiment.

Let $A_1, ..., A_n$ be m-bit integers, chosen uniformly and independently, let x be an n-bit row vector, chosen uniformly and independently. and let e be an m-bit row vector, chosen uniformly and independently among all such vectors with Hamming weight no more than $\lfloor \varepsilon m \rfloor$. Then let $\alpha = xA + e$ and $(x', e') = I(A, \alpha)$. Define $p(n) = \Pr[x' \odot A + e' = \alpha]$.

Then for every constant c and for sufficiently large n, it holds that $p(n) < \frac{1}{n^c}$.

It is easy to see that this conjecture is equivalent to saying that the following function is one-way (as defined by Definition 1), assuming that the same parameterization $m = g(n)$ is used.

Definition 22 [Candidate one-way function from DLC w.r.t. to the parameterization $m = g(n)$] *Let $n \in \mathbb{N}^+$ and let $m = g(n)$. Let A be an $n \times m$ bit matrix, x an n-bit row vector, and $e(i)$ an m-bit row vector such that $w(e(i)) \leq \lfloor \varepsilon m \rfloor$. The parameter i indexes the $\sum_{j=0}^{\lfloor \varepsilon m \rfloor} \binom{m}{j}$ error vectors in a natural way. Define the function $f_{code,g}$ as*

$$f_{code,g}(A, x, i) = (A, xA + e(i))$$

mapping $nm + n + \log\left(\sum_{j=0}^{\lfloor \varepsilon m \rfloor} \binom{m}{j}\right)$ bits to $nm + m$ bits.

Solved instances Conjecture 21 seems reasonable when $\varepsilon = \frac{1}{4}$ and m is about $\frac{1}{C(\varepsilon)} n$. It is certainly not true for some other values of m and n. For instance, the problem is easy when $m \leq n$ or, for a constant $c > 0$, when $m \geq 2^{n/c}$.

If $m \leq n$, then Average case **DLC** can be solved via linear algebra with constant probability greater than 0.28. This is approximatively the probability that a random $n \times n$ matrix has full rank [BvH82, Lemma 2]. This probability is greater for $n \times m$ matrix with $m < n$.

If for a constant $c > 0$, it holds that $m \geq 2^{n/c}$, then Worst case **DLC** can be solved in polynomial time. Going through all possible 2^n solutions requires time at most m^c.

A survey of positive and negative results on the hardness of decoding can be found in [Var97, Section 2.2], in particular for families of linear codes, that is, for linear codes with additional properties that are useful in practice. From this survey, it stands out that worst case "polynomial time maximum-likelihood decoding algorithms are not known for any specific family of useful codes". However, trellises (a type of graph) can be used to implement worst case "randomized 'sequential' decoding algorithms (for linear codes) [Fan63, For74], whose performance is close to maximum-likelihood and whose running time is polynomial with appreciably high probability."

Results in this thesis The subject matter of this thesis also includes average case reductions for **DLC**. As for **SS**, we will ask: *if a solution can be found with some probability, for given dimensions of the problem, then with what probability can a solution be found for other dimensions?* Rackoff conjectured four theorems which correspond to the ones enumerated for **SS**, at the end of Section 1.2.2. We will prove them in Chapter 3. The first two theorems, corresponding to the above parameterizations in n are the following.

1. Suppose $g(n) \leq g'(n)$ for all n, and $g(n) \geq \lceil \frac{n}{C(\varepsilon)} \rceil$ for n sufficiently large, so that $\lceil \frac{n}{C(\varepsilon)} \rceil \leq g(n) \leq g'(n)$. Then hardness w.r.t. to the parameterization $m' = g'(n)$ implies hardness w.r.t. to the parameterization $m = g(n)$.

2. Suppose $g(n) \geq g'(n)$ for all n, and $g(n) \leq \lfloor \frac{n}{C(\varepsilon)} \rfloor$ for n sufficiently large, so that $\lfloor \frac{n}{C(\varepsilon)} \rfloor \geq g(n) \geq g'(n)$. Then hardness w.r.t. to the parameterization $m' = g'(n)$ implies hardness w.r.t. to the parameterization $m = g(n)$.

Instead of parameterizing m as a function $g(n)$ of n, we can consider the parameterization of n as a function $h(m)$ of m. Just as we conjectured the Hardness for Average case **DLC** w.r.t. to the parameterization $m = g(n)$, we conjecture Hardness for Average case **DLC** w.r.t. to the parameterization $n = h(m)$, and similarly obtain an equivalent candidate one-way function. The relationship between these two types of parameterization will be discussed in Section 4.2.1.

Correspondingly, Rackoff conjectured these additional two theorems.

1. Suppose $h'(m) \leq h(m)$ for all m, and $h(m) \leq \lfloor C(\varepsilon)m \rfloor$ for m sufficiently large, so that $h'(m) \leq h(m) \leq \lfloor C(\varepsilon)m \rfloor$. Then hardness w.r.t. to the parameterization $n' = h'(m)$ implies hardness w.r.t. to the parameterization $n = h(m)$.

2. Suppose $h'(m) \geq h(m)$ for all m, and $h(m) \geq \lceil C(\varepsilon)m \rceil$ for m sufficiently large, so that $h'(m) \geq h(m) \geq \lceil C(\varepsilon)m \rceil$. Then hardness w.r.t. to the parameterization $n' = h'(m)$ implies hardness w.r.t. to the parameterization $n = h(m)$.

We will prove all four above stated theorems in Chapter 3. Each of the theorems is proven by a reduction, which are, with abusive notation, the following.

1. If $\lceil \frac{n}{C(\varepsilon)} \rceil \leq m \leq m'$ then $\mathbf{DLC}(n, m') \propto_{av} \mathbf{DLC}(n, m)$.

2. If $\lfloor \frac{n}{C(\varepsilon)} \rfloor \geq m \geq m'$ then $\mathbf{DLC}(n, m') \propto_{av} \mathbf{DLC}(n, m)$.

3. If $n' \leq n \leq \lfloor C(\varepsilon)m \rfloor$, then $\mathbf{DLC}(n', m) \propto_{av} \mathbf{DLC}(n, m)$.

4. If $n' \geq n \geq \lceil C(\varepsilon)m \rceil$, then $\mathbf{DLC}(n', m) \propto_{av} \mathbf{DLC}(n, m)$.

The meaning of this notation has to be explained, since the statements are asymptotic, although they do not appear to be. The simplest way is to give an explanation by example.

We mean by (1) that there is a probabilistic oracle algorithm J with the following properties. Algorithm J has for inputs n, m and m' such that $\lfloor \frac{n}{C(\varepsilon)} \rfloor \leq m \leq m'$, and a deterministic oracle I for average case $\mathbf{DLC}(n, m)$. We define p to be the probability that the correct solution to $\mathbf{DLC}(n, m)$ is returned, when the inputs are distributed "properly" for the average case $\mathbf{DLC}(n, m)$. That is, for A, an $n \times m$ bit matrix chosen uniformly and independently, for an n-bit row vector x chosen uniformly and independently, and for an m-bit row vector e chosen uniformly and independently among all such vectors with Hamming weight no more than $\lfloor \varepsilon m \rfloor$, let $\alpha = xA + e$. Then p is the probability that $I(A, \alpha)$ returns a pair (x', e'), consisting of an n-bit row vector x' and an m-bit row vector e', such that $x'A + e' = \alpha$.

Algorithm J uses oracle I in attempting to solve instances of (average case) **DLC**(n, m'). Algorithm J runs in time polynomial in $m' = \max(n, m, m', n')$. Let p' be the probability that J solves "properly" distributed instances of **DLC**(n, m'). That is, for for B, an $n \times m'$ bit matrix chosen uniformly and independently, for an n-bit row vector y chosen uniformly and independently, and for an m-bit row vector f chosen uniformly and independently among all such vectors with Hamming weight no more than $\lfloor \varepsilon m' \rfloor$, let $\beta = y \odot B + f$. Then p' is the probability that $J(B, \beta, n, m, m', I)$ returns a pair (y', f') , consisting of an n-bit row vector y' and an m'-bit row vector f', such that $y' \odot B + f' = \beta$. We will show that, for $m' = \max(n, m, m', n')$ sufficiently large, $p' \geq poly\left(\frac{p}{\max(n,m,m',n')}\right)$.

The four theorems above imply two simpler statements, which follow.

1. Hardness w.r.t. to the parameterization $m' = g'(n)$ implies hardness w.r.t. to the parameterization $m = \lfloor \frac{n}{C(\varepsilon)} \rfloor$.

2. Hardness w.r.t. to the parameterization $n' = h'(m)$ implies hardness w.r.t. to the parameterization $n = \lfloor C(\varepsilon) m \rfloor$.

Note that the choice of floors instead of ceilings in the two statements above is arbitrary. We will discussed this issue in more depth in Section 4.2.2.

As in the case of **SS**, the reason why these statements are not equivalent to one another is subtle and will be discussed in Section 4.2.1. In particular, one might think that if $g = \lfloor h^{-1} \rfloor$ or $h = \lfloor g^{-1} \rfloor$, then (1) and (2) are equivalent, but they are not.

The above statements (1) and (2) can be summarized by saying that if Average case **DLC** is hard for some dimensions $(n, g(n))$, then it is hard for $(n, \lfloor \frac{1}{C(\varepsilon)} n \rfloor)$; if Average case **SS** is hard for some dimensions $(h(m), m)$, then it is hard for $(\lfloor C(\varepsilon) m \rfloor, m)$. One might think that this means that **DLC** is hardest when n is about $C(\varepsilon)m$, i.e. when the dimension of the code is about equal to the channel capacity times the length of the words. However, what this means is not clear and is actually misleading.

All we have is that certain reductions where the probability of solution is never decreased "too much" by using solutions for instances where n is about $C(\varepsilon)m$. This is comparable to when $n = m$ for **SS**, in Section 1.2.2. One might think that, while **DLC**$(\lfloor C(\varepsilon)m \rfloor, m)$ is hard, the larger m is, the harder it gets, but this is not clear. For example, we do not know how to compare **DLC**$(\lfloor C(\varepsilon)m^2 \rfloor, m^2)$ and **DLC**$(\lfloor C(\varepsilon)m \rfloor, m)$. This open question will be discussed in Section 4.2.1.

1.4 Approximation results for SS and DLC

Another indication that a problem is hard is that it is hard to approximate. There is a sense in which **SS** is efficiently approximable. In another sense, a problem closely related to **DLC** is inapproximable in any efficient way, unless the hierarchy of some complexity classes collapses.

1.4.1 Subset Sum problem

Consider the functional version of **SS** as defined in [GJ79, $\mathcal{NP}$-complete problem SP13], which is as Definition 3, but with actual addition (not $\bmod 2^m$). As discussed before, the two worst case problems are equivalent. This version of Functional **SS** is approximable in the following sense [KPS97].

Theorem 23 [Approximation of a subset sum] *Let $0 < \delta < 1$. For n integers of m bits of length, let z^* be the sum of a subset of the n integers. Then there is an algorithm which returns a subset of the n integers that sums to z such that $\frac{|z^*-z|}{z^*} \leq \delta$. This algorithm runs in time $\mathcal{O}(\min[n/\delta, n + (1/\delta^2)\log(1/\delta)])$ and space $\mathcal{O}(n + 1/\delta)$.*

1.4.2 Decoding of Linear Codes problem

The maximum likelihood decoding problem, as introduced at the end of Section 1.3.1, aims to the decoding of a received word to a nearest codeword. Since its solution is a particular solution to **DLC**, it is also $\mathcal{NP}$-hard. Define the distance between two words x and y to be the Hamming weight of their difference $w(x-y)$. Then the distance between a received word and a nearest codeword is inapproximable in the following sense [ABSS97].

Theorem 24 [Hardness of approximation of the distance to a nearest codeword] *Let $\delta > 0$. For an $n \times m$ code A and a received m-bit word α, let a nearest codeword to α be x, and their distance be $d^* = w(\alpha - x)$. Then there is no polynomial time algorithm which returns solutions d such that $\frac{|d^*-d|}{d^*} \leq 2^{\log^{1-\delta} m}$, unless $\mathcal{NP} \subseteq \mathcal{QP}$, the class of deterministic quasi-polynomial time algorithms.*

A problem related to the maximum likelihood decoding problem is the minimum distance of a code problem (as defined at the end of Section 1.3.1, Definition 14). The corresponding decision problem is $\mathcal{NP}$-complete [Var97], and the functional problem is hard to approximate in the two following ways [DMS99].

Theorem 25 [Hardness of approximation of the minimum distance of a code] *Let $\delta > 0$. For an $n \times m$ code A, let the minimum distance be d^*. Then there is no random polynomial time algorithm which returns solutions d such that $\frac{|d^*-d|}{d^*} \leq \delta$, unless $\mathcal{NP} = \mathcal{RP}$, the class of random polynomial time algorithms.*

Theorem 26 [Hardness of approximation of the minimum distance of a code] *Let $\delta > 0$. For an $n \times m$ code A, let the minimum distance be d^*. Then there is no random quasi-polynomial time algorithm which returns solutions d such that $\frac{|d^*-d|}{d^*} \leq 2^{\log^{1-\delta} m}$, unless $\mathcal{NP} \subseteq \mathcal{RQP}$, the class of random quasi-polynomial time algorithms.*

1.5 Notes

The ideas developed at the beginning of Section 1.1 were taken from [LP98, Section 7.4] and [Gol98, Chapter 2]. Other complexity theory references are [vLe90, Volume A], [HU79], and [Pap94]. Another cryptography reference is [Lub96]. Parts of [Gol98] can also be found as [Gol99].

Some parts of Section 1.2 borrowed ideas from [Odl90]. The greater part of the discussion of the solved instances of **SS** in Section 1.2.2 was taken from [IN96].

A proof of Section 1.3.2's Lemma 17 can be found in [TV91, p. 77]. It can be proven similarly to the Gilbert-Varshamov Bound [PW72, Section 4.1]. An alternate discussion can be found in [vL82, Section 5.1]. More historical details about Shannon's Theorems can be found in [Abr63, p. 149, 174]. Other discussions and proofs of Shannon's Theorems (for not necessarily linear codes) can be found in [vL82, Chapter 2], [PW72, Section 4.2], and [Ham86, Chapter 10]. The Converse of Shannon's Theorem can be found in [SW64, p. 71], [Ham86, Section 10.8], and [Abr63, Chapter 6]. Other coding theory references include [MS77], [Gal68], and [Sti95, Chapter 10] .

The facts concerning approximation schemes for coding theory problems, at the end of Section 1.4, are taken from [DMS99].

Chapter 2

Reductions for the Subset-Sum Problem

2.1 Introduction

In a 1996 paper, R, Impagliazzo and M. Naor show two average-case reductions for **SS** [IN96, Proposition 1.2]. We use similar ideas to obtain stronger and additional reductions.

The necessary background for this chapter can be found in Chapter 1, Sections 1.1 and 1.2. Technical lemmata are proven in Chapter 5.

2.1.1 Preliminary conventions and lemma

Conventions for the figures

On the figures, $\mathbf{SS}(n, m)$ will always be represented with bold lines. The problem $\mathbf{SS}(n', m)$ or $\mathbf{SS}(n, m')$ will always be the problem which is to be reduced to $\mathbf{SS}(n, m)$, with dimensions n' or m' that may be different from n or m respectively. On the figures, $\mathbf{SS}(n', m)$ or $\mathbf{SS}(n, m')$ will always be represented with thin lines.

Preliminary lemma

Lemma 27 *Consider $SS(n, m)$. If $x \neq x'$, then* $\Pr_A[x \odot A = x' \odot A] = \frac{1}{2^m}$.

Proof: Since $x \neq x'$, assume without loss of generality that $x_j = 1$ and $x'_j = 0$. We will prove that $x \odot A$ is distributed uniformly, and independently of $x' \odot A$.

Fix A_i for all $i \neq j$. This determines $x' \odot A = \sum_{i=1}^{n} x'_i A_i$.

Next, choose A_j uniformly. Then $x \odot A = \sum_{i=1}^{n} x_i A_i$ is uniformly distributed. ■

2.1.2 Results

In Section 2.2.2, we will show the following. In each case, we let p be the probability that the given $\mathbf{SS}(n, m)$ oracle succeeds on a properly distributed input.

1. **Theorem 28** If $n \leq m \leq m'$, then $\mathbf{SS}(n, m') \propto_{av} \mathbf{SS}(n, m)$. The probability for $\mathbf{SS}(n, m')$ is $> \frac{p^2}{2}$. See Figure 2.2.

   This is a stronger version of [IN96, Proposition 1.2, Part 1].

2. **Theorem 30** If $n \geq m \geq m'$, then $\mathbf{SS}(n, m') \propto_{av} \mathbf{SS}(n, m)$. The probability for $\mathbf{SS}(n, m')$ is $> \frac{p^2}{4}$. Corollary 33 has limit of probability $> \frac{p^2}{2}$. See Figure 2.3.

   This is a stronger version of [IN96, Proposition 1.2, Part 2].

3. **Theorem 35** If $n' \leq n \leq m$, then $\mathbf{SS}(n', m) \propto_{av} \mathbf{SS}(n, m)$. The probability for $\mathbf{SS}(n', m)$ is $> \frac{p^2}{2}$. See Figure 2.4.

4. **Theorem 36** If $n' \geq n \geq m$, then $\mathbf{SS}(n', m) \propto_{av} \mathbf{SS}(n, m)$. The probability for $\mathbf{SS}(n', m)$ is $> \frac{p^2}{2}$. See Figure 2.5.

There are no statements, in [IN96], such as items 3 and 4, that allow the parameter n, the number of integers, to vary to n'.

2.2 Four reductions

2.2.1 Notation and definitions

Inputs to the assumed algorithm corresponding to instances of SS(n, m)

Suppose that n and m are fixed and I is an oracle for the **SS** problem.

Generating function Consider **SS**(n, m). Define the function f mapping $mn + n$ bits to $mn + m$ bits as $f(A, x) = (A, x \odot A)$, and define the equivalence relation

$$(A, x) \equiv (A', x') \iff f(A, x) = f(A', x')$$

Equivalence classes Define $S = \{(A, x) : A \in \{0,1\}^{mn}, x \in \{0,1\}^n\}$ and define the equivalence classes on S by

$$\mathcal{EQ}(A, x) = \{(A', x') : (A, x) \equiv (A', x')\}$$

Solved instances Denote by U the subset of S which are solved by I, with success. It is well-defined since I is a deterministic oracle.

$$U = \{(A, x) \in S : A \odot I(A, x \odot A) = x \odot A\}$$

Note that if $(A, x) \equiv (A', x')$, then

$$(A, x) \in U \iff (A', x') \in U$$

so that the elements of an equivalence class are either all in U or all outside U.

Generalized inputs

Let $T = \{(A, \alpha) : A \in \{0,1\}^{mn}, \alpha \in \{0,1\}^m\}$, the set of all possible inputs to I, not necessarily corresponding to instances of **SS**(n, m).

Denote by V the subset of T of instances which are solved by I, with success.

$$V = \{(A, \alpha) \in T : A \odot I(A, \alpha) = \alpha\}$$

Relations between inputs for SS(n, m) and generalized inputs

Note that $(A, x) \in U$ if and only if $(A, x \odot A) \in V$. Also, $(A, \alpha) \in V$ is an input to I for **SS**(n, m) if and only if $\alpha = x \odot A$ for some $x \in \{0,1\}^n$. Therefore, $(A, \alpha) \in V$ implies that, for some $x \in \{0,1\}^n$, the second argument $\alpha = x \odot A$.

There is a 1-1 correspondence between the equivalence classes that partition U and the elements in V. See Figure 2.1. Note that the point shown in U_i is (A^i, x^i), and that the second argument of this pair, x^i, is the solution returned by $I(A^i, \alpha^i)$. Also notice that $f(U) = V$ and $f^{-1}(V) = U$.

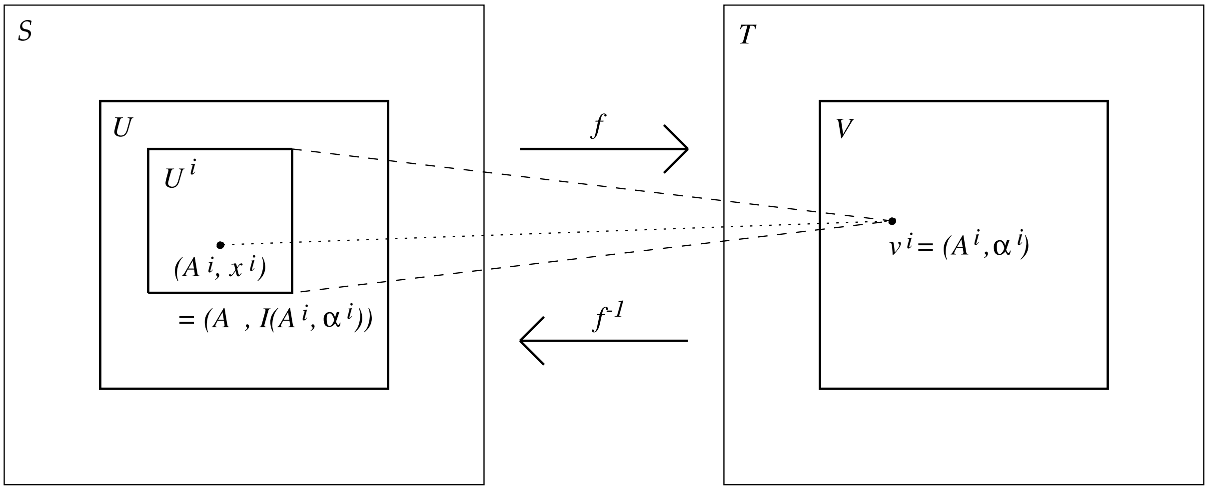

Figure 2.1: Inputs to I

Notation Let $k = |V|$ and denote the elements of V by $v^i = (A^i, \alpha^i)$ so that

$$V = \{(A^i, \alpha^i)\}_{i=1}^k$$

Let $x^i = I(A^i, \alpha^i)$ and denote the equivalence classes that partition U by $U^i = \mathcal{EQ}(A^i, x^i)$, so that

$$U = \cup_{i=1}^k U^i$$

Then $U^i = \mathcal{EQ}(A^i, x^i) \subset U$ corresponds to $v^i = (A^i, \alpha^i) \in V$ via f.

2.2.2 Reductions

Theorem 28 *If* $n \leq m \leq m'$, *then* $\mathbf{SS}(n, m') \propto_{av} \mathbf{SS}(n, m)$, *and the probability for* $\mathbf{SS}(n, m')$ *is* $> \frac{p^2}{2}$.

Proof: We prove that $\mathbf{SS}(n, m') \propto_{av} \mathbf{SS}(n, m)$ by showing that the probability for $\mathbf{SS}(n, m')$ is $> \frac{p^2}{2}$, using an oracle for $\mathbf{SS}(n, m)$. We transform an average-case instance of $\mathbf{SS}(n, m')$ to an instance of $\mathbf{SS}(n, m)$, by algorithm J, illustrated in Figure 2.2. The idea is simply to truncate the input to submit it to I. This process returns a good solution to the larger, original problem with a probability bounded below by $\frac{p^2}{2}$.

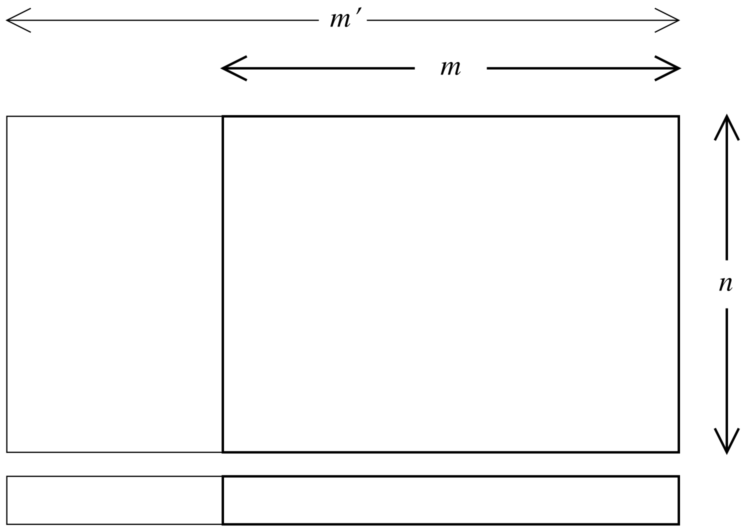

Figure 2.2: The case $n \leq m \leq m'$

Let algorithm $J(B, \beta)$, for $B \in \{0,1\}^{m'n}$ and $\beta \in \{0,1\}^{m'}$, behave in the following way.

1. Truncate the n integers B_i and β, each of m' bits, into n integers of m bits, A_i and α

   for $i = 1$ to n do
   $$A_i := B_i \bmod 2^m$$
   and
   $$\alpha := \beta \bmod 2^m$$

2. Output $y' := I(A, \alpha)$

Statistical experiment: Uniformly select an $m'n$-bit B and an n-bit y, and then compute $y' = J(B, y \odot B)$.

Event of interest: The algorithm B is successful if $y \odot B = y' \odot B$.

Claim: $\Pr[y \odot B = y' \odot B] > \frac{p^2}{2}$.

That I is successful on $(A, \alpha) = (A, y \odot A)$ does not imply that J is successful on $(B, y \odot B)$, because there may be more solutions to $\mathbf{SS}(n, m)$ than to $\mathbf{SS}(n, m')$. Nevertheless, it is sufficient for J to be successful that the returned $y' = y$. With A and α defined as computed in algorithm J, we bound the probability that J solves $\mathbf{SS}(n, m')$ by

$$\begin{aligned} \Pr[y \odot B = y' \odot B] &\geq \Pr[I(A, \alpha) = y] \\ &= \Pr[I(A, y \odot A) = y]. \end{aligned}$$

As defined in algorithm J, the argument A is uniformly distributed whenever B is, and the $\alpha = y \odot A$ has the distribution corresponding to $\mathbf{SS}(n, m)$, whenever A and y are chosen uniformly, that is, whenever B and y are chosen uniformly. Consequently, it suffices to show that for a mn-bit vector A and a n-bit vector y chosen uniformly

$$\Pr[I(A, y \odot A) = y] > \frac{p^2}{2}$$

which is given by Lemma 29, which follows. ■

Lemma 29 [Approximate one-oneness] *Let $n \leq m$. If algorithm I solves Average case $\mathbf{SS}(n, m)$ with probability p, then, for inputs $A \in_U \{0,1\}^{mn}$ and $x \in_U \{0,1\}^n$*

$$\Pr[I(A, x \odot A) = x] > \frac{p^2}{2}$$

Proof: Following the definitions and notations introduced in Section 2.2.1, there are $|V|$ inputs in S for which algorithm I returns the same x we started with, which are precisely the equivalence class representatives under the algorithm A. Since the distribution of the generators of the input to I, namely (A, x), is uniform, we have a uniform probability of selecting the class representatives. Thus, we want to show that if

$$\frac{|U|}{|S|} = p$$

then

$$\frac{|V|}{|S|} = \frac{k}{|S|} > \frac{p^2}{2}$$

We want to compute a lower bound on the unknown k, the number of equivalence classes. So consider the probability that two elements of U are in the same equivalence class, i.e. the probability distribution of I's inputs, as generated by the function f, that is

$$\Pr[f(A,x) = f(A',x')] = \Pr[(A,x) \equiv (A',x')]$$

for mn-bit A, A' and n-bit x, x' uniformly chosen.

Bound from above

$$\begin{aligned}
\Pr[(A,x) \equiv (A',x')] &= \Pr[(A, x \odot A) = (A', x' \odot A')] \\
&= \Pr[A = A'] \ (\Pr[x \neq x'] \ \Pr[x \odot A = x' \odot A \mid x \neq x'] \\
&\quad + \Pr[x = x'] \ \Pr[x \odot A = x' \odot A \mid x = x'])
\end{aligned}$$

Then the first line below is obtained via Lemma 27. The third line below is because $\frac{1}{2^m} \leq \frac{1}{2^n}$, since $n \leq m$.

$$\begin{aligned}
\Pr[(A,x) \equiv (A',x')] &= \frac{1}{2^{mn}} \left[\left(1 - \frac{1}{2^n}\right) \frac{1}{2^m} + \frac{1}{2^n} \, 1 \right] \\
&< \frac{1}{2^{mn}} \left(\frac{1}{2^m} + \frac{1}{2^n} \right) \\
&\leq \frac{2}{2^{mn+n}} = \frac{2}{|S|}
\end{aligned}$$

Bound from below On the first line below, the bound below is obtained via the added restriction that (A, x) and (A', x') are in U. To obtain the second line, note that it suffices that $(A, x) \equiv (A', x')$ for them to be in the same equivalence class because I is deterministic i.e., for equal inputs, I has equal outputs.

$$\begin{aligned}
\Pr[(A,x) \equiv (A',x')] &\geq \Pr[(A,x) \equiv (A',x') \in U] \\
&= \sum_{i=1}^{k} \left(\Pr[(A,x) \in U^i]\right)^2 = \sum_{i=1}^{k} \left(\frac{|U^i|}{|S|}\right)^2 \\
&= \left(\frac{|U|}{|S|}\right)^2 \cdot \sum_{i=1}^{k} \left(\frac{|U^i|}{|U|}\right)^2 = p^2 \cdot \sum_{i=1}^{k} \left(\frac{|U^i|}{|U|}\right)^2
\end{aligned}$$

In Lemma 54 from Chapter 5, let $n = k$, $x_i = \frac{|U^i|}{|U|}$ and $f(x) = x^2$. Then $\frac{1}{k}\sum_{i=1}^{k}\left(\frac{|U^i|}{|U|}\right)^2 \geq \left(\frac{\sum_{i=1}^{k}|U^i|}{k|U|}\right)^2$ so

$$\sum_{i=1}^{k} \left(\frac{|U^i|}{|U|}\right)^2 \geq k\left(\frac{\sum_{i=1}^{k}|U^i|}{k|U|}\right)^2 = k\left(\frac{|U|}{k|U|}\right)^2 = \frac{1}{k}$$

The use of Lemma 54 from Chapter 5 corresponds to the intuition that to minimize the probability that two elements are in the same equivalence class, all classes should be made to have the same size, $\frac{|U|}{k}$. The bound from below

$$\Pr[(A,x) \equiv (A',x')] \geq \frac{p^2}{k}$$

will be referred to and used in the proofs of Lemma 34 in this chapter, and Lemmata 39 and 44 in Chapter 3.

We have found that

$$\begin{aligned}
\frac{p^2}{k} \leq \Pr[f(A,x) &= f(A',x')] < \frac{2}{|S|} \\
\frac{k}{|S|} &> \frac{p^2}{2}
\end{aligned}$$

■

Theorem 30 *If $n \geq m \geq m'$, then* $\mathbf{SS}(n, m') \propto_{av} \mathbf{SS}(n, m)$ *and the probability for* $\mathbf{SS}(n, m')$ *is* $> \frac{p^2}{4}$.

Proof: We transform an average-case instance of $\mathbf{SS}(n, m')$ to an instance of $\mathbf{SS}(n, m)$ where the inputs are uniformly distributed, by algorithm J, illustrated in Figure 2.3. The most obvious strategy would be simply to append random bits to the input, and submit it to I. However, we have not been able to analyze it, and this question will be discussed in Section 4.1.2.

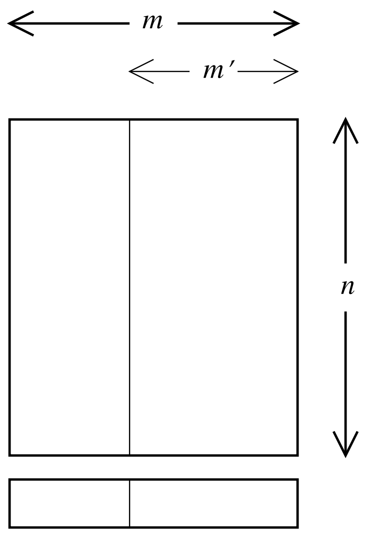

Figure 2.3: The case $n \geq m \geq m'$

Informally, algorithm J works in the following way. First, its input matrix and sum are modified. One of the integers, corresponding to one of the matrix's rows, is replaced by a new random one. According to whether we guess that this integer was used or not in the given sum, and whether we guess that it is going to be used or not by the oracle's answer, we modify the sum to be given to the oracle. Then random bits are appended to the matrix and sum, and the resulting matrix and sum are passed to the oracle I. The oracle's solution is modified, according to the guesses made.

We are given a random matrix B, and, for a random selection of its rows, in the form of an incidence vector y, we are given the sum of these rows $\beta = y \odot B$. It is convenient to assume that we are also given $i = y_1$. More precisely, algorithm J could guess whether the first integer (corresponding to the first row of matrix B) is or is not used in the given sum. In algorithm J below, $i = 0$ corresponds to the guess that the first row is not used, and vice versa for $i = 1$. Each of these possibilities happens with probability exactly $\frac{1}{2}$. Nevertheless, by repeating this procedure twice with different guesses, one of them is certain to be true. So we analyze the case when i is the correct guess.

Then algorithm J guesses whether or not the first integer (this time corresponding to the first row of matrix A, obtained from B) is going to be used in the oracle's solution. In algorithm J below, j is the guess of the first bit of the oracle's answer. We will show that oracle I can not distinguish which guesses were made. For one oracle call, the probability that the guess was right is hence exactly equal to $\frac{1}{2}$.

Let algorithm $J(B, \beta, i)$, for $B \in \{0,1\}^{m'n}$, $\beta \in \{0,1\}^{m'}$, and $i \in \{0,1\}$, behave in the following way.

1. Uniformly flip a bit $j \in_U \{0,1\}$
2. Modify and extend B to A
   (a) $B' := B$ with B_1 replaced by $B'_1 \in_U \{0,1\}^{m'}$
   (b) Extend B' to A with $m - m'$ uniformly chosen bits on the left of each A_k
      i. Choose $B'' \in_U \{0,1\}^{(m-m')n}$
      ii. $A := B'' | B'$
3. Modify and extend β to α
   (a) $\beta' := \beta + (1-i)B_1 + jB'_1$
   (b) Extend β' to α with $m - m'$ uniformly chosen bits on the left
      i. Choose $\beta'' \in_U \{0,1\}^{m-m'}$
      ii. $\alpha := \beta'' | \beta'$
4. $x := I(A, \alpha)$
5. Output $y' := x$ but with $y'_1 = i - 1$

Note that even though the statistical experiment is to select an $m'n$-bit B and an n-bit y uniformly, and then to compute $y' = J(B, y \odot B)$, the leftmost bits of α do not, in general, correspond to the sum of the rows selected by y.

Statistical experiment: Uniformly select an $m'n$-bit B and an n-bit y, and then compute $y' = J(B, y \odot B, y_1)$.

Event of interest: The algorithm J is successful if $y \odot B = y' \odot B$.

Claim: $\Pr[y \odot B = y' \odot B] > \frac{p^2}{4}$.

We first show that

$$\Pr[y \odot B = y' \odot B] \geq \Pr[x \odot A = \alpha \ \& \ x_1 = j]$$

From line (3a) of algorithm J we have $\beta' := \beta + (1-i)B_1 + jB_1'$. Following the notation of algorithm J, the following two lines are equivalent.

$$\begin{aligned} x \odot A &= \alpha \\ x \odot (B''|B') &= \beta''|(\beta + (1-i)B_1 + jB_1') \end{aligned}$$

Neglect the leftmost bits, which correspond to B'' and β''. This neglect of the leftmost bits is the reason why $\Pr[y \odot B = y' \odot B] \geq \Pr[x \odot A = \alpha \ \& \ x_1 = j]$ contains a "$\geq$", and not a "$=$". Then the following holds.

$$x \odot B' = \beta + (1-i)B_1 + jB_1'$$

Overall, it holds that

$$\begin{aligned} y' \odot B &= (i-1)B_1 + \sum_{k=2}^{n} x_k B_k = (i-1)B_1 + \sum_{k=2}^{n} x_k B_k' \\ &= (i-1)B_1 - x_1 B_1' + x \odot B' \\ &= (i-1)B_1 - x_1 B_1' + \beta + (1-i)B_1 + jB_1' = \beta + (j - x_1)B_1' \\ &= \beta \end{aligned}$$

By the multiplication rule for two events

$$\Pr[x \odot A = \alpha \ \&\ x_1 = j] \;=\; \Pr[x \odot A = \alpha]\ \Pr[x_1 = j \mid x \odot A = \alpha]$$

Independence of the distribution of (A, α) from j We assume that $i = y_1$. Then there are two cases, corresponding to the two guesses that can be made for j. We now show that in each of these cases, (A, α) is uniformly distributed. In other words, regardless of which guess is made for j, algorithm J can not extract information of the value of that j.

On line (3a) of algorithm J

$$\begin{aligned} \beta' &:= y \odot B + (1 - y_1)B_1 + jB'_1 \\ &= B_1 + [(j-1)B'_1 + y \odot B'] \end{aligned}$$

Fact 31 *If $j = 0$, then (A, α) is uniformly distributed.*

Then $\beta' = B_1 + [-B'_1 + y \odot B']$. Since B_1 is not correlated to B', it makes β' uniformly distributed, and independent from B'.

Fact 32 *If $j = 1$, then (A, α) is uniformly distributed.*

Then $\beta' = B_1 + [y \odot B']$. Since B_1 is not correlated to B', it makes β' uniformly distributed, and independent from B'.

Distribution of x_1 given that the oracle is successful We have shown that the two distributions of (A, α), for $j = 0, 1$, are the same, regardless of the specific guess j. Therefore, it is not possible for oracle I to distinguish them, even given the event that it returns a correct answer. Since j is chosen uniformly, the first bit of the oracle's correct answer x_1 is exactly j with probability $\frac{1}{2}$.

$$\Pr[x_1 = j \mid x \odot A = \alpha] = \frac{1}{2}$$

Finally, since (A, α) is uniformly distributed, $\Pr[x \odot A = \alpha] > \frac{p^2}{2}$. This is precisely given by Lemma 34, which follows this proof.

$$\begin{aligned}
\Pr[y \odot B = y' \odot B] &\geq \Pr[x \odot A = \alpha \;\&\; x_1 = j] \\
&= \Pr[x \odot A = \alpha]\ \Pr[x_1 = j \mid x \odot A = \alpha] \\
&> \frac{p^2}{2}\frac{1}{2} = \frac{p^2}{4}
\end{aligned}$$

■

Corollary 33 *If $n \geq m \geq m'$, then $\mathbf{SS}(n, m') \propto_{av} \mathbf{SS}(n, m)$ and the lower bound for the probability for $\mathbf{SS}(n, m')$ approaches $\frac{p^2}{2}$ as the number of oracle calls increases.*

Proof: We can improve the bound of the Theorem 30 by a factor close to 2.

By running algorithm J many times on the same $(B, y \odot B, y_1)$, that is, by repeating the guess for j and the oracle calls, the probability of a wrong guess every time approaches 0. Indeed, by making N independent guesses, the probability of N wrong guesses is $\frac{1}{2^N}$.

Therefore, by repeating the algorithm many times, we can make the bound below on the probability of the reduction, in the proof of Theorem 30, to be as close as we wish to $\frac{p^2}{2}$. ■

Lemma 34 **[Approximate onto-ness]** *Let $n \geq m$. If algorithm I solves Average case $\mathbf{SS}(n, m)$ with probability p, then, for inputs $A \in_U \{0,1\}^{mn}$ and $\alpha \in_U \{0,1\}^m$, and letting $x = I(A, \alpha)$*

$$\Pr[x \odot A = \alpha] > \frac{p^2}{2}$$

Proof:

We use the notation and definitions given in Section 2.2.1. We will show that if $\frac{|U|}{|S|} = p$ then $\frac{|V|}{|T|} > \frac{p^2}{2}$.

Bound from above We consider $\Pr[(A,x) \equiv (A',x')]$. Compute the bound from above as follows. The first line below is obtained via Lemma 27. The third line below is because $\frac{1}{2^m} \geq \frac{1}{2^n}$, since $n \geq m$.

$$\begin{aligned}
\Pr[(A,x) \equiv (A',x')] &= \frac{1}{2^{mn}} \left[\left(1 - \frac{1}{2^n}\right) \frac{1}{2^m} + \frac{1}{2^n}\, 1 \right] \\
&< \frac{1}{2^{mn}} \left(\frac{1}{2^m} + \frac{1}{2^n} \right) \\
&\leq \frac{2}{2^{mn+m}} = \frac{2}{|T|}
\end{aligned}$$

Bound from below We bound $\Pr[(A,x) \equiv (A',x')] \geq \Pr[(A,x) \equiv (A',x') \in U]$. Take the bound from below as computed in the proof of Lemma 29, in this chapter, that is $\Pr[(A,x) \equiv (A',x') \in U] \geq \frac{p^2}{k}$.

We have found that $\frac{p^2}{k} < \frac{2}{|T|}$ i.e. $\frac{k}{|T|} > \frac{p^2}{2}$. ■

Theorem 35 *If $n' \leq n \leq m$, then $\mathbf{SS}(n',m) \propto_{av} \mathbf{SS}(n,m)$ and the probability for $\mathbf{SS}(n',m)$ is $> \frac{p^2}{2}$.*

Proof: See Figure 2.4.

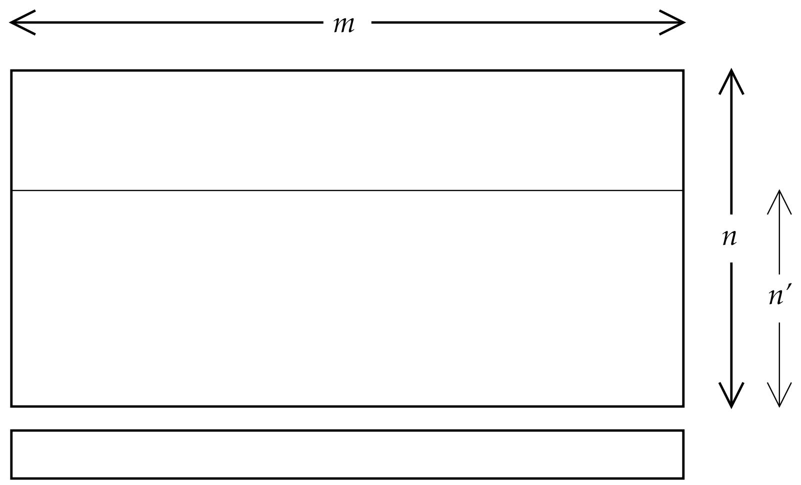

Figure 2.4: The case $n' \leq n \leq m$

Let algorithm $J(B, \beta)$, for $B \in \{0,1\}^{mn'}$ and $\beta \in \{0,1\}^m$, behave in the following way.

1. Let $A := B$ with appended rows $A_i \in_U \{0,1\}^m$ for $n'+1 \leq i \leq n$
2. Uniformly choose $z \in_U \{0,1\}^{n-n'}$ indexed by $n'+1 \leq i \leq n$
3. Let $\alpha := \beta + \sum_{i=n'+1}^{n} A_i z_i$
4. Let $x' := I(A, \alpha)$
5. Output $y' := x'$, truncated of its positions $n'+1 \leq i \leq n$

Statistical experiment: Uniformly select an mn'-bit B and an n'-bit y, and then compute $y' = B(B, y \odot B)$.

Event of interest: The algorithm J is successful if $y \odot B = y' \odot B$.

Claim: $\Pr[y \odot B = y' \odot B] > \frac{p^2}{2}$.

Let $x = y|z$. Then $\alpha = \beta + \sum_{i=n'+1}^{n} z_i A_i = y \odot B + \sum_{i=n'+1}^{n} z_i A_i = x \odot A$, so that (A, α) is a proper average-case instance of $\mathbf{SS}(n, m)$.

That I is successful on $(A, x \odot A)$ does not imply that J is successful on $(B, y \odot B)$. Nevertheless, it is sufficient for J to be successful that the returned $y' = y$. For which it is in turn sufficient that I returns $x' = x$. So we bound the probability that J solves $\mathbf{SS}(n', m)$ by

$$\Pr[y \odot B = y' \odot B] \geq \Pr[I(A, \alpha) = x] = \Pr[I(A, x \odot A) = x]$$

It suffices to show that for a mn-bit vector A and a n-bit vector x chosen uniformly, then

$$\Pr[I(A, x \odot A) = x] > \frac{p^2}{2}$$

which is given by Lemma 29. ∎

Theorem 36 *If $n' \geq n \geq m$, then $\mathbf{SS}(n', m) \propto_{av} \mathbf{SS}(n, m)$ and the probability for $\mathbf{SS}(n', m)$ is $> \frac{p^2}{2}$.*

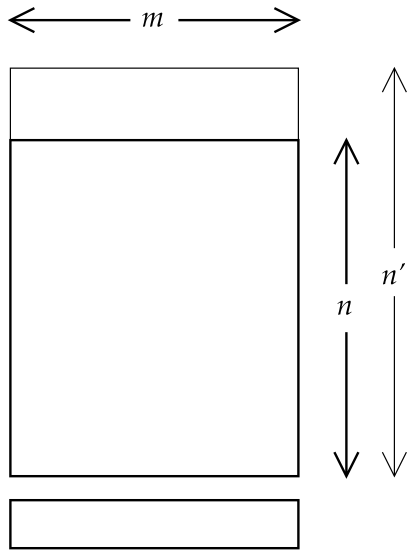

Figure 2.5: The case $n' \geq n \geq m$

Proof: See Figure 2.5.

Let algorithm $J(B, \beta)$, for $B \in \{0,1\}^{mn'}$ and $\beta \in \{0,1\}^m$, behave in the following way.

1. Let $A :=$ rows $1 \leq i \leq n$ of B
2. Let $\alpha := \beta$
3. Let $x' := I(A, \alpha)$
4. Output $y' := x'$, padded with 0's in positions $n+1 \leq i \leq n'$

Statistical experiment: Uniformly select an mn'-bit B and an n'-bit y, and then compute $y' = J(B, y \odot B)$.

Event of interest: The algorithm J is successful if $y \odot B = y' \odot B$.

Claim: $\Pr[y \odot B = y' \odot B] > \frac{p^2}{2}$.

We condition the analysis on two complementary classes of events. Let $E = [y_i = 0$ for all $n+1 \leq i \leq n']$. Consider E and its complement $\overline{E}$.

1. Subject to E, algorithm J has the same success probability as I, that is, $p > \frac{p^2}{2}$.

2. Otherwise, subject to the complementary event $\overline{E} = [y_i = 1 \text{ for some } n+1 \leq i \leq n']$, we have that $(A, y_1, ..., y_n, \sum_{i=n+1}^{n'} y_i B_i)$ is uniformly distributed.

   So $(A, y \odot A + \sum_{i=n+1}^{n'} y_i B_i)$ is uniformly distributed, and (A, α) is uniformly distributed.

   By Lemma 34, $I(A, \alpha)$ returns x' such that $x' \odot A = \alpha$ with probability $> \frac{p^2}{2}$. It suffices that $x' \odot A = \alpha$ for $y \odot B = y' \odot B$. So J is successful with probability $> \frac{p^2}{2}$.

■

2.3 Notes

The statements and proof outlines of Lemmata 29 and 34 and Theorems 28, 30, 35 and 36 in Section 2.2 are due to Rackoff [Rac98]. Lemma 27 is taken from [IN96, Proposition 1.1].

Chapter 3

Reductions for the Decoding of Linear Codes Problem

3.1 Introduction

We consider that **DLC** is a natural continuation of Chapter 2, as the reductions in the proofs of the theorems are similar to the corresponding ones for **SS**.

The necessary background for the complexity theoretic content of this chapter can be found in Section 1.1. The background for the information theoretic content of this chapter can be found in Section 1.3. Technical lemmata are proven in Chapter 5.

3.1.1 Preliminary conventions and lemma

Conventions for the figures

On the figures, $\mathbf{DLC}(n, m)$ will always be represented with bold lines. The problem $\mathbf{DLC}(n', m)$ or $\mathbf{DLC}(n, m')$ will always be the problem which is to be reduced to $\mathbf{DLC}(n, m)$, with dimensions n' or m' that may be different from n or m respectively. On the figures, $\mathbf{DLC}(n', m)$ or $\mathbf{DLC}(n, m')$ will always be represented with thin lines.

Preliminary lemma

Lemma 37 *Consider* $\mathbf{DLC}(n,m)$*. If* $x \neq x'$*, then for all* e, e'*, it holds that* $\Pr_A[xA+e = x'A+e'] = \frac{1}{2^m}$.

Proof: Since $x \neq x'$, assume without loss of generality that $x_j = 1$ and $x'_j = 0$. We will prove that $xA+e$ is distributed uniformly, and independently of $x'A+e'$.

Fix A_i for all $i \neq j$. This determines $x'A = \sum_{i=1}^n x'_i A_i$. Next, choose A_j uniformly. Then $xA = \sum_{i=1}^n x_i A_i$ is uniformly distributed.

Consequently, $\Pr_A[xA+e = x'A+e'] = \Pr_A[xA = x'A+e'-e] = \frac{1}{2^m}$. ■

3.1.2 Results

In Section 3.2.2, we will show the following. In each case, we let p be the probability that the given $\mathbf{DLC}(n,m)$ oracle succeeds on a properly distributed input.

1. **Theorem 38** If $\lceil \frac{n}{C(\varepsilon)} \rceil \leq m \leq m'$ then $\mathbf{DLC}(n,m') \propto_{av} \mathbf{DLC}(n,m)$. The probability for $\mathbf{DLC}(n,m')$ is $\Omega\left(\frac{p^2}{\sqrt{m}}\right)$. See Figure 3.2 and note that it corresponds to Figure 2.2 in Chapter 2.

2. **Theorem 40** If $\lfloor \frac{n}{C(\varepsilon)} \rfloor \geq m \geq m'$ then $\mathbf{DLC}(n,m') \propto_{av} \mathbf{DLC}(n,m)$. The probability for $\mathbf{DLC}(n,m')$ is $\Omega\left(\frac{p^2}{\sqrt{m}}\right)$. See Figure 3.3 and note that it corresponds to Figure 2.3 in Chapter 2.

3. **Theorem 45** If $n' \leq n \leq \lfloor C(\varepsilon)m \rfloor$, then $\mathbf{DLC}(n',m) \propto_{av} \mathbf{DLC}(n,m)$. The probability for $\mathbf{DLC}(n',m)$ is $> \frac{p^2}{2}$. See Figure 3.4 and note that it corresponds to Figure 2.4 in Chapter 2.

4. **Theorem 46** If $n' \geq n \geq \lceil C(\varepsilon)m \rceil$, then $\mathbf{DLC}(n',m) \propto_{av} \mathbf{DLC}(n,m)$. The probability for $\mathbf{DLC}(n',m)$ is $\Omega\left(\frac{p^2}{\sqrt{m}}\right)$. See Figure 3.5 and note that it corresponds to Figure 2.5 in Chapter 2.

3.2 Four reductions

3.2.1 Notation and definitions

Inputs corresponding to instances of DLC(n, m)

Suppose that n and m are fixed and I is an oracle for the **DLC** problem.

Generating function Consider **DLC**$(n, m,)$. Define the function f_{code} mapping $mn+n+m$ bits to $mn+m$ bits as $f_{code}(A, x, e) = (A, xA+e)$, and define the equivalence relation

$$(A, x, e) \equiv (A', x', e') \iff f_{code}(A, x, e) = f_{code}(A', x', e').$$

Equivalence classes Define $S = \{(A, x, e) : A \in \{0,1\}^{mn}, x \in \{0,1\}^n, e \in \{0,1\}^m \ \&\ w(e) \leq \lfloor \varepsilon m \rfloor\}$ and define the equivalence classes on S by

$$\mathcal{EQ}(A, x, e) = \{(A', x', e') : (A, x, e) \equiv (A', x', e')\}.$$

Solved instances Denote by U the subset of S of instances which are solved by I, with $(x', e') = I(A, xA+e)$, such that $w(e') \leq \lfloor \varepsilon m \rfloor$.

$$U = \{(A, x, e) \in S : (A, I(A, xA+e)) \equiv (A, x, e)\}.$$

Note that if $(A, x, e) \equiv (A', x', e')$, then

$$(A, x, e) \in U \iff (A', x', e') \in U$$

so that the elements of an equivalence class are either all in U or all outside U.

Generalized inputs

Let $T = \{(A, \alpha) : A \in \{0,1\}^{mn}, \alpha \in \{0,1\}^m\}$, the set of all possible inputs to I, not necessarily corresponding to instances of **DLC**(n, m). Indeed, α might not be equal to $xA+e$ for any given x and e such that $w(e) \leq \lfloor \varepsilon m \rfloor$.

Denote by V the subset of T of instances which are solved by I.

$$V = \{(A, \alpha) \in T : \text{ if } I(A, \alpha) = (x', e') \text{ then } x'A + e' = \alpha \text{ and } w(e') \leq \lfloor \varepsilon m \rfloor\}.$$

Relations between inputs for DLC(n,m) and generalized inputs

Correspondence of solved instances Note that $(A,x,e) \in U$ if and only if $(A, xA + e) \in V$. Also, $(A,\alpha) \in V$ is an input to I for **DLC**(n,m) if and only if $\alpha = xA + e$ for some $x \in \{0,1\}^n$ and some $e \in \{0,1\}^m$ such that $w(e) \leq \lfloor \varepsilon m \rfloor$. Therefore, if $(A,\alpha) \in V$, then for some $x \in \{0,1\}^n$ and some $e \in \{0,1\}^m$ such that $w(e) \leq \lfloor \varepsilon m \rfloor$, the second argument α is exactly $xA + e$.

There is a 1-1 correspondence between the equivalence classes that partition U and the elements in V. See Figure 3.1. Note that the point shown in U_i is (A^i, x^i, e^i), and that the pair formed by the second and third arguments of this triple, (x^i, e^i), is the solution returned by $I(A^i, \alpha^i)$. Also notice that $f(U) = V$ and $f^{-1}(V) = U$.

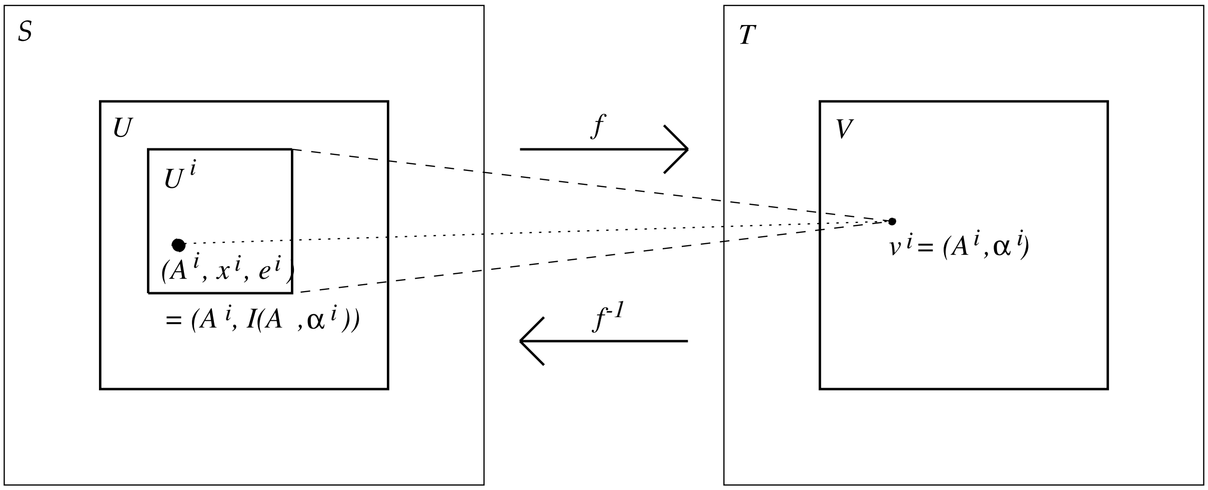

Figure 3.1: Inputs to I

Notation Let $k = |V|$ and denote the elements of V by $v^i = (A^i, \alpha^i)$ so that

$$V = \{(A^i, \alpha^i)\}_{i=1}^k$$

Let $(x^i, e^i) = I(A^i, \alpha^i)$ and denote the equivalence classes that partition U by $U^i = \mathcal{EQ}(A^i, x^i, e^i)$ so that

$$U = \cup_{i=1}^k U^i$$

Then $U^i = \mathcal{EQ}(A^i, x^i, e^i) \subset U$ corresponds to $v^i = (A^i, \alpha^i) \in V$ via f.

3.2.2 Reductions

Theorem 38 *If* $\lceil \frac{n}{C(\varepsilon)} \rceil \leq m \leq m'$ *then* $\mathbf{DLC}(n, m') \propto_{av} \mathbf{DLC}(n, m)$. *The probability for* $\mathbf{DLC}(n, m')$ *is* $\Omega\left(\frac{p^2}{\sqrt{m}}\right)$.

Proof: See Figure 3.2. Note that it corresponds to Figure 2.2 in Chapter 2.

We prove that $\mathbf{DLC}(n, m') \propto_{av} \mathbf{DLC}(n, m)$ by showing that the probability for $\mathbf{DLC}(n, m')$ is $\Omega\left(\frac{p^2}{\sqrt{m}}\right)$, using an oracle for $\mathbf{DLC}(n, m)$. We transform an average-case instance of $\mathbf{DLC}(n, m')$ to an instance of $\mathbf{DLC}(n, m)$, by algorithm J, illustrated in Figure 3.2. As in the proof of Theorem 28, the idea is simply to truncate the input to submit it to I.

Let $trunc_m(\alpha)$ denote the m rightmost bits of a m'-bit vector α. The function $trunc_m$ corresponds to mod2^m, in Chapter 2.

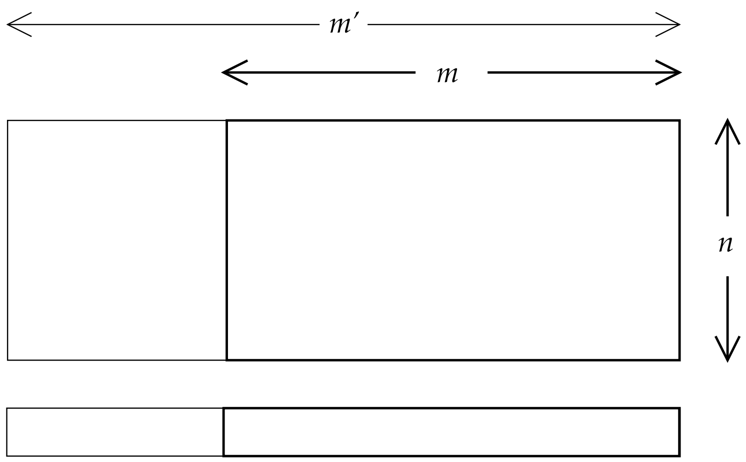

Figure 3.2: The case $n \leq \lfloor C(\varepsilon)m \rfloor \leq \lfloor C(\varepsilon)m' \rfloor$

Let algorithm $J(B, \beta)$, for $B \in \{0,1\}^{m'n}$ and $\beta \in \{0,1\}^{m'}$, behave in the following way.

1. Truncate the n integers B_i and β, each of m' bits, into n integers of m bits, A_i and α

   for $i = 1$ to n do

$$A_i := trunc_m(B_i)$$

   and

$$\alpha := trunc_m(\beta)$$

2. Let $(x, e) := I(A, \alpha)$
3. Output $(y', f') := (x, \beta - xB)$

Statistical experiment: Uniformly select an $m'n$ -bit B, an n-bit y and an m'-bit f such that $w(f) \leq \lfloor \varepsilon m' \rfloor$, and then compute $(y', f') = J(B, yB + f)$.

Event of interest: The algorithm J is successful if $yB+f = y'B+f'$ and $w(f') \leq \lfloor \varepsilon m' \rfloor$.

Claim: $\Pr[yB + f = y'B + f' \ \& \ w(f') \leq \lfloor \varepsilon m' \rfloor] \in \Omega\left(\frac{p^2}{\sqrt{m}}\right)$.

Let $f_R = trunc_m(f)$, the m rightmost bits of f. We first show that

$$\Pr[yB + f = y'B + f' \ \& \ w(f') \leq \lfloor \varepsilon m' \rfloor] \geq \Pr[(x, e) = (y, f_R)]$$

In other words, it is sufficient for J to be successful, that the oracle's answer (x, e) is the partially truncated original (y, f_R). Indeed if $(x, e) = (y, f_R)$ then, from the algorithm, $y = y'$. Then, since $y = y'$, we have

$$\begin{aligned} y'B + f' &= xB + \beta - xB = \beta \\ f' &= \beta - xB = yB + f - yB = f \end{aligned}$$

Distribution on (A, α) We need to show that, with "good" probability, the inputs to oracle I are distributed properly. Since B is uniformly distributed, so is A. However

$\alpha = yA + f_R$, and it is not necessary that f_R be uniformly distributed among the m-bit vectors with weight $\leq \lfloor \varepsilon m \rfloor$. Nevertheless, we are able to analyze this experiment by conditioning on the event E that $(f - f_R)$, that is, the leftmost bits of f, has weight exactly $\lfloor \varepsilon m' \rfloor - \lfloor \varepsilon m \rfloor$.

$$\begin{aligned} \Pr[yB + f = y'B + f' \ \& \ w(f') \leq \lfloor \varepsilon m' \rfloor] &\geq \Pr[(x,e) = (y, f_R) \ \& \ E] \\ &= \Pr[E] \ \Pr[(x,e) = (y, f_R) \mid E] \end{aligned}$$

The probability of event E is bounded by Lemma 62 from Chapter 5.

$$\Pr[E] > \frac{\sqrt{\pi}\varepsilon^{5/2}(1-2\varepsilon)}{8e^4} \frac{1}{\sqrt{m}}$$

We will show that the above experiment, conditioned on event E, is equivalent to the following new experiment.

New statistical experiment: Uniformly select an mn -bit C, an n-bit z and an m-bit g such that $w(g) \leq \lfloor \varepsilon m \rfloor$, and then compute $(z', g') = I(C, zC + g)$.

New event of interest: The algorithm I is successful if $(z, g) = (z', g')$.

New claim: $\Pr[I(C, zC + g) = (z, g)] > \frac{p^2}{2}$.

Note that this new claim will be proven as Lemma 39.

In the original statistical experiment, we are interested in the event $[(x,e) = (y, f_R) \mid E]$ where $(x, e) = I(A, yA + f_R)$, i.e the event $[I(A, yA + f_R) = (y, f_R) \mid E]$. The parameters A and y, which are independent of event E, were selected accordingly to C and z, respectively, in the new statistical experiment. It remains to show that f_R, subject to event E, is distributed as g is in the new statistical experiment.

It is clear that $w(f_R) \leq \lfloor \varepsilon m \rfloor$. Let l_i be the number of m-bit strings of weight i, for $i = 0, ..., \lfloor \varepsilon m \rfloor$. Then the number of m'-bit strings of weight $i + \lfloor \varepsilon m' \rfloor - \lfloor \varepsilon m \rfloor$ is

$$\binom{m' - m}{\lfloor \varepsilon m' \rfloor - \lfloor \varepsilon m \rfloor} l_i$$

Then, conditioning on event E

$$\Pr[w(f_R) = i] = \frac{\binom{m'-m}{\lfloor \varepsilon m' \rfloor - \lfloor \varepsilon m \rfloor} l_i}{\sum_{i=0}^{\lfloor \varepsilon m \rfloor} \binom{m'-m}{\lfloor \varepsilon m' \rfloor - \lfloor \varepsilon m \rfloor} l_i} = \frac{l_i}{\sum_{i=0}^{\lfloor \varepsilon m \rfloor} l_i}$$

In words, the probability of an error of weight i is uniform, that is, equal to the number of such vectors divided by the total number of vectors.

Overall, (A, y, f_R), in the original statistical experiment conditioned on E, is distributed as (C, z, g), in the new statistical experiment. Therefore, assuming the new claim, which we recall will be proven as Lemma 39 which follow, we have

$$\Pr[(x, e) = (y, f_R) \mid E] = \Pr[I(C, zC + g) = (z, g)] > \frac{p^2}{2}$$

Then

$$\begin{aligned} \Pr[yB + f = y'B + f' \;\&\; w(f') \leq \lfloor \varepsilon m' \rfloor] &\geq \Pr[E] \; \Pr[(x, e) = (y, f_R) \mid E] \\ &> \frac{\sqrt{\pi}\varepsilon^{5/2}(1 - 2\varepsilon)}{8e^4} \frac{1}{\sqrt{m}} \frac{p^2}{2} \end{aligned}$$

■

Lemma 39 [Approximate one-oneness] *Let $m \geq \lceil \frac{n}{C(\varepsilon)} \rceil$. If algorithm I solves Average case* **DLC**(n, m) *with probability p, then, for inputs $A \in_U \{0,1\}^{mn}$, $x \in_U \{0,1\}^n$ and $e \in_U \{0,1\}^m$ such that $w(e) \leq \lfloor \varepsilon m \rfloor$*

$$\Pr[I(A, xA + e) = (x, e)] > \frac{p^2}{2}$$

Proof:

Following the definitions and notations introduced in Section 3.2.1, there are $|V|$ inputs in S for which algorithm I returns the same (x, e) we started with, which are precisely the equivalence class representatives under the algorithm I. Since the distribution of the generators of the input to I, namely (A, x, e), is uniform, we have a uniform probability of selecting the class representatives. Thus, we want to show that if $\frac{|U|}{|S|} = p$ then $\frac{|V|}{|S|} = \frac{k}{|S|} > \frac{p^2}{2}$.

We want to compute a lower bound on the unknown k, the number of equivalence classes. So consider the probability that two elements of U are in the same equivalence class, i.e. the probability distribution of I's inputs, as generated by the function f_{code}, that is

$$\Pr[f_{code}(A, x, e) = f_{code}(A', x', e')] = \Pr[(A, x, e) \equiv (A', x', e')]$$

Bound from above

$$\begin{aligned}
\Pr[(A, x, e) \equiv (A', x', e')] &= \Pr[(A, xA + e) = (A', x'A + e')] \\
&= \Pr[A = A']\ \Pr[x = x']\ (\Pr[xA + e = xA + e'] \\
&\quad + \Pr[x \neq x']\ \Pr[xA + e = x'A + e' \mid x \neq x']) \\
&= \frac{1}{2^{mn}} \left[\frac{1}{2^n} \Pr[e = e']\ \frac{1}{2^n} \right. \\
&\quad \left. + \left(1 - \frac{1}{2^n}\right) \Pr[xA + e = x'A + e' \mid x \neq x'] \right]
\end{aligned}$$

Using Lemma 37, if $x \neq x'$, we find $Pr[xA + e = x'A + e'] = \frac{1}{2^m}$.

$$\Pr[(A, x, e) \equiv (A', x', e')] = \frac{1}{2^{mn}} \left[\frac{1}{2^n} \Pr[e = e']\ \frac{1}{2^n} + \left(1 - \frac{1}{2^n}\right) \frac{1}{2^m} \right]$$

Let E denote the set of error vectors, so that $|E| = \sum_{i=0}^{\lfloor \varepsilon m \rfloor} \binom{m}{i}$. On the first line below, use $\Pr[e = e'] = \frac{1}{|E|}$. On the third line, use $\frac{1}{2^m} \leq \frac{1}{2^{n+mH(\varepsilon)}}$, which comes from $m \geq \lceil \frac{n}{C(\varepsilon)} \rceil \geq \frac{n}{C(\varepsilon)}$ and then $\frac{1}{2^{C(\varepsilon)m}} \leq \frac{1}{2^n}$. On the fifth line, use $|S| = 2^{mn+n}|E|$, and then, by the upper bound from Lemma 60 from Chapter 5, use $|E| = \sum_{i=0}^{\lfloor \varepsilon m \rfloor} \binom{m}{i} \leq 2^{mH(\varepsilon)}$.

$$\begin{aligned}
\Pr[(A, xA + e) = (A', x'A + e')] &= \frac{1}{2^{mn}} \left[\frac{1}{2^n|E|} + \frac{1}{2^m}(1 - \frac{1}{2^n}) \right] \\
&< \frac{1}{2^{mn}} \left[\frac{1}{2^n|E|} + \frac{1}{2^m} \right] \\
&\leq \frac{1}{2^{mn}} \left[\frac{1}{2^n|E|} + \frac{1}{2^{n+mH(\varepsilon)}} \right] \\
&< \frac{1}{2^{mn+n}|E|} \left[1 + \frac{|E|}{2^{mH(\varepsilon)}} \right] \\
&= \frac{1}{|S|} \left[1 + \frac{|E|}{2^{mH(\varepsilon)}} \right] \leq \frac{2}{|S|}
\end{aligned}$$

Overall

$$\Pr[(A, xA + e) = (A', x'A + e')] < \frac{2}{|S|}$$

Bound from below We bound $\Pr[(A, x, e) \equiv (A', x', e')] \geq \Pr[(A, x, e) \equiv (A', x', e') \in U]$. Take the bound from below as computed in the proof of Lemma 29, in Chapter 2. Here, the set S here consists of triples instead of pairs, so the proof of this bound should be changed accordingly, formally speaking. Nevertheless, no result is affected by that, so it would be redundant to give another proof of the bound. We have $\Pr[(A, x, e) \equiv (A', x', e') \in U] \geq \frac{p^2}{k}$.

We have found that $\frac{p^2}{k} \leq \Pr[f(A, x, e) = f(A', x', e')] < \frac{2}{|S|}$ i.e. $\frac{k}{|S|} < \frac{p^2}{2}$ ■

Theorem 40 *If* $\lfloor \frac{n}{C(\varepsilon)} \rfloor \geq m \geq m'$ *then* $\mathbf{DLC}(n, m') \propto_{av} \mathbf{DLC}(n, m)$. *The probability for* $\mathbf{DLC}(n, m')$ *is* $\Omega\left(\frac{p^2}{\sqrt{m}}\right)$.

Proof: See Figure 3.3. Note that it corresponds to Figure 2.3 in Chapter 2.

We transform an average-case instance of $\mathbf{DLC}(n, m')$ to an instance of $\mathbf{DLC}(n, m)$ where the inputs are uniformly distributed, by algorithm J, illustrated in Figure 3.3. As in the proof of Theorem 30, the most obvious strategy would be simply to append random bits to the input, and submit it to I. However, we have not been able to analyze it, and this question will be discussed in Section 4.2.2.

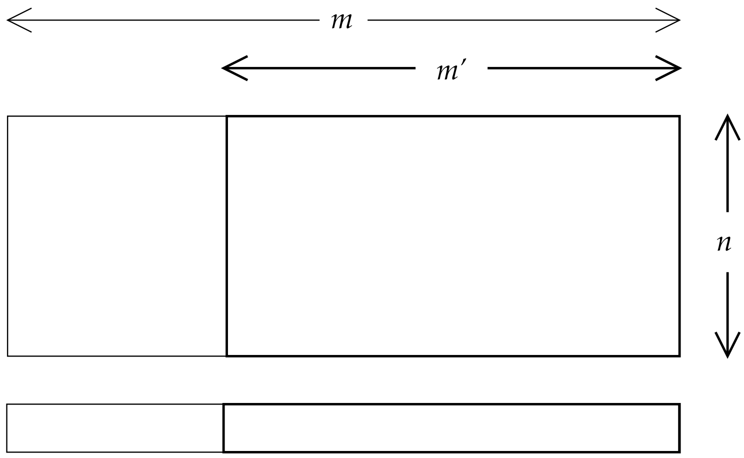

Figure 3.3: The case $\lfloor \frac{n}{C(\varepsilon)} \rfloor \geq m \geq m'$

Informally, algorithm J works in the following way. First, its input matrix and word are modified. One of matrix's rows is replaced by a new random one. According to whether we guess that this row was used or not in the original codeword, and whether we guess that it is going to be used or not by the oracle, we modify the word to be given to the oracle. Then random bits are appended to the matrix and word, and the resulting matrix and word are passed to the oracle I. The codeword from the oracle's solution is modified, according to the guesses made.

We are given a random matrix B. Also, for a random selection of B's rows, in the form of an incidence vector y, and for an random error vector f of weight bounded above by $\varepsilon m'$, we are given the word $\beta = yB + f$. It is convenient to assume that we are also given $i = y_1$. More precisely, algorithm J could guess whether the first row of matrix B is or is not used in the given sum. In algorithm J below, $i = 0$ corresponds to the guess that the first row is not used, and vice versa for $i = 1$. Each of these possibilities happens with probability exactly $\frac{1}{2}$. Nevertheless, by repeating this procedure twice with different guesses, one of them is certain to be true. So we analyze the case when i is the correct guess.

Then algorithm J guesses whether or not the first row of matrix A (obtained from

B) is going to be used in the oracle's solution. In algorithm J below, j is the guess of the first bit of the oracle's answer. We will show that oracle I can not distinguish which guesses were made. For one oracle call, the probability that the guess was right is hence exactly equal to $\frac{1}{2}$.

Note that algorithm J does not append random bits to the original input, as in the reduction from the proof of Theorem 30, in Chapter 2. Instead, algorithm J "sticks in" random bits in randomly chosen columns. These columns are denoted by C, and are the same in the new input matrix and word, formed by this process.

Informally, if the random bits were always appended in the same positions, it would be possible for the oracle to concentrate the errors in positions that would minimize algorithm J's success probability. Oracle I could let its returned error vector have as many ones as possible in the bit positions of the word that corresponds to our original one. For randomly "stuck in" bits, the oracle distributes the errors it returns uniformly, at the very worst.

The operations of normal concatenation $|$ and truncation (modulo), as used in the proof of Theorem 30, in Chapter 2, need to be modified for these purposes. We extend these usual operations in order to make them dependent on the concerned columns C. We define the extended operations with respect to any C, in a way that a random choice of C yields the intended randomized form of these extended operations.

First, we define what is meant by the operation of "sticking in" columns. Let B'', B', and A be vectors or matrices. If B'' has $m - m'$ columns, and B' has m' columns, let C be a vector of length m', where each distinct entry is in $\{1, ..., m\}$. It is convenient that the entries of C be in increasing order. Let $|_C$ denote the action of sticking in bits, so that $A = B''|_C B'$ means to form A as follows. If $C = \{c_1, c_2, ..., c_{m'}\}$ then for each $1 \leq j \leq m'$, the ${c_j}^{th}$ column of A is the j^{th} column of B'. The remaining columns of A are the columns of B'', in order.

Second, we define what is meant by the operation of truncating columns. Consider

C, B', and A as in the previous paragraph. Let $trunc_C$ denote the action of removing columns, so that $B' = trunc_C(A)$ means to form B' as follows. If k is an entry in C, let B''s k^{th} column be A's k^{th} column. The process is continued until all entries of C are used.

Let algorithm $J(B, \beta, i)$, for $B \in \{0,1\}^{m'n}$, $\beta \in \{0,1\}^{m'}$, and $i \in \{0,1\}$, behave in the following way.

1. Uniformly flip a bit $j \in_U \{0,1\}$
   Uniformly choose a $1 \times m'$ vector C of distinct elements in $\{1, ..., m\}$
2. Modify and extend B to $A = B''|_C B'$
   (a) $B' := B$ but with B_1 replaced by $B'_1 \in_U \{0,1\}^{m'}$
   (b) Extend B' to A with B'' consisting of $m - m'$ uniformly chosen bits, for each row
      i. Choose $B'' \in_U \{0,1\}^{(m-m')n}$
      ii. $A := B''|_C B'$
3. Modify and extend β to $\alpha = \beta''|_C \beta'$
   (a) $\beta' := \beta + (1-i)B_1 + jB'_1$
   (b) Extend β' to α with β'' consisting of $m - m'$ uniformly chosen bits
      i. Choose $\beta'' \in_U \{0,1\}^{m-m'}$
      ii. $\alpha := \beta''|\beta'$
4. Apply oracle I and transform its output
   (a) $(x, e) := I(A, \alpha)$
   (b) $y' := x$ but with $y'_1 = i - 1$
   (c) $f' := trunc_C(e)$
5. Output (y', f')

Statistical experiment: Uniformly select an $m'n$-bit B, an n-bit y and an m'-bit f such that $w(f) \leq \lfloor \varepsilon m' \rfloor$, and then compute $(y', f') = J(B, yB + f, y_1)$.

Event of interest: The algorithm J is successful if $yB + f = y'B + f'$ such that $w(f') \leq \lfloor \varepsilon m' \rfloor$.

Claim: $\Pr[yB + f = y'B + f' \ \&\ w(f') \le \lfloor \varepsilon m' \rfloor] \in \Omega\left(\frac{p^2}{\sqrt{m}}\right)$.

We first show that

$$\Pr[yB + f = y'B + f' \ \&\ w(f') \le \lfloor \varepsilon m' \rfloor] \quad \ge \quad \Pr[xA + e = \alpha \ \&\ x_1 = j \ \&\ w(f') \le \lfloor \varepsilon m' \rfloor]$$

For similar reasons as in the proof of Theorem 30 in Chapter 2, it holds that $xA+e = \alpha$ and $x_1 = j$ imply that $yB + f = y'B + f'$. For completeness, we now prove this. From line (3a) of algorithm J we have $\beta' := \beta + (1-i)B_1 + jB'_1$. Following the notation of algorithm J, the following two lines are equivalent.

$$\begin{aligned} xA + e &= \alpha \\ x(B''|_C B') + e &= \beta''|_C(\beta + (1-i)B_1 + jB'_1) \end{aligned}$$

We neglect the bits *not* indicated by C, which correspond to B'' and β''.

$$xB' + trunc_C(e) \quad = \quad xB' + f' = \beta + (1-i)B_1 + jB'_1$$

Overall, it holds that

$$\begin{aligned} y'B + f' &= (i-1)B_1 + \sum_{k=2}^{n} x_k B_k + f' \\ &= (i-1)B_1 - x_1 B'_1 + xB' + f' \\ &= (i-1)B_1 - x_1 B'_1 + \beta + (1-i)B_1 + jB'_1 = \beta + (j - x_1)B'_1 \\ &= \beta \end{aligned}$$

So we have the following, using the multiplication rule for two events, on the fourth line below.

$$\begin{aligned} & \Pr[yB + f = y'B + f' \ \&\ w(f') \le \lfloor \varepsilon m' \rfloor] \\ \ge\ & \Pr[xA + e = \alpha \ \&\ x_1 = j \ \&\ w(f') \le \lfloor \varepsilon m' \rfloor] \\ \ge\ & \Pr[xA + e = \alpha \ \&\ w(e) \le \lfloor \varepsilon m \rfloor \ \&\ x_1 = j \ \&\ w(f') \le \lfloor \varepsilon m' \rfloor] \\ =\ & \Pr[xA + e = \alpha \ \&\ w(e) \le \lfloor \varepsilon m \rfloor] \cdot \\ & \Pr[x_1 = j \ \&\ w(f') \le \lfloor \varepsilon m' \rfloor \mid xA + e = \alpha \ \&\ w(e) \le \lfloor \varepsilon m \rfloor] \end{aligned}$$

Independence of the distribution of (A, α) from j and C We assume that $i = y_1$. Then there are two cases, corresponding to the two guesses that can be made for j. We now show that in each of these cases, (B', β') is uniformly distributed. In other words, regardless of which guess is made for j, algorithm J can not extract information of the value of that j.

On line (3a) of algorithm J

$$\begin{aligned} \beta' &:= yB + f + (1 - y_1)B_1 + jB'_1 \\ &= B_1 + \sum_{k=2}^{n} y_k B_k + jB'_1 + f \end{aligned}$$

Fact 41 *If $j = 0$, then (B', β') is uniformly distributed.*

Then $\beta' = B_1 + [y \odot B' - B'_1 + f]$. Since B_1 is not correlated to B', it makes β' uniformly distributed, and independent from B'.

Fact 42 *If $j = 1$, then (B', β') is uniformly distributed.*

Then $\beta' = B_1 + [y \odot B' + f]$. Since B_1 is not correlated to B', it makes β' uniformly distributed, and independent from B'.

Furthermore, since C and (B'', β'') are chosen uniformly, the fact that (B', β') is uniformly distributed implies that $(A, \alpha) = (B''|_C B', \beta''|_C \beta')$ is also uniformly distributed. Because (A, α) is uniformly distributed, $\Pr[x \odot A = \alpha \;\&\; w(e) \leq \lfloor \varepsilon m \rfloor] \in \Omega\left(\frac{p^2}{\sqrt{m}}\right)$. This is precisely given by Lemma 44, which follows this proof.

Distribution of x_1 given that the oracle is successful We have shown that the two distributions of (A, α), for $j = 0, 1$, are the same, regardless of the specific guesses of j and C. It is not possible for oracle I to distinguish them. Therefore, oracle I has no information about the values of j and C, even given the event that it returns a correct answer. Consequently, the probabilities are the same as they would be if we chose j and C uniformly and independently, *after* the oracle call.

Since j is chosen uniformly, the first bit of the oracle's correct answer x_1 is exactly j with probability $\frac{1}{2}$. We have $\Pr[x_1 = j \mid xA + e = \alpha \;\&\; w(e) \leq \lfloor \varepsilon m \rfloor] = \frac{1}{2}$.

We now consider the event $w(f') \leq \lfloor \varepsilon m' \rfloor$ conditioned on the event that $[xA + e = \alpha \;\&\; w(e) \leq \lfloor \varepsilon m \rfloor \;\&\; x_1 = j]$. Fix e such that $w(e) = \lfloor \varepsilon m \rfloor$, since, as an adversary, I can not do anything worse than to return an e with a maximum weight. We consider the following experiment. Choose m' bits from e without replacement. We want a lower bound on the probability that no more than $\lfloor \varepsilon m' \rfloor$ of these bits are one's. As we have argued above, the experiment is equivalent to choosing C uniformly after the oracle call.

By Lemma 63 (Chapter 5), this probability is bounded below as follows.

$$\Pr[w(f') \leq \lfloor \varepsilon m' \rfloor \mid xA + e = \alpha \;\&\; w(e) = \lfloor \varepsilon m \rfloor \;\&\; x_1 = j] \quad \in \quad \Omega(1)$$

Therefore

$$\begin{aligned}
& \Pr[x_1 = j \;\&\; w(f') \leq \lfloor \varepsilon m' \rfloor \mid xA + e = \alpha \;\&\; w(e) \leq \lfloor \varepsilon m \rfloor] \\
\geq \;& \Pr[x_1 = j \mid xA + e = \alpha \;\&\; w(e) \leq \lfloor \varepsilon m \rfloor] \cdot \\
& \Pr[w(f') \leq \lfloor \varepsilon m' \rfloor \mid xA + e = \alpha \;\&\; w(e) = \lfloor \varepsilon m \rfloor \;\&\; x_1 = j] \in \frac{1}{2}\,\Omega(1) = \Omega(1)
\end{aligned}$$

Overall

$$\begin{aligned}
\Pr[yB + f = y'B + f' \;\&\; w(f') \leq \lfloor \varepsilon m' \rfloor] \quad &\in \quad \Omega\left(\frac{p^2}{\sqrt{m}}\right)\;\Omega(1) \\
&= \quad \Omega\left(\frac{p^2}{\sqrt{m}}\right)
\end{aligned}$$

■

Remark 43 *As for Theorem 30 in Chapter 2, it is possible to improve the above value of p' by a factor close to 2, by running the algorithm many times.*

Lemma 44 [Approximate onto-ness] *Let $\lfloor \frac{n}{C(\varepsilon)} \rfloor \geq m$. If algorithm I solves Average case* **DLC**(n, m) *with probability p, then, for inputs $A \in_U \{0,1\}^{mn}$ and $\alpha \in_U \{0,1\}^m$, and letting $(x, e) = I(A, \alpha)$*

$$\Pr[xA + e = \alpha \;\&\; w(e) \leq \lfloor \varepsilon m \rfloor] \in \Omega\left(\frac{p^2}{\sqrt{m}}\right)$$

Proof:

We use the notation and definitions given in Section 3.2.1. We will show that if $\frac{|U|}{|S|} = p$, then $\frac{|V|}{|T|} \in \Omega\left(\frac{p^2}{\sqrt{m}}\right)$. Consider $\Pr[(A,x,e) \equiv (A',x',e')]$.

Bound from above As in the proof of Lemma 39, let E denote the set of error vectors, so that $|E| = \sum_{i=0}^{\lfloor \varepsilon m \rfloor} \binom{m}{i}$. Since $m \le \lfloor \frac{n}{C(\varepsilon)} \rfloor \le \frac{n}{C(\varepsilon)}$, then $\frac{1}{2^n} \le \frac{1}{2^{mC(\varepsilon)}} = \frac{2^{mH(\varepsilon)}}{2^m}$. Also, $|T| = 2^{mn+m}$ and by the lower bound from Lemma 60 from Chapter 5, use $|E| = \sum_{i=0}^{\lfloor \varepsilon m \rfloor} \binom{m}{i} > \frac{\varepsilon}{e^2\sqrt{8m}} 2^{mH(\varepsilon)}$.

$$\begin{aligned}
\Pr[(A,x,e) \equiv (A',x',e')] &= \frac{1}{2^{mn}}\left[\frac{1}{2^n|E|} + \left(1 - \frac{1}{2^n}\right)\frac{1}{2^m}\right] \\
&< \frac{1}{2^{mn}}\left[\frac{1}{2^n|E|} + \frac{1}{2^m}\right] < \frac{1}{2^{mn+m}}\left[\frac{2^{mH(\varepsilon)}}{|E|} + 1\right] \\
&= \frac{1}{T}\left[\frac{\sqrt{8}e^2}{\varepsilon}\sqrt{m} + 1\right] < \frac{1}{T}\left[\frac{8e^2}{\varepsilon}\sqrt{m}\right]
\end{aligned}$$

Bound from below We bound $\Pr[(A,x,e) \equiv (A',x',e')] \ge \Pr[(A,x,e) \equiv (A',x',e') \in U]$. Take the bound from below as computed in the proof of Lemma 29, in Chapter 2. Here, as in the proof of Lemma 39, earlier in this chapter, the set S here consists of triples instead of pairs, so the proof of this bound should be changed accordingly, formally speaking. Nevertheless, no result is affected by that, so it would be redundant to give another proof of the bound. We have $\Pr[(A,x,e) \equiv (A',x',e') \in U] \ge \frac{p^2}{k}$.

We have found that $\frac{p^2}{k} < \frac{1}{T}\left[\frac{8e^2}{\varepsilon}\sqrt{m}\right]$ i.e. $\frac{k}{|T|} > \frac{\varepsilon p^2}{8e^2\sqrt{m}}$. ■

Theorem 45 *If $n' \le n \le \lfloor C(\varepsilon)m \rfloor$, then* $\mathbf{DLC}(n',m) \propto_{av} \mathbf{DLC}(n,m)$. *The probability for* $\mathbf{DLC}(n',m)$ *is* $> \frac{p^2}{2}$.

Proof: See Figure 3.4. Note that it corresponds to Figure 2.4 in Chapter 2. This proof also is similar to the one of Theorem 35.

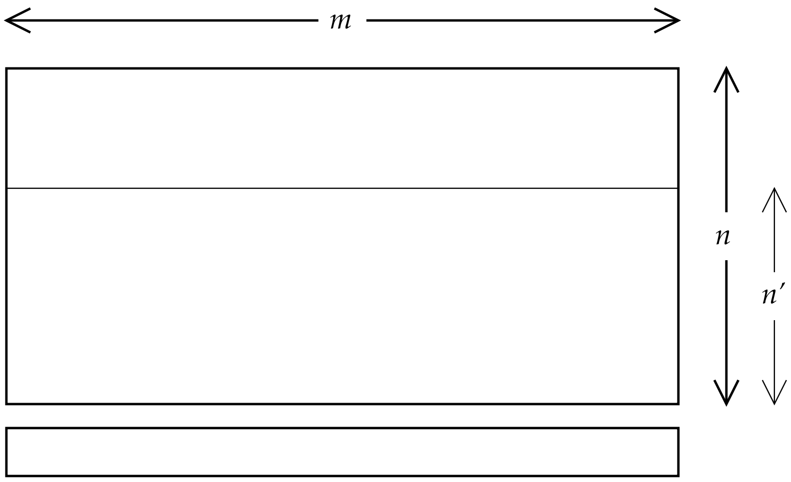

Figure 3.4: The case $n' \le n \le \lfloor C(\varepsilon)m \rfloor$

Let algorithm $J(B,\beta)$, for $B \in \{0,1\}^{mn'}$ and $\beta \in \{0,1\}^m$, behave in the following way.

1. $A = B$ with appended rows $A_i \in_U \{0,1\}^m$ for $n'+1 \le i \le n$
2. Choose $z \in_U \{0,1\}^{n-n'}$ indexed by $n'+1 \le i \le n$
3. $\alpha := \beta + \sum_{i=n'+1}^{n} a_i z_i$
4. Apply algorithm I and transform its output
   (a) $(x', e') := I(A, \alpha)$
   (b) $y' := x'$, truncated of its positions $n'+1 \le i \le n$
   (c) $f' := e'$
5. Output (y', f')

Statistical experiment: Uniformly select an mn'-bit B, an n'-bit y and an m-bit f such that $w(f) \le \lfloor \varepsilon m \rfloor$, and then compute $(x', f') = J(B, yB + f)$.

Event of interest: The algorithm J is successful if $xB+f = x'B+f'$ and $w(f') \le \lfloor \varepsilon m \rfloor$.

Claim: $\Pr[xB + f = x'B + f' \,\&\, w(f') \le \lfloor \varepsilon m \rfloor] > \frac{p^2}{2}$.

Let $x = y|z$. Then $\alpha = \beta + \sum_{i=n'+1}^{n} a_i z_i = yB + \sum_{i=n'+1}^{n} a_i z_i + f = xA + f$, so that

(A, α) is a properly distributed instance of Average case $\mathbf{DLC}(n, m)$.

That I is successful on $(A, xA+f)$ does not imply that J is successful on $(B, yB+f)$. Nevertheless, it is sufficient for J to be successful that the returned $(y', f') = (y, f)$. For which it is in turn sufficient that I returns $(x', f') = (x, f)$. So we bound the probability that J solves $\mathbf{DLC}(n', m)$ by

$$\begin{aligned} \Pr[yB + f = y'B + f' \;\&\; w(f') \leq \lfloor \varepsilon m \rfloor] &\geq \Pr[I(A, \alpha) = (x, f)] \\ &= \Pr[I(A, xA + f) = (x, f)] \end{aligned}$$

It suffices to show that for a mn-bit A, an n-bit y and an m-bit f such that $w(f) \leq \lfloor \varepsilon m \rfloor$ chosen uniformly

$$\Pr[I(A, yA + f) = (y, f)] > \frac{p^2}{2}$$

which is given by Lemma 39. ■

Theorem 46 *If $n' \geq n \geq \lceil C(\varepsilon) m \rceil$, then $\mathbf{DLC}(n', m) \propto_{av} \mathbf{DLC}(n, m)$. The probability for $\mathbf{DLC}(n', m)$ is $\Omega\left(\frac{p^2}{\sqrt{m}}\right)$.*

Proof: See Figure 3.5. Note that it corresponds to Figure 2.5 in Chapter 2. This proof also is similar to the one of Theorem 36.

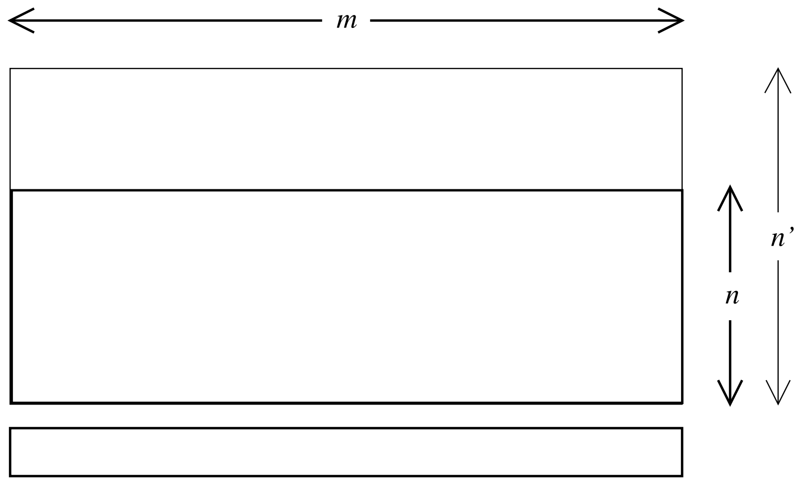

Figure 3.5: The case $n' \geq n \geq \lceil C(\varepsilon) m \rceil$

Let algorithm $J(B, \beta)$, for $B \in \{0,1\}^{mn'}$ and $\beta \in \{0,1\}^m$, behave in the following way.

1. $A :=$ rows $1 \leq i \leq n$ of B
2. $\alpha := \beta$
3. Apply algorithm I and transform its output
   (a) $(x', e') := I(A, \alpha)$
   (b) $y' := x'$, padded with 0's in positions $n+1 \leq i \leq n'$
   (c) $f' := e'$
4. Output (y', f')

Statistical experiment: Uniformly select an mn'-bit B, an n'-bit y and an m-bit f such that $w(f) \leq \lfloor \varepsilon m \rfloor$, and then compute $(y', f') = J(B, yB + f)$.

Event of interest: The algorithm J is successful if $yB+f = y'B+f'$ and $w(f') \leq \lfloor \varepsilon m \rfloor$.

Claim: $\Pr[yB + f = y'B + f' \ \&\ w(f') \leq \lfloor \varepsilon m \rfloor] \in \Omega\left(\frac{p^2}{\sqrt{m}}\right)$.

Condition the analysis on two classes of events; let $E = [y_i = 0$ for all $n+1 \leq i \leq n']$.

1. Subject to E, algorithm J has the same success probability as I, that is, $p \in \Omega\left(\frac{p^2}{\sqrt{m}}\right)$.

2. Otherwise, subject to $\overline{E} = [y_i = 1$ for some $n+1 \leq i \leq n']$, we have

$$(A, y_1, ..., y_n, \sum_{i=n+1}^{n'} y_i B_i, f)$$

   uniformly distributed. So $(A, yA + \sum_{i=n+1}^{n'} y_i B_i + f)$ is uniformly distributed, and (A, α) is uniformly distributed. By Lemma 44, $I(A, \alpha)$ returns (x', f') such that $x'A + f' = \alpha$ with probability $\Omega\left(\frac{p^2}{\sqrt{m}}\right)$. It suffices that $x'A + f' = \alpha$ for $yB + f = y'B + f'$. So J is successful with probability $\Omega\left(\frac{p^2}{\sqrt{m}}\right)$.

■

3.3 Notes

Theorems 38, 40, 45 and 46 in Section 3.2 were conjectured by Rackoff. Lemma 37 is based on Lemma 27 from Chapter 2.

Chapter 4

Conclusion and open problems

4.1 Subset Sum

4.1.1 Conclusion

For each of the four theorems for Average case **SS**, we have found a reduction with, for some constant c, probability of success $> \frac{p^2}{c}$. These theorems imply two hardness theorems.

In all theorems, we assume that the parameterization functions, g, g', h, and h' can be computed easily.

Theorem 47 *For any $g'(n)$, hardness w.r.t. to the parameterization $m' = g'(n)$ implies hardness w.r.t. to the parameterization $m = n$.*

Theorem 28 and Corollary 33 together give a reduction, for n, m' positive integers, such that $\mathbf{SS}(n, m') \propto_{av} \mathbf{SS}(n, n)$ with $p' > \frac{p^2}{2}$.

Theorem 47 is stronger than the one implied by [IN96, Proposition 1.2]. This proposition, states the following. Suppose that for all constants $c_1 > 1$ and $c_2 < 1$, it is the case that for all $n \in \mathbb{N}^+$ sufficiently large $g'(n) \geq g(n) \geq c_1 n$, or for all $n \in \mathbb{N}^+$ sufficiently large $g'(n) \leq g(n) \leq c_2 n$. Then, hardness w.r.t. to the parameterization $m' = g'(n)$

implies hardness w.r.t. to the parameterization $m = g(n)$.

Theorem 48 *For any $h'(m)$, hardness w.r.t. to the parameterization $n' = h'(m)$ implies hardness w.r.t. to the parameterization $n = m$.*

Theorems 35 and 36 together give a reduction, for n', m positive integers, such that $\mathbf{SS}(n', m) \propto_{av} \mathbf{SS}(m, m)$ with $p' > \frac{p^2}{2}$.

No such theorem is implied by the results of [IN96].

Relationship between Theorems 47 and 48 As mentioned in Chapter 1, Theorems 47 and 48 appear to be related. We address the question as to whether or not they are equivalent to one another, that is whether or not hardness with respect to any g' or h', implies hardness with respect to some appropriate h' or g', respectively. However this is not the case, which we will show through a counter-example.

We show that Theorem 48 does not imply Theorem 47. Suppose that it is possible to break the hardness assumption with respect to the parameterization $m = n$ infinitely often. Then it is possible to break the hardness assumption with respect to the parameterization $n = m$ infinitely often. By Theorem 48, since it is possible to break the hardness assumption with respect to the parameterization $n = m$ infinitely often, then it is possible to break the hardness assumption with respect to the parameterization $h'(m) = \lfloor \sqrt[3]{m} \rfloor$ infinitely often.

Then we want to show that there is no function $g'(n)$ such that the hardness assumption with respect to $g'(n)$ would be broken infinitely often. The only function for which this could hold is the inverse of h', since these are the only parameters for which we have some knowledge. Hence, we take $g'(n) = n^3$.

We want to show that it is not the case that this implies that we can break the hardness assumption with respect to the parameterization $g'(n) = n^3$ infinitely often. This would be true if there were infinitely many m such that $\lfloor \sqrt[3]{m} \rfloor = n$, for some n, *and* such that the hardness assumption fails on these particular m. However, we only assumed that the hardness assumption fails on infinitely many m, and that set could possibly exclude all perfect cubes.

We make the following observation. Suppose that for any g' or h', it would be possible to find a h' or g', respectively, which is an exact inverse of the other, that is without floors or ceilings. Then the two hardness theorems would be equivalent.

Thus, the two hardness theorems state similar results, that complement each other.

4.1.2 Improvements

We conjecture that the proof of Theorem 30 can be improved in the following way. Without using repeated oracle calls, we were only able to get a probability bounded below by $\frac{p^2}{8}$. With two oracle calls, we were only able to get a probability bounded below by $\frac{p^2}{4}$. It is only with an unbounded number of oracle calls that we were able to bound below the probability by $\frac{p^2}{2}$.

We conjecture that the bound below of $\frac{p^2}{2}$ is attainable without repeated oracle calls. To obtain this, we also conjecture that proof to consist in the analysis of the suggested simpler algorithm mentioned at the beginning of the proof of Theorem 30. It is the algorithm by which the instance to be passed to the oracle is the same as the original one, with appended random bits. It is a more "natural" reduction algorithm. This is how the proof of Proposition 1.2, Part 2 is done in [IN96]. This proposition is weaker than Theorem 30 in the sense that the later applies to more restricted values of (n, m), though it needs only one call to the oracle for the reduction algorithm's success probability to be bounded below by $\frac{p^2}{2}$.

Open problems

We do not know if the following are true.

$$SS(n,n) \quad \propto_{av} \quad SS(2n,2n)$$
$$SS(n,n) \quad \propto_{av} \quad SS(n^2,n^2)$$

The reductions we have are of a very special nature, and we did not find a way to analyze simultaneous variations of the two parameters. Nevertheless, Kabanets has proven the following [Rac98].

Theorem 49 *For $\frac{n}{2} < m < n$, for some c constant, and for all $\varepsilon > 0$*

$$SS(m, m^{1+\varepsilon}) \propto_{av} SS(cn \log n, cn \log n)$$

The proof involves obtaining a lattice with a n^d-unique shortest vector, using results from [LO85, Fri86], and finding this vector, using results from [Ajt96].

4.2 Decoding of Linear Codes

4.2.1 Conclusion

For each of the four theorems for Average case **DLC**, we have found a reduction with probability of success $> poly\left(\frac{p}{m}\right)$. These theorems imply two hardness theorems.

In all theorems, we assume that the parameterization functions, g, g', h, and h' can be computed easily.

Theorem 50 *For any $g'(n)$, hardness w.r.t. to the parameterization $m' = g'(n)$ implies hardness w.r.t. to the parameterization $m = \lfloor \frac{n}{C(\varepsilon)} \rfloor$.*

Theorems 38 and Theorem 40 together give a reduction, for n, m' positive integers, such that $\mathbf{DLC}(n, m') \propto_{av} \mathbf{DLC}(n, \lfloor \frac{n}{C(\varepsilon)} \rfloor)$ with, for some c constant, $p' > \frac{p^2}{c\sqrt{n}}$.

Theorem 51 *For any $h'(m)$, hardness w.r.t. to the parameterization $n' = h'(m)$ implies hardness w.r.t. to the parameterization $n = \lfloor C(\varepsilon)m \rfloor$.*

Theorems 45 and 46 together give a reduction, for n', m positive integers, such that $\mathbf{DLC}(n', m) \propto_{av} \mathbf{DLC}(\lfloor C(\varepsilon)m \rfloor, m)$ with, for some c constant, $p' > \frac{p^2}{c\sqrt{m}}$.

The relationship between Theorems 50 and 51 is the same as the one between Theorems 47 and 48.

4.2.2 Improvements

Statement of the theorems In the proofs of Lemmata 39 and 44, we use Lemma 60 from Chapter 5, while Lemma 61 gives tighter bounds. Changing the rates in the Theorems' statements to, for a constant c, n being about $(C(\varepsilon) + (c/m)\log m)m$ would leave Lemma 39's bound unchanged. On the other hand, this change would better Lemma 44's bound by a factor of $\sqrt{m}$, and this would better Theorems 40 and 46 by a factor of $\sqrt{m}$ as well.

However, we chose to leave the theorems statements in their simpler forms. Note that n being about $(C(\varepsilon)+(c/m)\log m)m$ is letting n be larger than in the hardness theorems in the previous section. The expression $(c/m)\log m)m$ approaches 0 as m increases. For increasing m, that n is about $(C(\varepsilon) + (c/m)\log m)m$ tends to be the same as n being about $C(\varepsilon)m$.

Improvements for the reductions As in Section 4.1.2, we conjecture that the probability of Theorem 40 can be improved by a factor 2 without repeated oracle calls. Analyzing a more "natural" reduction, as discussed in Section 4.1.2, should give this result.

Rounding The statements of the theorems of this chapter are similar to the ones in the previous chapter. They however contain an additional, fixed parameter ε, the probability of error on a bit, in turn affecting the the dimensions of the problems.

First, the dimensions of the problems are affected by the capacity $C(\varepsilon)$ associated with ε. On one hand, a theorem's assumption may be of the form $n \leq C(\varepsilon)m$, although we do not wish to require that $C(\varepsilon)m$ be an integer [1]. Since $n \in \mathbb{N}$, the inequality $n \leq C(\varepsilon)m$ is the same as $n \leq \lfloor C(\varepsilon)m \rfloor$. We thus choose to round-down this number, yielding a new assumption of the form $n \leq \lfloor C(\varepsilon)m \rfloor$. Similarly a theorem's assumption of the form $n \geq C(\varepsilon)m$ becomes a new assumption of the form $n \geq \lceil C(\varepsilon)m \rceil$.

Changing the floors and ceilings respectively for ceilings and floors in the statements of the theorems makes them stronger, while preserving the results. This is because a difference of 1 in either dimension of **DLC** will not affect the success probabilities significantly. In fact, a difference of the order of the logarithm of a dimension does not affect our results, as these are formally stated towards the end of Section 1.3.2.

Similarly, the **DLC** problem statement of Section 1.3.2 includes that an m-bit error vector may have weight $\leq \varepsilon m$. Following the same logic, we instead state that the weight is $\leq \lfloor \varepsilon m \rfloor$.

Choice of distribution of the error In the references on coding theory mentioned in Section 1.3, the standard choice of distribution of the error is a Bernoulli distribution of parameter ε. We instead chose to uniformly select the error vector from all the vectors with weight no larger than $\lfloor \varepsilon m' \rfloor$. This makes it easier to generate an instance of **DLC** than with a Bernoulli-distributed error.

In the case of a Bernoulli-distributed error, the bits of the error vector are one's with probability ε, so there is no guarantee that a random error vector has fewer than a fraction ε of one's, although that is their expected fraction. Therefore, an instance generating algorithm is needed, so that instances with error vectors with two many one's are rejected. It can be a Las Vegas algorithm, which always returns a correct answer,

[1] Which would be impossible if $C(\varepsilon)$ were irrational. Such a requirement would restrict our choice of ε and m. Clearly, we wish to avoid such restrictions.

but has probabilistic running time. [MR95, Section 1.2]

All results in Chapter 3 would be affected by a change to a Bernoulli-distributed error. For instance, in the proofs of Lemmata 39 and 44, $\Pr[e = e']$ would not be the inverse of an approximation to the partial sum of a row of Pascal's triangle. If the number of one's was not limited, we would have $\Pr[e = e'] = (1 - 2\varepsilon(1-\varepsilon))^m$. For a limit of $\lfloor \varepsilon m \rfloor$ one's, the expression of this probability is more complicated. The binomial distribution has mean εm and is in some sense comparable to the distribution on errors that we chose. Hence, it is possible that changing the distribution of the error from binomial to Bernoulli would not affect the results dramatically.

Open problems

We do not know if the following are true.

$$DLC\left(n, \left\lfloor \frac{n}{C(\varepsilon)} \right\rfloor\right) \quad \propto_{av} \quad DLC\left(2n, \left\lfloor \frac{2n}{C(\varepsilon)} \right\rfloor\right)$$
$$DLC\left(n, \left\lfloor \frac{n}{C(\varepsilon)} \right\rfloor\right) \quad \propto_{av} \quad DLC\left(n^2, \left\lfloor \frac{n^2}{C(\varepsilon)} \right\rfloor\right)$$

As for **SS**, the reductions we have are of a very special nature, and we did not find a way to analyze simultaneous variations of the two parameters.

In Section 1.3, ε was fixed to be a constant between 0 and $\frac{1}{2}$. There is in fact no reason at all why ε should be constant, other than it allowed us to get our results.

Another natural version of **DLC** could hence have three parameters, (n, m, ε), where ε varies as a function of n or m. Note that if ε is very small, then a very large m is needed so that $\lfloor \varepsilon m \rfloor > 0$. Otherwise, as mentioned before for $\varepsilon = 0$, **DLC** would be trivial since it would be efficiently solvable via linear programming.

A simple open question of this form is the following. For two different fixed constants ε and ε', can we prove the following?

$$DLC\left(n, \left\lfloor \frac{n}{C(\varepsilon')} \right\rfloor, \varepsilon'\right) \quad \propto_{av} \quad DLC\left(n, \left\lfloor \frac{n}{C(\varepsilon)} \right\rfloor, \varepsilon\right)$$

Chapter 5

Appendix: Lemmata

The definitions and results stated without proofs, in this Chapter, are taken from books or papers. The details of their references are in Section 5.5.

5.1 Jensen's inequality

Definition 52 *A function f is* convex *(or* concave up*) if its value at the midpoint of any interval in its domain does not exceed its value at the ends of the interval. In other words, f is convex on $[a, b]$ if for any $x_1, x_2 \in [a, b]$*

$$f\left(\frac{1}{2}(x_1 + x_2)\right) \leq \frac{1}{2}\left(f(x_1) + f(x_2)\right)$$

Remark 53 *If f has a second derivative in $[a, b]$, then f is convex on $[a, b]$ if and only if $f''(x) > 0$ for all $x \in [a, b]$.*

The following lemma in used in Chapter 2, in Lemma 29. It is also involved in Chapter 2, in Lemma 34 and in Chapter 3, Lemmata 44 and 39, although it does not appear in them explicitly (their proofs refer to the one of Lemma 29).

Lemma 54 [Jensen's inequality] *For every function $f(x)$, convex on an open interval (a, b), the following inequality holds for any value of n*

$$f\left(\frac{1}{n}\sum_{i}^{n} x_i\right) \leq \frac{1}{n}\sum_{i}^{n} f(x_i)$$

where x_i is a point on the interval (a, b) and n is a positive integer.

5.2 Bounds on the entropy function

The definition of the entropy function was first given in Chapter 1, Definition 15.

Definition 55 [Entropy function] *For $0 < \varepsilon < 1$, the entropy function is defined as*

$$H(\varepsilon) = -\varepsilon\log(\varepsilon) - (1-\varepsilon)\log(1-\varepsilon)$$

where the logarithms are taken in base 2.

The following lemma is used in this chapter, in Lemmata 57, 60, 61, and 63.

Lemma 56 *The entropy function $H(\varepsilon)$ is strictly increasing on $0 < \varepsilon < \frac{1}{2}$.*

While the entropy function is strictly increasing on $0 < \varepsilon < \frac{1}{2}$, given $\varepsilon' \leq \varepsilon$ but sufficiently close to ε, we bound $H(\varepsilon')$ to be larger than $H(\varepsilon)$ minus a positive term that tend to 0.

The following lemma is used in this chapter, in Lemmata 60 and 62.

Lemma 57 *Let $0 < \varepsilon < \frac{1}{2}$ and $\varepsilon' = \frac{\lfloor \varepsilon m \rfloor}{m}$, with $m \in \mathbb{N}^+$. Then*

$$H(\varepsilon') > H(\varepsilon) - \frac{1}{m}\log\left(\frac{e^2}{\varepsilon}\right)$$

.

Proof: Since $\varepsilon' = \frac{\lfloor \varepsilon m \rfloor}{m}$, then $\varepsilon - \frac{1}{m} < \varepsilon' \leq \varepsilon$. By Lemma 56, $H(\varepsilon') > H(\varepsilon - \frac{1}{m})$. We expand the right hand side and reduce it to the form $H(\varepsilon)$ minus a positive constant over m.

$$\begin{aligned}
H\left(\varepsilon-\frac{1}{m}\right) &= -\left(\varepsilon-\frac{1}{m}\right)\log\left(\varepsilon-\frac{1}{m}\right)-\left(1-\varepsilon+\frac{1}{m}\right)\log\left(1-\varepsilon+\frac{1}{m}\right)\\
&= -\left(\varepsilon-\frac{1}{m}\right)\log\left(\varepsilon\left(1-\frac{1}{\varepsilon m}\right)\right)\\
&\quad -\left(1-\varepsilon+\frac{1}{m}\right)\log\left((1-\varepsilon)\left(1+\frac{1}{(1-\varepsilon)m}\right)\right)\\
&= -\varepsilon\log\varepsilon-(1-\varepsilon)\log(1-\varepsilon)-\frac{1}{m}\log\left(\frac{1}{\varepsilon}\right)-\frac{1}{m}\log(1-\varepsilon)\\
&\quad -\left(\varepsilon-\frac{1}{m}\right)\log\left(1-\frac{1}{\varepsilon m}\right)-\left(1-\varepsilon+\frac{1}{m}\right)\log\left(1+\frac{1}{(1-\varepsilon)m}\right)\\
&= H(\varepsilon)-\frac{1}{m}\left[\log\left(\frac{1}{\varepsilon}\right)+\log(1-\varepsilon)\right]-\log\left(1+\frac{1}{(1-\varepsilon)m}\right)\\
&\quad +\left(\varepsilon-\frac{1}{m}\right)\left[\log\left(\frac{1}{1-\frac{1}{\varepsilon m}}\right)+\log\left(1+\frac{1}{(1-\varepsilon)m}\right)\right]\\
&\geq H(\varepsilon)-\frac{1}{m}\log\left(\frac{1-\varepsilon}{\varepsilon}\right)-\log\left(1+\frac{1}{(1-\varepsilon)m}\right)\\
&= H(\varepsilon)-\frac{1}{m}\left(\log\left(\frac{1-\varepsilon}{\varepsilon}\right)+\log\left(1+\frac{1}{(1-\varepsilon)m}\right)^{m}\right)
\end{aligned}$$

On the fourth line above, we ignored the term $\left(\varepsilon-\frac{1}{m}\right)\log\left(\frac{1+\frac{1}{(1-\varepsilon)m}}{1-\frac{1}{\varepsilon m}}\right)$, since it is nonnegative. Next, use $\left(1+\frac{t}{n}\right)^n \leq e^t$ for all $t, n \in \mathbb{R}^+$, with $t=\frac{1}{1-\varepsilon}$ and $n=m$.

$$\begin{aligned}
H\left(\varepsilon-\frac{1}{m}\right) &\geq H(\varepsilon)-\frac{1}{m}\left(\log\left(\frac{1-\varepsilon}{\varepsilon}\right)+\log\left(e^{\frac{1}{1-\varepsilon}}\right)\right)\\
&= H(\varepsilon)-\frac{1}{m}\log\left(\frac{(1-\varepsilon)e^{\frac{1}{1-\varepsilon}}}{\varepsilon}\right)\\
&> H(\varepsilon)-\frac{1}{m}\log\left(\frac{e^2}{\varepsilon}\right)
\end{aligned}$$

The last line comes from the facts that $1-\varepsilon<1$, and that since $0<\varepsilon<\frac{1}{2}$, we have that $e<e^{\frac{1}{1-\varepsilon}}<e^2$. ■

Notice that $\frac{e^2}{\varepsilon}>1$. So $\log\left(\frac{e^2}{\varepsilon}\right)>0$ is a positive constant, and $\frac{1}{m}\log\left(\frac{(1-\varepsilon)e^{\frac{1}{1-\varepsilon}}}{\varepsilon}\right)$ tends to 0 as m increases.

5.3 Bounds in Pascal's triangle

The following lemma is used in this chapter in Lemmata 61, and 63.

Lemma 58 [Bounds for a binomial coefficient] *Suppose εm is an integer where $0 < \varepsilon < 1$. Then*

$$\frac{1}{\sqrt{8m\varepsilon(1-\varepsilon)}}\, 2^{mH(\varepsilon)} \leq \binom{m}{\varepsilon m} \leq \frac{1}{\sqrt{2\pi m\varepsilon(1-\varepsilon)}}\, 2^{mH(\varepsilon)}$$

The following lemma is used in this chapter, in Lemma 60.

Lemma 59 [Bounds for a sum of a row of Pascal's triangle] *Let $0 < \varepsilon < \frac{1}{2}$ and $m \in \mathbb{N}^+$. Suppose εm is an integer. Then*

$$\frac{1}{\sqrt{8m\varepsilon(1-\varepsilon)}}\, 2^{mH(\varepsilon)} \leq \sum_{k=0}^{\varepsilon m} \binom{m}{k} \leq 2^{mH(\varepsilon)}$$

The following lemma is used in Chapter 3, in Theorems 39 and 44.

Lemma 60 [Bounds for a sum of a row of Pascal's triangle (General Form)] *Let $0 < \varepsilon < \frac{1}{2}$ and $m \in \mathbb{N}^+$.*

$$\frac{\varepsilon}{e^2\sqrt{8m}}\, 2^{mH(\varepsilon)} < \sum_{k=0}^{\lfloor \varepsilon m \rfloor} \binom{m}{k} \leq 2^{mH(\varepsilon)}$$

Proof: Let $\varepsilon' = \frac{\lfloor \varepsilon m \rfloor}{m}$, then $0 \leq \varepsilon' \leq \varepsilon$.

First consider the RHS. By Lemma 59

$$\sum_{k=0}^{\varepsilon' m} \binom{m}{k} \leq 2^{mH(\varepsilon')}$$

Since $0 \leq \varepsilon' \leq \varepsilon$ and since the entropy function is increasing between 0 and $\frac{1}{2}$ by Lemma 56, we have $H(\varepsilon') \leq H(\varepsilon)$. It follows that

$$\sum_{k=0}^{\lfloor \varepsilon m \rfloor} \binom{m}{k} \leq 2^{mH(\varepsilon)}$$

Now, consider the LHS. We have $\varepsilon' \leq \varepsilon$. Furthermore, suppose that $\varepsilon' > 0$ (we will deal with the case $\varepsilon' = 0$ separately), so that $1 - \varepsilon' < 1$. Then, by Lemma 59

$$\sum_{k=0}^{\varepsilon' m} \binom{m}{k} \geq \frac{1}{\sqrt{8m\varepsilon'(1-\varepsilon')}}\, 2^{mH(\varepsilon')} > \frac{1}{\sqrt{8m\varepsilon}}\, 2^{mH(\varepsilon')}$$

By Lemma 57, $H(\varepsilon') > H(\varepsilon) - \frac{1}{m}\log\left(\frac{e^2}{\varepsilon}\right)$. Then

$$2^{mH(\varepsilon')} > 2^{mH(\varepsilon)-\log\left(\frac{e^2}{\varepsilon}\right)} = 2^{mH(\varepsilon)}\, \frac{\varepsilon}{e^2}$$

It follows that

$$\sum_{k=0}^{\lfloor \varepsilon m \rfloor} \binom{m}{k} > \sqrt{\frac{\varepsilon}{8m}}\, \frac{1}{e^2}\, 2^{mH(\varepsilon)} > \frac{\varepsilon}{e^2}\, \frac{1}{\sqrt{8m}}\, 2^{mH(\varepsilon)}$$

Now suppose that $\varepsilon' = 0$ and consider the LHS. Then $\sum_{k=0}^{\lfloor \varepsilon m \rfloor} \binom{m}{k} = \binom{m}{0} = 1$, so we want to show that $\frac{\varepsilon}{e^2}\frac{1}{\sqrt{8m}}\, 2^{mH(\varepsilon)} < 1$. As before, by Lemma 57, $2^{mH(\varepsilon)} < \frac{e^2}{\varepsilon}\, 2^{mH(\varepsilon')}$.

$$\begin{aligned} \frac{\varepsilon}{e^2}\, \frac{1}{\sqrt{8m}}\, 2^{mH(\varepsilon)} &< \frac{1}{\sqrt{8m}}\, 2^{mH(\varepsilon')} \\ &= \frac{1}{\sqrt{8m}} \end{aligned}$$

It is easy to see that $\frac{1}{\sqrt{8m}} < 1$ holds since $m \geq 1$. ■

The following lemma is used in this chapter, in Lemma 62.

Lemma 61 [Bounds for a sum of a row of Pascal's triangle (Improved)] *Let $0 < \varepsilon < \frac{1}{2}$, and let $m \in \mathbb{N}^+$ be sufficiently large, i.e. $m \geq \lceil 1/\varepsilon \rceil$.*

$$\frac{\varepsilon}{e^2\sqrt{8m}}\, 2^{mH(\varepsilon)} < \sum_{k=0}^{\lfloor \varepsilon m \rfloor} \binom{m}{k} < \frac{1}{(1-2\varepsilon)\sqrt{2\pi \lfloor \varepsilon m \rfloor}}\, 2^{mH(\varepsilon)}$$

Proof: The LHS is the same as Lemma 60. We want to bound the RHS using the upper bound of Lemma 58, so we want to RHS to be bounded above by $\binom{m}{\lfloor \varepsilon m \rfloor}$ times a constant.

Let $r = \frac{\varepsilon}{1-\varepsilon}$. Since $\lfloor \varepsilon m \rfloor \leq \varepsilon m$, we have that

$$\binom{m}{k} \leq r\binom{m}{k+1} \leq r^{\lfloor \varepsilon m \rfloor - k}\binom{m}{\lfloor \varepsilon m \rfloor}$$

Then

$$\sum_{k=0}^{\lfloor \varepsilon m \rfloor} \binom{m}{k} \leq \binom{m}{\lfloor \varepsilon m \rfloor} \sum_{k=0}^{\lfloor \varepsilon m \rfloor} r^k < \binom{m}{\lfloor \varepsilon m \rfloor} \frac{1-\varepsilon}{1-2\varepsilon}$$

The last line is obtained, from $\sum_{k=0}^{\lfloor \varepsilon m \rfloor} r^k$ being a geometric series, with $0 < r < 1$.

$$\sum_{k=0}^{\lfloor \varepsilon m \rfloor} r^k = \frac{1 - \left(\frac{\varepsilon}{1-\varepsilon}\right)^{\lfloor \varepsilon m \rfloor + 1}}{1 - \frac{\varepsilon}{1-\varepsilon}} < \frac{1}{1 - \frac{\varepsilon}{1-\varepsilon}} = \frac{1-\varepsilon}{1-2\varepsilon}$$

Let $\varepsilon' = \frac{\lfloor \varepsilon m \rfloor}{m}$. By the upper bound of Lemma 58

$$\binom{m}{\lfloor \varepsilon m \rfloor} \leq \frac{1}{\sqrt{2\pi m \varepsilon'(1-\varepsilon')}} 2^{mH(\varepsilon')}$$

Since $\varepsilon' \leq \varepsilon$, we have that $\frac{1}{1-\varepsilon'} \leq \frac{1}{1-\varepsilon}$. From Lemma 56, we have that $2^{mH(\varepsilon')} \leq 2^{mH(\varepsilon)}$. Since $m \geq 1/\varepsilon$, we have that $\varepsilon' > 0$, and $\lfloor \varepsilon m \rfloor \geq 1$.

$$\binom{m}{\lfloor \varepsilon m \rfloor} \leq \frac{1}{\sqrt{2\pi \lfloor \varepsilon m \rfloor (1-\varepsilon)}} 2^{mH(\varepsilon)}$$

Therefore

$$\sum_{k=0}^{\lfloor \varepsilon m \rfloor} \binom{m}{k} < \frac{1}{(1-2\varepsilon)} \sqrt{\frac{(1-\varepsilon)}{2\pi \lfloor \varepsilon m \rfloor}} \, 2^{mH(\varepsilon)} < \frac{1}{(1-2\varepsilon)} \sqrt{\frac{1}{2\pi \lfloor \varepsilon m \rfloor}} \, 2^{mH(\varepsilon)}$$

■

5.4 Technical lemmata

5.4.1 Error vectors from an instance of DLC(n, m', ε)

For $m' > m$, fix some m bit positions of any m'-bit vector f. This defines an m-bit vector e.

We are interested in m'-bit f chosen accordingly to the distribution of input error vectors for average case **DLC**(n, m', ε). In other words, f is uniformly chosen among all m'-bit strings with weight $\leq \varepsilon m'$.

We are interested in m-bit e such that $w(e) \leq \varepsilon m$, and such that all of these are equally likely. In other words, we are interested in e distributed as are the input vectors for average case $\mathbf{DLC}(n,m,\varepsilon)$. For e to be properly distributed, it is sufficient that the other $m'-m$ bits of f has weight exactly $\lfloor \varepsilon m' \rfloor - \lfloor \varepsilon m \rfloor$ (see proof of Theorem 38 in Chapter 3). Define such error strings to be "properly distributed input errors" for $\mathbf{DLC}(n,m,\varepsilon)$.

The following lemma is used in Chapter 3, in Theorem 38. Part of the proof of Lemma 63 is similar.

Lemma 62 [Fraction of properly distributed input errors for $\mathbf{DLC}(n,m,\varepsilon)$] *Let $0 < \varepsilon < \frac{1}{2}$ and $m, m' \in \mathbb{N}^+$. If m' is sufficiently large and $m' > m$, then the fraction of properly distributed input errors for $\mathbf{DLC}(n,m,\varepsilon)$ is*

$$\frac{\binom{m'-m}{\lfloor \varepsilon m' \rfloor - \lfloor \varepsilon m \rfloor} \sum_{i=0}^{\lfloor \varepsilon m \rfloor} \binom{m}{i}}{\sum_{i=0}^{\lfloor \varepsilon m' \rfloor} \binom{m'}{i}} > \frac{\sqrt{\pi}\varepsilon^{5/2}(1-2\varepsilon)}{8e^4} \frac{1}{\sqrt{m}}$$

Proof: The proof consist in applying Lemma 61 three times. To use this lemma for the for the first term, $\binom{m'-m}{\lfloor \varepsilon m' \rfloor - \lfloor \varepsilon m \rfloor}$, note that $\lceil \varepsilon(m'-m) \rceil \geq \lfloor \varepsilon m' \rfloor - \lfloor \varepsilon m \rfloor \geq \lfloor \varepsilon(m'-m) \rfloor$.

Denoting the fraction part of εm by $frac(\varepsilon m)$, this is because

$$\begin{aligned}
\lfloor \varepsilon(m'-m) \rfloor &= \lfloor \lfloor \varepsilon m' \rfloor - \lfloor \varepsilon m \rfloor + frac(\varepsilon m') - frac(\varepsilon m) \rfloor \\
&= \lfloor \varepsilon m' \rfloor - \lfloor \varepsilon m \rfloor + \lfloor frac(\varepsilon m') - frac(\varepsilon m) \rfloor \\
&\leq \lfloor \varepsilon m' \rfloor - \lfloor \varepsilon m \rfloor
\end{aligned}$$

The last line above holds since $-1 < frac(\varepsilon m') - frac(\varepsilon m) < 1$. Similarly, $\lceil \varepsilon(m'-m) \rceil \geq \lfloor \varepsilon m' \rfloor - \lfloor \varepsilon m \rfloor$.

To use the above mentioned lemma for the first term, we need that $\lfloor \varepsilon m' \rfloor - \lfloor \varepsilon m \rfloor \leq \frac{1}{2}(m'-m)$. This is true because

$$\begin{aligned}
\lfloor \varepsilon m' \rfloor - \lfloor \varepsilon m \rfloor &\leq \lceil \varepsilon(m'-m) \rceil \\
&< \left\lceil \frac{1}{2}(m'-m) \right\rceil
\end{aligned}$$

Use the above mentioned lemma's lower bound twice, and its upper bound, once.

$$\begin{aligned}
\frac{\binom{m'-m}{\lfloor\varepsilon m'\rfloor-\lfloor\varepsilon m\rfloor}\sum_{i=0}^{\lfloor\varepsilon m\rfloor}\binom{m}{i}}{\sum_{i=0}^{\lfloor\varepsilon m'\rfloor}\binom{m'}{i}} &\geq \frac{\binom{m'-m}{\lfloor\varepsilon(m'-m)\rfloor}\sum_{i=0}^{\lfloor\varepsilon m\rfloor}\binom{m}{i}}{\sum_{i=0}^{\lfloor\varepsilon m'\rfloor}\binom{m'}{i}} \\
&> \frac{\varepsilon}{e^2\sqrt{8(m'-m)}}\,2^{(m'-m)H(\varepsilon)}\,\frac{\varepsilon}{e^2\sqrt{8m}}\,2^{mH(\varepsilon)} \\
&\quad (1-2\varepsilon)\sqrt{2\pi\lfloor\varepsilon m'\rfloor}\,\frac{1}{2^{m'H(\varepsilon)}} \\
&= \frac{\sqrt{2\pi}\varepsilon^2(1-2\varepsilon)}{8e^4}\sqrt{\frac{\lfloor\varepsilon m'\rfloor}{m(m'-m)}}
\end{aligned}$$

Since $\lfloor\varepsilon m'\rfloor > \varepsilon m' - 1$, and since $m' - m < m'$, then for m' sufficiently large

$$\frac{\lfloor\varepsilon m'\rfloor}{m(m'-m)} > \frac{\varepsilon m'-1}{mm'} = \frac{\varepsilon - 1/m'}{m} \geq \frac{\varepsilon}{2m}$$

Overall

$$\frac{\binom{m'-m}{\lfloor\varepsilon m'\rfloor-\lfloor\varepsilon m\rfloor}\sum_{i=0}^{\lfloor\varepsilon m\rfloor}\binom{m}{i}}{\sum_{i=0}^{\lfloor\varepsilon m'\rfloor}\binom{m'}{i}} > \frac{\sqrt{\pi}\varepsilon^{5/2}(1-2\varepsilon)}{8e^4}\frac{1}{\sqrt{m}}$$

■

5.4.2 Error vectors from an oracle solution to DLC(n, m, ε)

Suppose an m-bit error vector e that has been returned by an oracle for **DLC**(n, m, ε). Also suppose that e has maximal weight, that is, weight exactly $\lfloor\varepsilon m\rfloor$.

We select $m' < m$ bits of e randomly and without replacement. This defines an m'-bit vector f'. We are interested in f' such that $w(f') \leq \varepsilon m'$. In other words, we are interested in f' that are as the error vectors in correct solutions to **DLC**(n, m', ε) (see proof of Theorem 40 in Chapter 3). Define such strings to be "properly distributed output errors" for **DLC**(n, m', ε).

The f''s are distributed accordingly to the hypergeometric distribution [Fel50, Section 2.5]. The desired probability is exactly

$$\begin{aligned}
\Pr[w(f') \leq \lfloor\varepsilon m'\rfloor \mid w(e) = \lfloor\varepsilon m\rfloor] &= \frac{\sum_{k=0}^{\lfloor\varepsilon m'\rfloor}\binom{\lfloor\varepsilon m\rfloor}{k}\binom{\lceil(1-\varepsilon)m\rceil}{m'-k}}{\binom{m}{m'}} \\
&= \frac{\sum_{k=0}^{\lfloor\varepsilon m'\rfloor}\binom{m'}{k}\binom{m-m'}{\lfloor\varepsilon m\rfloor-k}}{\binom{m}{\lfloor\varepsilon m\rfloor}}
\end{aligned}$$

The following lemma is used in Chapter 3, in Theorem 40.

Lemma 63 [Fraction of properly distributed output errors for DLC(n, m', ε)] *Let $0 < \varepsilon < \frac{1}{2}$ and $m, m' \in \mathbb{N}^+$. If $m > m'$, then the fraction of properly distributed output errors for* **DLC**(n, m', ε) *is*

$$\frac{\sum_{k=0}^{\lfloor \varepsilon m' \rfloor} \binom{m'}{k} \binom{m-m'}{\lfloor \varepsilon m \rfloor - k}}{\binom{m}{\lfloor \varepsilon m \rfloor}} \in \Omega(1)$$

Proof: We first note that the floor in $\lfloor \varepsilon m \rfloor$ can be dropped, for the following reason. Dropping the floor means to assume that εm is an integer. This is equivalent to considering strings of length equal to a possibly smaller m, so that εm is exactly $\lfloor \varepsilon m \rfloor$. The probability of selecting a small number of one's with these m positions, is less or equal to the probability of selecting a small number of one's with the original number of positions. Therefore, the probability obtained when dropping the floor in $\lfloor \varepsilon m \rfloor$ is a lower bound for the probability when the floor is not dropped.

In the case that ε is irrational, we consider a slightly larger or smaller, rational ε instead. This does not affect the final constant bound of this lemma, since we are not preoccupied by its dependence on the constant ε.

We will show that some number of the terms of the sum are large enough, so that their sum is bounded below by a constant. Firstly, we show that the term with $k = \lfloor \varepsilon m' \rfloor - \left\lfloor \sqrt{\varepsilon(1-\varepsilon)m'(m-m')/m} \right\rfloor$ is a lower bound for all terms with $\lfloor \varepsilon m' \rfloor - \left\lfloor \sqrt{\varepsilon(1-\varepsilon)m'(m-m')/m} \right\rfloor \leq k \leq \lfloor \varepsilon m' \rfloor$. Secondly, we show a lower bound on the term with $k = \lfloor \varepsilon m' \rfloor - \left\lfloor \sqrt{\varepsilon(1-\varepsilon)m'(m-m')/m} \right\rfloor$. Finally, we use the first two results to show that the sum of the terms with $\lfloor \varepsilon m' \rfloor - \left\lfloor \sqrt{\varepsilon(1-\varepsilon)m'(m-m')/m} \right\rfloor \leq k \leq \lfloor \varepsilon m' \rfloor$ is bounded below by a constant.

The terms of the sum are increasing w.r.t. k

For $k \leq \lfloor \varepsilon m' \rfloor - 1$, we show that

$$\frac{\binom{m'}{k}\binom{m-m'}{\varepsilon m - k}}{\binom{m}{\varepsilon m}} < \left(\frac{\binom{m'}{k+1}\binom{m-m'}{\varepsilon m - k - 1}}{\binom{m}{\varepsilon m}} \right)$$

Consider the ratio

$$\begin{aligned}
\frac{\binom{m'}{k}\binom{m-m'}{\varepsilon m-k}}{\binom{m'}{k+1}\binom{m-m'}{\varepsilon m-k-1}} &= \frac{(k+1)\,((1-\varepsilon)m-m'+k+1)}{(m'-k)\,(\varepsilon m-k)} \\
&\leq \frac{\lfloor \varepsilon m' \rfloor\,((1-\varepsilon)m-\lceil (1-\varepsilon)m' \rceil)}{(\lceil (1-\varepsilon)m' \rceil+1)\,(\varepsilon m-\lfloor \varepsilon m' \rfloor+1)} \\
&< \frac{\varepsilon m'(1-\varepsilon)(m-m')}{(1-\varepsilon)m'\varepsilon(m-m')} \\
&= 1
\end{aligned}$$

Lower bound on the smallest term to be summed

We want a lower bound on

$$\frac{\binom{m'}{k}\binom{m-m'}{\varepsilon m-k}}{\binom{m}{\varepsilon m}}$$

where $k = \lfloor \varepsilon m' \rfloor - \left\lfloor \sqrt{\varepsilon(1-\varepsilon)m'(m-m')/m} \right\rfloor$.

Use Stirling's approximation to obtain bounds on binomial coefficients. For $x \in \mathbb{R}^+$ and $x \geq 1$

$$\begin{aligned}
\binom{m}{\lfloor x \rfloor} &> \frac{1}{4\sqrt{2\pi}}\sqrt{\frac{m}{\lfloor x \rfloor\,(m-\lfloor x \rfloor)}}\,\frac{m^m}{\lfloor x \rfloor^{\lfloor x \rfloor}\,(m-\lfloor x \rfloor)^{m-\lfloor x \rfloor}} \\
\binom{m}{\lfloor x \rfloor} &< \frac{2}{\sqrt{2\pi}}\sqrt{\frac{m}{\lfloor x \rfloor\,(m-\lfloor x \rfloor)}}\,\frac{m^m}{\lfloor x \rfloor^{\lfloor x \rfloor}\,(m-\lfloor x \rfloor)^{m-\lfloor x \rfloor}}
\end{aligned}$$

In the case that $x = \varepsilon m' < 1$, we have $k = 0$, so there is only one term in the sum, which is bounded as follows.

$$\begin{aligned}
\frac{\binom{m'}{0}\binom{m-m'}{\varepsilon m}}{\binom{m}{\varepsilon m}} &= \frac{\binom{m-m'}{\varepsilon m}}{\binom{m}{\varepsilon m}} = \frac{(m-m')!(\varepsilon m)!\,((1-\varepsilon)m)!}{(\varepsilon m)!\,((1-\varepsilon)m-m')!} \\
&> \frac{1}{32\sqrt{2\pi}}\sqrt{\frac{(m-m')(1-\varepsilon)m}{m\,((1-\varepsilon)m-m')}}\,\frac{(m-m')^{m-m'}\,((1-\varepsilon)m)^{(1-\varepsilon)m}}{m^m\,((1-\varepsilon)m-m')^{(1-\varepsilon)m-m'}} \\
&> \frac{1}{32\sqrt{2\pi}}\sqrt{\frac{(m-m')(1-\varepsilon)m}{m\,(1-\varepsilon)(m-m')}}\,\frac{(m-m')^{m-m'}\,((1-\varepsilon)m)^{(1-\varepsilon)m}}{m^m\,((1-\varepsilon)(m-m'))^{(1-\varepsilon)(m-m')}} \\
&= \frac{1}{32\sqrt{2\pi}} \in \Omega(1)
\end{aligned}$$

It is not possible that $\varepsilon m - k = 0$, because it is not possible that $\varepsilon m = \lfloor \varepsilon m' \rfloor$, since $m > m'$ and εm is an integer. Therefore, in all remaining cases, $x \geq 1$ in $\binom{m}{\lfloor x \rfloor}$, so we may now apply the above bounds for the binomial coefficients.

Let $j = \left\lfloor \sqrt{\varepsilon(1-\varepsilon)m'(m-m')/m} \right\rfloor$. We first approximate the expression with the bound for binomial coefficients. Then, we approximate some of the floors, and then we collect the terms we wish to cancel.

$$
\begin{aligned}
&\frac{\binom{m'}{\lfloor \varepsilon m' \rfloor - j}\binom{m-m'}{\varepsilon m - \lfloor \varepsilon m' \rfloor + j}}{\binom{m}{\varepsilon m}} \\
>\ & \frac{\sqrt{2\pi}}{2^{13}} \cdot \\
& \sqrt{\frac{m'(m-m')\varepsilon m(1-\varepsilon)m}{m\left(\lfloor \varepsilon m' \rfloor - j\right)\left(\lceil (1-\varepsilon)m' \rceil + j\right)\left(\varepsilon m - \lfloor \varepsilon m' \rfloor + j\right)\left((1-\varepsilon)m - \lceil (1-\varepsilon)m' \rceil - j\right)}} \cdot \\
& \frac{m'^{m'}(m-m')^{m-m'}(\varepsilon m)^{\varepsilon m}\left((1-\varepsilon)m\right)^{(1-\varepsilon)m}}{m^m\left(\lfloor \varepsilon m' \rfloor - j\right)^{\lfloor \varepsilon m' \rfloor - j}\left(\lceil (1-\varepsilon)m' \rceil + j\right)^{\lceil (1-\varepsilon)m' \rceil + j}} \cdot \\
& \frac{1}{\left(\varepsilon m - \lfloor \varepsilon m' \rfloor + j\right)^{\varepsilon m - \lfloor \varepsilon m' \rfloor + j}\left((1-\varepsilon)m - \lceil (1-\varepsilon)m' \rceil - j\right)^{(1-\varepsilon)m - \lceil (1-\varepsilon)m' \rceil - j}} \\
>\ & \frac{\sqrt{2\pi}}{2^{13}} \sqrt{\frac{m'(m-m')\varepsilon m(1-\varepsilon)m}{m\varepsilon m' 2(1-\varepsilon)m' 2\varepsilon(m-m')(1-\varepsilon)(m-m')}} \cdot \\
& \frac{m'^{m'}(\varepsilon m)^{\varepsilon m}\left((1-\varepsilon)m\right)^{(1-\varepsilon)m}}{m^m\left(\varepsilon m' - j\right)^{\lfloor \varepsilon m' \rfloor - j}\left((1-\varepsilon)m' + j + 1\right)^{\lceil (1-\varepsilon)m' \rceil + j}} \cdot \\
& \frac{(m-m')^{m-m'}}{\left(\varepsilon(m-m') + j + 1\right)^{\varepsilon m - \lfloor \varepsilon m' \rfloor + j}\left((1-\varepsilon)(m-m') - j + 1\right)^{(1-\varepsilon)m - \lceil (1-\varepsilon)m' \rceil - j}} \\
=\ & \frac{\sqrt{2\pi}}{2^{14}} \sqrt{\frac{m}{\varepsilon(1-\varepsilon)m'(m-m')}} \cdot \\
& \frac{m'^{m'}\varepsilon^{\varepsilon m} m^{\varepsilon m}(1-\varepsilon)^{(1-\varepsilon)m} m^{(1-\varepsilon)m}}{m^m\left(\varepsilon m'\right)^{\lfloor \varepsilon m' \rfloor - j}\left(1 - \frac{j}{\varepsilon m'}\right)^{\lfloor \varepsilon m' \rfloor - j}\left((1-\varepsilon)m'\right)^{\lceil (1-\varepsilon)m' \rceil + j}\left(1 + \frac{j+1}{(1-\varepsilon)m'}\right)^{\lceil (1-\varepsilon)m' \rceil + j}} \cdot \\
& \frac{(m-m')^{m-m'}}{\left(\varepsilon(m-m')\right)^{\varepsilon m - \lfloor \varepsilon m' \rfloor + j}\left(1 + \frac{j+1}{\varepsilon(m-m')}\right)^{\varepsilon m - \lfloor \varepsilon m' \rfloor + j}} \cdot \\
& \frac{1}{\left((1-\varepsilon)(m-m')\right)^{(1-\varepsilon)m - \lceil (1-\varepsilon)m' \rceil - j}\left(1 - \frac{j-1}{(1-\varepsilon)(m-m')}\right)^{(1-\varepsilon)m - \lceil (1-\varepsilon)m' \rceil - j}}
\end{aligned}
$$

We cancel some terms and obtain

$$\begin{aligned}
&\frac{\binom{m'}{\lfloor \varepsilon m'\rfloor - j}\binom{m-m'}{\varepsilon m - \lfloor \varepsilon m'\rfloor + j}}{\binom{m}{\varepsilon m}} \\
> \; &\frac{\sqrt{2\pi}}{2^{14}}\sqrt{\frac{m}{\varepsilon(1-\varepsilon)m'(m-m')}}\frac{1}{\left(1-\frac{j}{\varepsilon m'}\right)^{\lfloor \varepsilon m'\rfloor - j}\left(1+\frac{j+1}{(1-\varepsilon)m'}\right)^{\lceil (1-\varepsilon)m'\rceil + j}} \cdot \\
&\frac{1}{\left(1+\frac{j+1}{\varepsilon(m-m')}\right)^{\varepsilon m - \lfloor \varepsilon m'\rfloor + j}\left(1-\frac{j-1}{(1-\varepsilon)(m-m')}\right)^{(1-\varepsilon)m - \lceil (1-\varepsilon)m'\rceil - j}}
\end{aligned}$$

Consider the term $\left(1-\frac{j}{\varepsilon m'}\right)^{\lfloor \varepsilon m'\rfloor - j} < \left(1-\frac{j}{\varepsilon m'}\right)^{\varepsilon m' - j - 1}$. Use $\left(1+\frac{t}{n}\right)^n \leq e^t$ with $\frac{t}{n} = -\frac{j}{\varepsilon m'}$ and $n = \varepsilon m' - j - 1$.

$$\left(1-\frac{j}{\varepsilon m'}\right)^{\lfloor \varepsilon m'\rfloor - j} < e^{-\frac{j(\varepsilon m' - j - 1)}{\varepsilon m'}}$$

Consider the term $\left(1+\frac{j+1}{(1-\varepsilon)m'}\right)^{\lceil (1-\varepsilon)m'\rceil + j} < \left(1+\frac{j+1}{(1-\varepsilon)m'}\right)^{(1-\varepsilon)m' + j + 1}$. Use $\left(1+\frac{t}{n}\right)^n \leq e^t$ with $\frac{t}{n} = \frac{j+1}{(1-\varepsilon)m'}$ and $n = (1-\varepsilon)m' + j + 1$.

$$\left(1+\frac{j+1}{(1-\varepsilon)m'}\right)^{\lceil (1-\varepsilon)m'\rceil + j} < e^{\frac{(j+1)\left((1-\varepsilon)m' + j + 1\right)}{(1-\varepsilon)m'}}$$

Consider the term $\left(1+\frac{j+1}{\varepsilon(m-m')}\right)^{\lfloor \varepsilon m\rfloor - \lfloor \varepsilon m'\rfloor + j} < \left(1+\frac{j+1}{\varepsilon(m-m')}\right)^{\varepsilon(m-m') + j + 1}$. Use $\left(1+\frac{t}{n}\right)^n \leq e^t$ with $\frac{t}{n} = \frac{j+1}{\varepsilon(m-m')}$ and $n = \varepsilon(m-m') + j + 1$.

$$\left(1+\frac{j+1}{\varepsilon(m-m')}\right)^{\lfloor \varepsilon m\rfloor - \lfloor \varepsilon m'\rfloor + j} < e^{\frac{(j+1)\left(\varepsilon(m-m') + j + 1\right)}{\varepsilon(m-m')}}$$

Consider the term $\left(1-\frac{j-1}{(1-\varepsilon)(m-m')}\right)^{\lceil (1-\varepsilon)m\rceil - \lceil (1-\varepsilon)m'\rceil - j} < \left(1-\frac{j-1}{(1-\varepsilon)(m-m')}\right)^{(1-\varepsilon)(m-m') - j - 1}$. Use $\left(1+\frac{t}{n}\right)^n \leq e^t$ with $\frac{t}{n} = -\frac{j-1}{(1-\varepsilon)(m-m')}$ and $n = (1-\varepsilon)(m-m') - j - 1$.

$$\left(1-\frac{j-1}{(1-\varepsilon)(m-m')}\right)^{\lceil (1-\varepsilon)m\rceil - \lceil (1-\varepsilon)m'\rceil - j} < e^{-\frac{(j-1)\left((1-\varepsilon)(m-m') - j - 1\right)}{(1-\varepsilon)(m-m')}}$$

The sum of the exponents of the last four terms is

$$
\begin{aligned}
& -\frac{j\,(\varepsilon m' - j - 1)}{\varepsilon m'} + \frac{(j+1)\,((1-\varepsilon)m' + j + 1)}{(1-\varepsilon)m'} + \frac{(j+1)\,(\varepsilon(m-m') + j + 1)}{\varepsilon(m-m')} + \\
& -\frac{(j-1)\,((1-\varepsilon)(m-m') - j - 1)}{(1-\varepsilon)(m-m')} \\
= \; & (-j + j + 1 + j + 1 - j + 1) + \frac{1}{(1-\varepsilon)m'} + \frac{1}{\varepsilon(m-m')} - \frac{1}{(1-\varepsilon)(m-m')} + \\
& j\left[\frac{1}{\varepsilon m'} + \frac{1}{(1-\varepsilon)m'} + \frac{1}{\varepsilon(m-m')} + \frac{1}{(1-\varepsilon)(m-m')}\right] + \\
& j^2\left[\frac{1}{\varepsilon m'} + \frac{1}{(1-\varepsilon)m'} + \frac{1}{\varepsilon(m-m')} + \frac{1}{(1-\varepsilon)(m-m')}\right] \\
= \; & 3 + \frac{1}{(1-\varepsilon)m'} + \frac{1}{\varepsilon(m-m')} - \frac{1}{(1-\varepsilon)(m-m')} + \\
& j\left[\frac{m}{\varepsilon(1-\varepsilon)m'(m-m')}\right] + j^2\left[\frac{m}{\varepsilon(1-\varepsilon)m'(m-m')}\right] \\
< \; & 3 + \frac{1}{(1-\varepsilon)} + \frac{1}{\varepsilon} + \sqrt{\frac{m}{\varepsilon(1-\varepsilon)m'(m-m')}} + 1 \le 4 + \frac{1}{\varepsilon(1-\varepsilon)} + \frac{2}{\sqrt{\varepsilon(1-\varepsilon)}} \\
< \; & 4 + \frac{3}{\varepsilon(1-\varepsilon)}
\end{aligned}
$$

Overall

$$
\begin{aligned}
\frac{\binom{m'}{\lfloor \varepsilon m' \rfloor - j}\binom{m-m'}{\varepsilon m - \lfloor \varepsilon m' \rfloor + j}}{\binom{m}{\varepsilon m}} \; & > \; e^{-4-3/[\varepsilon(1-\varepsilon)]}\,\frac{\sqrt{2\pi}}{2^{14}}\sqrt{\frac{m}{\varepsilon(1-\varepsilon)m'(m-m')}} \\
& \in \; \Omega\left(\sqrt{\frac{m}{\varepsilon(1-\varepsilon)m'(m-m')}}\right)
\end{aligned}
$$

Lower bound on the sum

First, we need a bound on the term with $k = \lfloor \varepsilon m' \rfloor$. From the proof of Lemma 62, it is easy to see that

$$
\frac{\binom{m'}{\lfloor \varepsilon m' \rfloor}\binom{m-m'}{\varepsilon m - \lfloor \varepsilon m' \rfloor}}{\binom{m}{\varepsilon m}} \; \in \; \Omega\left(\sqrt{\frac{m}{\varepsilon(1-\varepsilon)m'(m-m')}}\right)
$$

Finally, we sum the terms we bounded.

$$
\begin{aligned}
& \sum_{k=0}^{\lfloor \varepsilon m' \rfloor} \frac{\binom{m'}{k}\binom{m-m'}{\varepsilon m-k}}{\binom{m}{\varepsilon m}} \\
\geq \; & \sum_{k=\lfloor \varepsilon m' \rfloor - \left\lfloor \sqrt{\varepsilon(1-\varepsilon)m'(m-m')/m} \right\rfloor}^{\lfloor \varepsilon m' \rfloor} \frac{\binom{m'}{k}\binom{m-m'}{\varepsilon m-k}}{\binom{m}{\varepsilon m}} \\
> \; & \sum_{j=0}^{\left\lfloor \sqrt{\varepsilon(1-\varepsilon)m'(m-m')/m} \right\rfloor} \frac{\binom{m'}{\lfloor \varepsilon m' \rfloor - \left\lfloor \sqrt{\varepsilon(1-\varepsilon)m'(m-m')/m} \right\rfloor}\binom{m-m'}{\varepsilon m - \lfloor \varepsilon m' \rfloor + \left\lfloor \sqrt{\varepsilon(1-\varepsilon)m'(m-m')/m} \right\rfloor}}{\binom{m}{\varepsilon m}} \\
\in \; & \left\lfloor \sqrt{\varepsilon(1-\varepsilon)m'(m-m')/m} \right\rfloor \Omega\left(\sqrt{\frac{m}{\varepsilon(1-\varepsilon)m'(m-m')}}\right) \\
= \; & \Omega(1)
\end{aligned}
$$

5.5 Notes

For a proof of Jensen's inequality, see [Jen06] or [Mit70, p. 164-165]. The former is the original paper, the latter shows a generalization of Jensen's inequality.

A more detailed motivation for Definition 55 can be found in [Ros98, Section 9.3]. Lemma 56 was taken from [Ham86, Section 6.3], where it is proven and illustrated.

Section 5.3's Lemmata 58 and 59 were taken from [MS77, Section 10.11], where their proofs are given. A discussion of the fact that there is no closed form for the partial sum of a row of Pascal's triangle can be found in [GKP89, p. 165, 166] and a proof, in [GKP89, Chapter 5]. For a proof of Stirling's approximation for the factorial, see [Mit70, p. 181-185].

Bibliography

[Abr63] N. Abramson. *Information theory and coding.* McGraw-Hill, 1963.

[ABSS97] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes and systems of linear equations. *Journal of Computer and System Sciences*, 54:317–331, 1997.

[Ajt96] M. Ajtai. Generating hard instances of lattice problems. In *Proceedings of The Twenty-Eighth Annual ACM Symposium On The Theory Of Computing (STOC '96)*, pages 99–108, New York, USA, May 1996. ACM Press.

[BMv78] E. R. Berlekamp, R. J. McEliece, and H. C. A. van Tilborg. On the intractability of certain coding problems. *IEEE Trans. Information Theory*, IT-24(3):384–385, 1978.

[Bri83] E. F. Brickell. Solving low density knapsacks. *Proceedings Crypto '83*, pages 25–37, 1983.

[BvH82] A. Borodin, J. von zur Gathen, and J. Hopcroft. Fast parallel matrix and gcd computations. *Proceedings of the Twenty-Third Annual IEEE Symposium on Foundations of Computer Science*, pages 65–71, 1982.

[CJL+92] M. J. Coster, A. Joux, B. A. LaMacchia, A. M. Odlyzko, C. P. Schnorr, and J. Stern. Improved low-density subset sum algorithms. *Computational complexity*, 2(2):111–128, 1992.

[DMS99] I. Dumer, D. Micciancio, and M. Sudhan. Hardness of the minimum distance of a linear code. To appear in FOCS, 1999.

[Fan63] R. M. Fano. A heuristic discussion of probabilistic decoding. *IEEE Transactions on Information Theory*, 9:64–73, 1963.

[Fel50] W. Feller. *An Introduction to Probability Theory and its Applications.* John Wiley & Sons, Inc., New York, 1950.

[For74] G. D. Forney, Jr. Convolutional codes iii: sequential decoding. *Information Control*, 25:267–297, 1974.

[Fri86] A. M. Frieze. On the lagarias-odlyzko algorithm for the subset sum problem. *SIAM Journal of Computation*, 15(2):536–539, 1986.

[Gal68] R. G. Gallager. *Information Theory and Reliable Communiation.* John Wiley and Sons, Inc., 1968.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York, 1979.

[GKP89] R. L. Graham, D. E. Knuth, and O. Patashnick. *Concrete Mathematics.* Addison-Wesley, Reading, Massachusetts, 1989.

[GM91] Z. Galil and O. Margalit. An almost linear-time algorithm for the dense subset-sum problem. In Javier Leach Albert, Burkhard Monien, and Mario Rodríguez-Artalejo, editors, *Automata, Languages and Programming, 18th International Colloquium*, volume 510 of *Lecture Notes in Computer Science*, pages 719–727, Madrid, Spain, 8–12 July 1991. Springer-Verlag.

[Gol98] O. Goldreich. Foundations of cryptography (fragments of a book - version 2.03). manuscript, February 1998.

[Gol99] O. Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*. Springer, 1999.

[Ham86] R. W. Hamming. *Coding and Information Theory*. Prentice-Hall, New Jersey, 1986.

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[IN96] R. Impagliazzo and M. Naor. Efficient cryptographic schemes provably as secure as subset-sum. *Journal of Cryptology*, 9:199–216, 1996.

[Jen06] J. L. W. V. Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Math.*, 30:175–193, 1906.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[KPS97] H. Kellerer, U. Pferschy, and M. G. Speranzak. An efficient approximation scheme for the subset-sum problem. *Lecture Notes in Computer Science, ISAAC '97*, 1350:394–403, 1997.

[LO85] J. C. Lagrias and A. M. Odlyzko. Solving low density subset sum problems. *Journal of the ACM*, 32:229–246, 1985.

[LP98] H. R. Lewis and C.H. Papadimitriou. *Elements of the Theory of Computational Complexity, 2nd edition*. Prentice-Hall, 1998.

[Lub96] M. Luby. *Pseudorandomness and Cryptographic Applications*. Princeton University Press, Princeton, N.J., 1996.

[Mit70] D. S. Mitrinović. *Analytic Inequalities*. Springer-Verlag, Berlin, 1970.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[MS77] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error Correcting Codes.* North-Holland Pub. Co., Amsterdam, New York, 1977.

[Odl90] A. M. Odlyzko. The rise and fall of knapsack cryptosystems, in cryptology and computational number theory. *AMS Proc. Symp. Appl. Math*, 42:75–88, 1990.

[Pap94] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, Reading, Mass., 1994.

[PW72] W. W. Peterson and E. J. Weldon Jr. *Error-Correcting Codes, 2nd edition.* MIT, 1972.

[Rac98] C. Rackoff. Open questions in cryptozoology. Talk in *Workshop on Interactive Proofs, PCP's and the Fundamentals of Cryptography*, at the Fields Institute, University of Toronto, *unpublished*, May 1998.

[Rac99] C. Rackoff. Topics in cryptography. Course notes, Spring 1999.

[Ros98] S. Ross. *A First Course in Probability, 5th edition.* Prentice-Hall, New-Jersey, 1998.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423, 623–656, 1948.

[Sha49] C. E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28:656–715, 1949.

[SS79] R. Schroeppel and A. Shamir. A $T * S^2 = O(2^n)$ time/space tradeoff for certain NP-complete problems. In *20th Annual Symposium on Foundations of Computer Science*, pages 328–336, Long Beach, Ca., USA, October 1979. IEEE Computer Society Press.

[Sti95] D. R. Stinson. *Cryptography - Theory and Practice.* CRC Press, 1995.

[SW64] C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication.* Illini Books, 1964.

[TS86] G. Tinhofer and H. Schreck. The bounded subset sum problem is almost everywhere randomly decidable in $O(N)$. *Information Processing Letters*, 23(1):11–17, July 1986.

[TV91] M. A. Tsfasman and S. G. Vlădust. *Algebraic Geometry Codes.* Dodrecht: Kluwer Academic, 1991.

[Var97] A. Vardy. The intractability of computing the minimum distance of a code. *IEEE Trans. Information Theory*, IT-43(6), November 1997.

[vL82] J. H. van Lint. *Introduction to Coding Theory.* Springer-Verlag, New York, Heidelberg, Berlin, 1982.

[vLe90] J. van Leeuwen editor. *Handbook of Theoretical Computer Science.* MIT Press, Cambridge, Mass., 1990.

[Wil84] H. S. Wilf. Backtrack: an $O(1)$ expected time algorithm for the graph coloring problem. *Information Processing Letters*, 18(3):119–121, March 1984.

[Wol60] J. Wolfowitz. A note on the strong converse of the coding theorem for the general finite memory channel. *Information and Control*, 3(1):89–93, mar 1960.

[Yam97] T. Yamakami. Average case computational complexity theory. Technical Report 307/97, Department of Computer Science, University of Toronto, 1997.