
LAPPEENRANNAN TEKNILLINEN KORKEAKOULU
LAPPEENRANTA UNIVERSITY OF TECHNOLOGY

TIETEELLISIÄ JULKAISUJA 37
RESEARCH PAPERS

JUKKA ANTERO KOSKINEN

Knapsack sets for cryptography

ISBN 978-952-214-782-0 (PDF)

LAPPEENRANNAN TEKNILLINEN KORKEAKOULU
LAPPEENRANTA UNIVERSITY OF TECHNOLOGY

UDK 519.7 :
621.391.7

TIETEELLISIÄ JULKAISUJA
RESEARCH PAPERS

37

JUKKA ANTERO KOSKINEN

Knapsack sets for cryptography

Thesis for the degree of Doctor of Philosophy,
to be presented with due permission
for public examination and criticism
in Auditorium 3 at Lappeenranta University of Technology
on the 6 th of May, 1994, at 12 o'clock noon.

LAPPEENRANTA
1994

ISBN 951-763-845-0
ISSN 0356-8210

Errors:

Abstract, paragraph 3 from the bottom: "are used in Chapter 8", not 9
p. 17, line 2 below the table: "as - as - 16"
p. 56, Lemma 8.1 line 2: "integers *MA* and *MB"*
p. 64, paragraph 2, line 1 and p. 65, Example, paragraph 2, line 2: the word "generalized" is obsolete

Abstract

The basic goal of this study is to extend old and propose new ways to generate knapsack sets suitable for use in public key cryptography. The knapsack problem and its cryptographic use are reviewed in the introductory chapter. Terminology is based on common cryptographic vocabulary. For example, solving the knapsack problem (which is here a *subset sum* problem) is termed *decipherment*.

Chapter 1 also reviews the most famous knapsack cryptosystem, the Merkle-Hellman system. It is based on a superincreasing knapsack and uses modular multiplication as a trapdoor transformation. The insecurity caused by these two properties exemplifies the two general categories of attacks against knapsack systems. These categories provide the motivation for Chapters 2 and 4. Chapter 2 discusses the density of a knapsack and the dangers of having a low density.

Chapter 3 interrupts for a while the more abstract treatment by showing examples of small injective knapsacks and extrapolating conjectures on some characteristics of knapsacks of larger size, especially their density and number.

The most common trapdoor technique, modular multiplication, is likely to cause insecurity, but as argued in Chapter 4, it is difficult to find any other simple trapdoor techniques. This discussion also provides a basis for the introduction of various categories of non-injectivity in Chapter 5.

Besides general ideas of non-injectivity of knapsack systems, Chapter 5 introduces and evaluates several ways to construct such systems, most notably the "exceptional blocks" in superincreasing knapsacks and the usage of "too small" a modulus in the modular multiplication as a trapdoor technique.

The author believes that non-injectivity is the most promising direction for development of knapsack cryptosystems.

Chapter 6 modifies two well known knapsack schemes, the Merkle-Hellman multiplicative trapdoor knapsack and the Graham-Shamir knapsack. The main interest is in aspects other than non-injectivity, although that is also exploited. In the end of the chapter, constructions proposed by Desmedt et. al. are presented to serve as a comparison for the developments of the subsequent three chapters.

Chapter 7 provides a general framework for the iterative construction of injective knapsacks from smaller knapsacks, together with a simple example, the "three elements" system. In Chapters 8 and 9 the general framework is put into practice in two different ways.

Modularly injective small knapsacks are used in Chapter 9 to construct a large knapsack, which is called *the congruential knapsack*. The addends of a subset sum can be found by decrementing the sum iteratively by using each of the small knapsacks and their moduli in turn. The construction is also generalized to the non-injective case, which can lead to especially good results in the density, without complicating the deciphering process too much.

Chapter 9 presents three related ways to realize the general framework of Chapter 7. The main idea is to join iteratively small knapsacks, each element of which would satisfy the superincreasing condition. As a whole, none of these systems need become superincreasing, though the development of density is not better than that. The new knapsack systems are injective but they can be deciphered with the same searching method as the non-injective knapsacks with the "exceptional blocks" in Chapter 5.

The final Chapter 10 first reviews the Chor-Rivest knapsack system, which has withstood all cryptanalytic attacks. A couple of modifications to the use of this system are presented in order to further increase the security or make the construction easier. The latter goal is attempted by reducing the size of the Chor-Rivest knapsack embedded in the modified system.

Keywords: knapsack problem, public key cryptography

Preface

Integers can be hidden by adding them together. However there are also collections of integers where the addends of a sum can be recovered. Furthermore there seem to be ways to conceal this possibility from outsiders. These issues inspired a swimmer in the ocean of mathematics to submerge himself for several months. Perhaps the plunge was not very deep, but it still required encouragement. Fortunately it was available from the far depths of Arto Salomaa, summarized in his simple statement "work implies achievement".

Acknowledgments are also due to Alexandru Mateescu for further encouragement and Valtteri Niemi for comments. Colleagues from the Division of Applied Mathematics of Lappeenranta University of Technology provided extra oxygen for the diver by sharing their personal computers. The Mathematica software was an essential catalyst in the computational experiments.

Lappeenranta, April 1994

Jukka Koskinen

Contents

1 Introduction

This is a cryptographically oriented study of the knapsack problem, or actually of the knapsack itself. We are namely not interested in *solving* the knapsack problem. Instead we would like to create easily solvable knapsacks that look very difficult to those who do not know our secret parameters.

We base our vocabulary on the cryptographic concepts, although many issues could be presented with more common algebraic terms. Thus we will speak of a *message* that is *enciphered* with a *key* to get a *cryptotext* that is *sent* to a *receiver* who then *deciphers* it in order to recover the message. We hardly enter the world of more complicated cryptographic protocols.

Various parts of this study differ considerably in their degree of exactness. The question is usually about randomness being only approximate. An arbitrary knapsack system is not suitable for cryptographic use, but it is difficult to know how much a special choice restricts the randomness of certain parameters. This is quite typical in the field of cryptography, and especially in this subfield dealing with a public enciphering key. Also here the most rigorous parts of the study are those, that are not likely to contribute very much to practical cryptosystems.

This study presents some new concepts and results that have their own mathematical value, regardless of whether they turn out applicable in practice. These issues may have been overlooked by the mainstream of the cryptographical research of knapsacks, partly because the knapsack cryptosystems first looked very promising, and after the main systems were broken, the minor details were no more interesting.

We explicate the goals of this study in the end of this chapter, in Section 1.5. We start by describing the knapsack problem and its relation to public key cryptography in Section 1.1. As an example we review in Section 1.2 the "original" knapsack cryptosystem that was introduced by Merkle and Hellman in 1978.

In Section 1.3 we see how the concept of knapsack has been generalized. These examples will also give an impression of what there is beyond the limits of our present investigation. Some terminology and notations for this study will be defined in Section 1.4.

1.1 Knapsack problem and public key cryptography

The **knapsack problem** is either one of the following tasks that take as an input a finite set A of positive integers and an integer s.

- Decision problem: Decide whether s is a sum of some elements of A.
- Searching problem: Given that s is a sum of some elements of A, determine those elements.

Both of these problems are NP-complete (see [30] for proof, for the notion of NP-completeness see also [15]). This means that it is extremely unlikely that either of the problems can be solved in any reasonable time, if the set A

is large (say, over 150 elements) and consists of large arbitrary integers (say, $2^{150} \approx 10^{45}$).

A sum of some elements of A is referred to as a **sum instance** of the set A. Suppose that for a particular set A there is a feasible method for solving the searching problem with any sum instance. Suppose further that a person not knowing this method cannot distinguish the problem from an arbitrary knapsack problem. Such persons can of course easily generate sum instances of A but given one they are faced with the general NP-complete searching problem. This situation can be directly applied in *public key* cryptography in the following way.

The person knowing the feasible method publishes the set A as his public key. Other parties can *encrypt* their messages to him by computing sum instances: They first cut their binary message into blocks that have as many bits as A has elements. For each block they compute the sum of those elements of A that correspond to 1's in the block. The owner of the public key is able to carry out the *decryption*, i.e. find out the addends and hence the bits. As long as he keeps the method secret, this task is too difficult to anyone else, as we assumed above.

The method that transforms the public key to the feasible decryption method is generally called a **trapdoor**, which is of course a very basic notion in public key cryptosystems. Usually one starts from an easy problem and transforms it somehow to a problem that is considered difficult (in its general form). The transformed problem is then used as the public key and the inverse transformation is the trapdoor. For further information on general public key issues we refer the reader to [3], [12], [29], [31] or [34].

1.2 The Merkle-Hellman knapsack cryptosystem

There are several ways of creating a trapdoor to the knapsack problem. The first and most famous one was introduced by Merkle and Hellman in [24].

Suppose that a sequence of positive integers $a_1, a_2, a_3, \ldots, a_n$ satisfies $a_{j+1} > \sum_{i=1}^{j} a_i$, when $j = 1, 2, \ldots, n-1$. Such a sequence is called **superincreasing**. The knapsack searching problems associated with the set $A = \{a_1, \ldots, a_n\}$ are easy to solve: Given a sum instance, we subtract from it all those a_i that we can, trying always the largest one that has not yet been tried. Eventually we reach 0 and know the addends.

We transform the superincreasing sequence to another sequence $b_1, \ldots, b_n$ by multiplying each a_i by an integer w and reducing the product modulo another integer m (> 1): $b_i \equiv wa_i \pmod{m}$ and $0 \leq b_i < m$. Usually the multiplied sequence is not superincreasing, and the searching problems associated with the set $B = \{b_1, \ldots, b_n\}$ look difficult. So the new sequence, or the set B is taken as the public key.

In order to get always back to the original easy searching problems, that is, to have a proper trapdoor, we choose the secret parameters m and w in the following way.

Choosing the multiplier w relatively prime to m provides us with an inverse $u = w^{-1}$, that is, with a solution of $uw \equiv 1 \pmod{m}$. This means that the

corresponding sum instances s_a of A and s_b of B not only satisfy the congruence $s_b \equiv w s_a \pmod{m}$ but also $u s_b \equiv s_a \pmod{m}$.

Assume we have chosen the modulus m to be larger than the sum of all the a_i. Then the least non-negative remainder of $u s_b$ modulo m is not only congruent to but *equals* s_a. This means that by solving the easy searching problem with A and $u s_b \pmod{m}$ we get a solution to the seemingly difficult problem with B and s_b.

We state as a lemma the idea behind the choice of m and w.

Lemma 1.1 Assume that m, w, x and y are such integers that $0 \leq x, y < m$ and $\gcd(w, m) = 1$. Then the following equivalences hold:

$$wx \equiv wy \pmod{m} \quad \Leftrightarrow \quad x \equiv y \pmod{m} \quad \Leftrightarrow \quad x = y. \qquad \square$$

The cryptosystem with public key $\{b_1, \ldots, b_n\}$ and secret trapdoor information u and m is called the *basic Merkle-Hellman cryptosystem.*

A similar modular multiplication that transformed A to B can be applied to B. It will produce a third set C, where the easy superincreasing structure of A is obviously even more scrambled than in B. The searching problem can of course be solved as before if one knows both of the multipliers and moduli. When two or more modular multiplications are applied to a superincreasing sequence the result is called an *iterated Merkle-Hellman knapsack cryptosystem.*

The Merkle-Hellman knapsack system has been broken both in its basic and iterated form. Two main approaches can be distinguished here, as also in cryptanalysis of any knapsack systems:

1. A trapdoor can be found. The trapdoor need not be the original one, it suffices to find any such pair of integers U and M that the sequence $Ub_i \pmod{M}$ is superincreasing (in some order).

   This attack against the basic system was emerging in the early 80's and it was completed by Shamir in [36]. A more general method was presented by Salomaa in [32]. Both methods can be found also in [31].

2. The searching problem concerning a sum instance can be solved directly, without referring to any trapdoor. This was done by Lagarias and Odlyzko in [17] both for the basic and the iterated system by exploiting the fact that some of the numbers in these systems are very large in proportion to n, that is, the set has *low density.*

A mixed approach was developed by Brickell in [4] for the iterated system: First a partial superincreasing sequence is recovered from the public key. This can then be used for solving the searching problems.

Although we will avoid the superincreasing property in our constructions, we will use modular multiplication as a trapdoor technique. The main idea behind the attacks against modularly multiplied systems will be mentioned in Section 4.2. Reviews of these and other attacks can be found in [5], [6], [11] and [27].

The low density attack can be applied to any knapsack cryptosystems and hence a short account on it will be given in Section 2.2.

Example. Choose a superincreasing knapsack

$$A = \{3, 11, 17, 32, 73, 155, 304, 616, 1298, 2847, 5595, 11211\}.$$

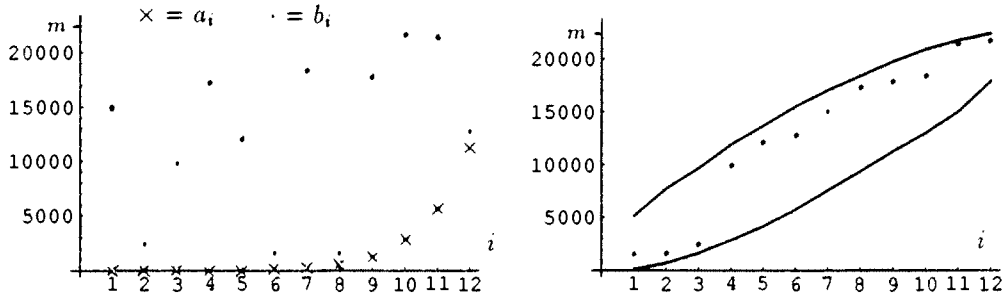
The sum of the elements of A is 22162 and thus we may choose a modulus $m = 22501$ (which is a prime) and a multiplier $w = 12495$. Multiplying elements of A by w and taking the remainders modulo m gives the set $B =$

$$\{14984, 2439, 9906, 17323, 12095, 1639, 18312, 1578, 17790, 21685, 21419, 12720\}$$

$$= \{1578, 1639, 2439, 9906, 12095, 12720, 14984, 17323, 17790, 18312, 21419, 21685\}.$$

The sets A and B are shown in the graph on the left with the corresponding elements above each other. The elements of B are shown in increasing order on the right. The lines in that graph depict more generally how the modular multiplication distributes the elements.

We chose 1000 random integers w from the range $11000, \ldots, 22000$ and multiplied the elements of A by each w modulo m. For each w we sorted the resulting set B_w. At each $i = 1, \ldots, 12$ the lower line shows a lower bound for 95% of the i'th elements in the ordered sets B_w. Similarly, at each i there are only 50 of the i'th elements above the upper line. What we want to show is that the elements are roughly evenly dispersed in the range $1, \ldots, m$.



The maximal element of B is almost twice as large as that of A. If we now want to iterate the Merkle-Hellman construction we need a modulus larger than the sum of B, which is 151 890. Assume we choose a modulus $m' = 152\,077$ and a multiplier $w' = 74\,865$. We get a new knapsack with 129 673 as the largest element. The sum of the new knapsack is as large as 1 055 868. A third modular multiplication would cause some elements to become nearly this large.

1.3 A variety of knapsack cryptosystems

The knapsack problem could of course be considered in any semigroup: Assume that G is a semigroup with respect to the binary operation $\oplus$, and the knapsack $A = \{a_1, \ldots, a_n\}$ is a finite nonempty subset of G. The knapsack (searching) problem associated with the set A is now:

Given an element $s \in G$, determine such indices $i_1, i_2, \ldots, i_r$ (if they exist) that $s = a_{i_1} \oplus a_{i_2} \oplus \cdots \oplus a_{i_r}$.

We have no need to require that the indices in the solution are distinct. Generally each element of A may have taken part more than once in the operation that gives s, which means that the solution is not determined by a *subset* of A. Furthermore, if the operation $\oplus$ is not commutative, then the order of the elements is significant and this must be taken into account also in the possible cryptographic usage (in coding messages to sequences of indices).

Multiplication of integers offers the simplest example of an easily solvable knapsack problem that does not deal with addition of integers. Assuming that the elements of A are relatively prime integers (and none equals 1) we can easily solve any subset product p of A: An element a of A is present in p if and only if a divides p. This *multiplicative* knapsack also offers the possibility of letting each element appear more than once in the operation.

In their seminal article [24] Merkle and Hellman also showed a way to transform the multiplicative knapsack to an additive trapdoor knapsack. We will review their method in Section 6.1.

We mention here a variety of additive knapsack problems, which do not involve subset sums of integers. The references show how to construct the cryptographic trapdoors.

- The knapsack is multidimensional, that is, its elements are vectors. Encryption means componentwise addition of a subset of these vectors. See for example [20] and [25]. In both of these systems the public key contains a modulus that is used to reduce the components of the sum.
- The knapsack elements belong to a finite field $GF(p^k)$, which means that they are polynomials over integers modulo p with degree at most $k-1$. Encryption is similar to addition of vectors modulo p. See [8].
- The knapsack consists of a few integers only. Sum instances are formed by adding these integers with multiplicities that can be fairly large integers. See [23] and [14]. Also in these systems the sum is reduced modulo an integer.
- The knapsack consists of disjoint sets of integers. One element from each set is used to form a sum instance. See [37].

We will restrict ourselves to such systems where

1. the knapsack consists of positive integers,
2. instances for the searching problem are formed additively (possibly with modular reduction) and
3. every sum instance is a subset sum.

We will omit the third requirement for a moment when presenting an encryption scheme in Section 2.4 and a modification in Section 6.1 (p. 46) and also when reviewing a specific system in Chapter 10. The checksum knapsack technique proposed in Section 5.3 can be viewed as a deviation from the first requirement.

1.4 Definitions and notations

We now define some more vocabulary that will be used in this study. Some concepts were introduced already before but they are defined here again. Some further definitions will be given in Section 5.1.

The basic concept is that of a knapsack. We want to manipulate it as a set, but on the other hand we want to keep it in order. The former need is stronger and so we define a **knapsack** as a finite nonempty set of positive integers. We denote knapsacks with capital letters $A, B, \ldots$ and their elements with corresponding small letters with subscipts: The knapsack A containing n elements is thus $A = \{a_1, a_2, \ldots, a_n\}$.

If not stated otherwise, when indexing the elements of a knapsack we assume that they appear in increasing order: $\min(A) = a_1 < a_2 < \ldots < a_n = \max(A)$. This means that we may consider a knapsack also as a **vector** or **sequence** of integers. In this ordered interpretation we always assume that the elements are distinct.

The number of elements in a knapsack A is denoted by $|A|$ and called the **size** of A. A **subset sum** of A means the sum of elements of some subset of A. Empty set as a subset gives the sum of no elements, which is defined to be 0. A **sum instance** of a knapsack can be a subset sum, but also a more general sum of elements.

When referring to subset sums we use the word *different* to indicate that their addends are not the same, and the word *distinct* to indicate that their values are not the same, or in other words they do not *coincide*. A knapsack is said to be **injective** if all its subset sums are distinct.

The vector interpretation of A provides another way to express the subset sums: they are of the form $S \cdot A$, where S is a binary vector of the same size as A and $\cdot$ denotes the scalar product of vectors.

Instead of the phrase "to solve the searching problem" we use the verb to **decipher** and the nouns **decipherment** or **decryption**. Deciphering a knapsack A thus refers to solving the addends of either a *given* or *any* sum instance of A, depending on the context. The decipherment of a knapsack is said to be **easy** if the knapsack belongs to a class where the decipherment time for any subset sum of any of the knapsacks is bounded by a low degree polynomial of the knapsack size. The time is commonly measured in terms of basic arithmetic operations required by the decryption algorithm. If a knapsack is easy to decipher we often simply say that the knapsack is easy. A knapsack is difficult if we do not know a polynomial time algorithm to decipher it (or actually the whole class that it represents).

With **the sum** of a knapsack A we mean the sum of all its elements and denote it by s_A. The **density** of a knapsack A is defined as

$$\rho(A) = \frac{|A|}{\log_2 \max(A)},$$

provided that $A \neq \{1\}$.

A knapsack $A = \{a_1, a_2, \ldots, a_n\}$ is **superincreasing** if its elements satisfy the condition $a_i > \sum_{j=1}^{i-1} a_j$ for $i = 2, \ldots, n$.

Assume m is an integer greater than 1. Integers in the range $0, \ldots, m-1$ represent the elements of the ring Z_m $(= Z/mZ)$ and they are called **modular integers** modulo m. Thus an integer a becomes a modular integer when we compute its least non-negative remainder modulo m. This remainder is denoted by $a \pmod m$. It is also called a *residual* and it is obtained by *reducing* a modulo m.

Operations on knapsacks

Assume that A and B are knapsacks.

1. If $A \cap B = \emptyset$, w and d are integers, $w > 0$, $d > -\min(A)$, we define the knapsacks wA, $A + d$ and $A \cup B$ as follows:

   - wA is obtained from A by multiplying each element of A by w.
   - $A + d$ is obtained from A by adding d to each element of A.
   - $A \cup B$ is the set theoretic union of A and B.

2. Assume that the integers m and w satisfy $m > \max(A)$ and $\gcd(w, m) = 1$. Let d be any integer.

   The knapsacks $wA \pmod m$ and $A + d \pmod m$ are defined to consist of the least *positive* remainders modulo m of the elements of wA and $A + d$.

   If $m > s_A$ we say that the knapsack $wA \pmod m$ is obtained by **strong modular multiplication** from A and m is a **strong modulus** (for A).

Remark. The requirement of positive remainders in the definition differs from non-negative remainders only in case $a + d \equiv 0 \pmod m$ for some $a \in A$. In this case $A + d \pmod m$ is defined to contain m, and not 0. We see that all the newly defined sets consist of positive integers, which means that they indeed are knapsacks.

All the operations except the union can be interpreted as transformations where an element x of the knapsack A is mapped to wx, $x + d$, or to a remainder of these modulo m. It is clear that all these mappings are one-to-one. In the modular cases this can be seen with Lemma 1.1. Since there is a correspondence between the elements of the original and the transformed knapsack, there is also a correspondence between their subsets. This gives a meaning to the frequent expression *corresponding subset sums*.

1.5 Problems

Desmedt, Vandewalle and Govaerts define in [10] the class of cryptographically useless knapsacks as those knapsacks that cannot be obtained from a certain class of easy knapsacks by strong modular multiplication (this class will be mentioned in Section 6.3). They show that the useless class is not empty. Obviously most of the really difficult knapsacks belong to that class. The cryptographically oriented research of the knapsack problem should then try

to reduce the useless class by providing both larger classes of easy knapsacks and other types of trapdoor transformations.

The literature on knapsack cryptosystems does not however show much optimism concerning the possibilities of these systems. When mentioning knapsack system in [3, pp. 27–28] as one of the best known candidates for public-key cryptosystem, Brassard wrote the following: "One of them, Merkle and Hellman's so-called *knapsack* cryptosystem was eventually broken. Although there still are unbroken variations on the original scheme, it does not seem advisable to trust on them."

Most of the authors seem to make a similar cautious statement that *almost* all knapsacks systems have been broken and give the one and same example of an unbroken system. For example Odlyzko wrote in [27]: "While most knapsack systems have been broken, there are a few that so far have resisted all attacks. One of the most attractive of these is the Chor-Rivest cryptosystem, which involves a combination of number theory ideas and knapsacks."

The fact that the knapsack problem is NP-complete does not necessarily help, because this concept only deals with the worst case complexity. It may be that almost all knapsack problems turn out to be easy after all. It seems unlikely that the most difficult knapsacks could ever be used as public keys, because the requirement of a practical trapdoor would mean that the problem is not genuinely difficult.

Although Odlyzko gave his article [27] the pessimistic title 'The rise and fall of knapsack cryptosystems', he wrote also: "The search for secure knapsacks continues, both because of the attractively high speed that knapsack systems offer, and because of the desire to have a wide variety of cryptosystems available." We hope to contribute to this search. And regardless of the pessimism expressed above, we try to do this in the very elementary case of additive integer knapsacks and their subset sums. We investigate the following problems in this thesis.

1. How to construct a knapsack that is
   - injective,
   - dense or at least not superinceasing and
   - easy to decipher?
2. Are there other simple trapdoor techniques than the strong modular multiplication?
3. What could non-injective knapsacks offer to public-key cryptography?
4. Since the Chor-Rivest knapsacks of sufficient size are not very easy to construct and use, could we make use of smaller ones in connection with simpler knapsacks of other type?
5. What kind of insight could experiments with small knapsacks offer concerning the following questions?
   - What is the maximal density of an injective knapsack of given size?

- What is the number of injective knapsacks of given size and density?
- Given the number of elements and the density, what is the smallest possible sum of an injective knapsack?

We will present a couple of new constructions and modify some old knapsack cryptosystems. The development of density is satisfactory in one of the constructions (Chapter 8.1), quite poor in the others (Chapter 9) and intolerable in one of the modifications (Section 6.1) — even if it improves the old system.

We believe that non-injectivity is the most promising direction for development of knapsack cryptosystems. In addition to theoretical and general considerations we will take explicit steps in this direction with one of the new constructions (Section 8.3). Non-injectivity will also provide our only answers to the question concerning new trapdoor transformations.

We will also present a couple of modifications to the use of the Chor-Rivest knapsack system. The treatment of the last question will be heuristic. We cannot present any general answers yet.

2 About the density

2.1 Two bounds for the density

For a knapsack A of size n there are 2^n subset sums. If A is injective, these must be all distinct, which means that the largest of them must be at least $2^n - 1$ (the smallest is 0). Hence $s_A \geq 2^n - 1$. On the other hand, denoting $a = \max(A)$, we have $s_A \leq \sum_{i=0}^{n-1}(a-i) = na - \frac{1}{2}n(n-1)$. Hence the largest element of an injective knapsack satisfies

$$a \geq \tfrac{1}{n}s_A + \tfrac{1}{2}(n-1) \geq \tfrac{1}{n}(2^n - 1) + \tfrac{1}{2}(n-1).$$

For large knapsacks this is not very accurate as a lower bound, because an injective knapsack cannot contain many elements from the beginning of the sequence $a, a-1, a-2, a-3, \ldots$. Thus we may as well make the bound somewhat more inaccurate by dropping the asymptotically small terms. It turns out that their value is positive as soon as $n \geq 3$ and we have then $a > \frac{1}{n}2^n$. This also gives an upper bound for density as stated in the following lemma.

Lemma 2.1 An injective knapsack A with $|A| = n$ satisfies $s_A \geq 2^n - 1$. If further $n \geq 3$, then

$$\max(A) > \frac{2^n}{n} \quad \text{and} \quad \rho(A) < \frac{n}{n - \log_2 n}. \qquad \square$$

Suppose $A = \{1, 2, 4, \ldots, 2^{n-1}\}$. Then $s_A = 2^n - 1$. It is clear that A is the only injective knapsack of size n that has $2^n - 1$ as its sum. This knapsack A is also the densest possible superincreasing knapsack with n elements. Its density is denoted by $\rho_2(n)$. Thus

$$\rho_2(n) = \frac{n}{\log_2 2^{n-1}} = \frac{n}{n-1} = 1 + \frac{1}{n-1}.$$

Even for a dense injective knapsack A the density cannot differ very much from the highest superincreasing density $\rho_2(|A|)$. We see namely from Lemma 2.1 that the upper bound for the density of an injective knapsack tends to 1 as the knapsack size grows. Also $\rho_2(n) \to 1$ as $n \to \infty$. The following table shows the upper bound $\rho_b(n)$ of the lemma for some knapsack sizes n together with $\rho_2(n)$.

n	$\rho_2(n)$	$\rho_b(n)$
10	1.1111	1.497
20	1.0526	1.276
50	1.0204	1.127
100	1.0101	1.071
150	1.0067	1.051
200	1.0050	1.040
250	1.0040	1.033

Lower bounds for the density of the densest knapsack of given size can be computed from the following lemma. Using $\{1\}$ as the initial knapsack A_0 gives only the superincreasing density ρ_2. To exceed ρ_2 generally we need first find at least one knapsack that does this. The smallest possible is $A_0 = \{3, 5, 6, 7\}$, which has $\alpha = \frac{7}{8} = 0.875$. In Chapter 3 we will see more examples.

The lemma also shows that the number of injective knapsacks grows very fast with the size. The construction of A_k in the lemma is a special case of the method of Desmedt et al., cited in Section 6.3.

Lemma 2.2 Let A_0 be an injective knapsack of size n. Denote $\alpha = \dfrac{\max(A_0)}{2^{n-1}}$. Construct the knapsacks A_i, $i = 1, \ldots, k$ by defining

$$A_i = 2A_{i-1} \cup \{b_i\}, \quad \text{where } b_i \text{ is odd}, \quad 1 \leq b_i < 2\max(A_{i-1}).$$

(i) A_k is an injective knapsack with $n + k$ elements.

(ii) $\rho(A_k) = \dfrac{n+k}{n+k-1+\log_2 \alpha}$, which is higher than $\rho_2(n+k)$, if $\alpha < 1$.

(iii) For each knapsack A_0 there are $\alpha^k 2^{kn+k(k-3)/2}$ different knapsacks A_k.

(iv) If A_0 is easy to decipher, then the same is true of A_k.

Proof. Decryption of a subset sum s of A_k follows the construction of the knapsack in the reverse order: repeat the following steps for $i = k, k-1, \ldots, 1$.

> If s is even, then b_i is not in the sum; compute $s := \frac{1}{2}s$.
> Otherwise b_i is in the sum and $s - b_i$ is even.
> In this case compute $s := \frac{1}{2}(s - b_i)$.

After these k steps we know which of the elements b_i were in the sum and the remaining value of s is a sum instance of the initial knapsack A_0. This proves the injectivity. If A_0 is easy to decipher then the same is true for A_k.

The knapsacks A_i satisfy $\max(A_i) = \alpha 2^{n-1+i}$, for $i = 0, 1, \ldots, k$. Using $i = k$ we get the density. On the other hand we see that at the construction step i there are $\max(A_{i-1}) = \alpha 2^{n-2+i}$ alternatives for b_i. Every alternative leads to a different result in the end. Hence the number of different knapsacks A_k is

$$\prod_{i=1}^{k} \alpha 2^{n-2+i} = \alpha^k 2^{kn+k(k-3)/2}.$$

□

Remark. The construction of the lemma gives disjoint collections of new knapsacks for every different initial knapsack A_0 of the same size.

2.2 When is the density "low"?

We will describe briefly the idea of the low density attack proposed by Lagarias and Odlyzko in [17]. The improved algorithm of [9] has the same principle. As a result of these attacks it seems that densities below 0.95 must be considered low in the sense that they cause insecurity. We will also discuss the possibility of higher densities being more secure.

Assume that the knapsack is $A = \{a_1, a_2, \ldots, a_n\}$ and we are given also an integer s that is known to be a subset sum of A. Thus

$$s = \sum_{i=1}^{n} e_i a_i,$$

where each e_i is 0 or 1. The task is to find the e_i.

Define the $n+1$-dimensional vectors $\bar{b}_i$

$$\begin{aligned}
\bar{b}_1 &= (1, 0, 0, \ldots, 0, a_1) \\
\bar{b}_2 &= (0, 1, 0, \ldots, 0, a_2) \\
&\vdots \qquad\qquad \vdots \\
\bar{b}_n &= (0, 0, 0, \ldots, 1, a_2) \\
\bar{b}_{n+1} &= (0, 0, 0, \ldots, 0, -s)
\end{aligned}$$

The set L consisting of all linear combinations of the $\bar{b}_i$ with integer coefficients is a **lattice** spanned by the b_i. Thus

$$L = \{c_1 \bar{b}_1 + \ldots + c_{n+1} \bar{b}_{n+1} \mid c_i \in Z\}. \tag{1}$$

Since the vectors $\bar{b}_i$ are linearly independent in R^{n+1}, all of them are needed to span L. Any collection of $n+1$ vectors that span L is called a **basis** of L.

What makes the lattice L interesting is that the vector

$$\hat{e} = (e_1, \ldots, e_n, 0)$$

belongs to it and is likely to be the shortest non-zero vector of the lattice if the density of A is low. The first statement is trivial and the conditions for the second one have been the subject of elaborate study for example in [17], [13], [9] and [33].

If the vector $\hat{e}$ indeed is the shortest or even among the shortest vectors in the lattice, then the cryptanalyst has good chances of success, that is finding $\hat{e}$. There are namely polynomial time algorithms that find short vectors in lattices. Actually these algorithms produce such bases for the lattice that all the vectors in them are relatively short. They are called *basis reduction algorithms*. The most popular algorithm has been that of Lenstra, Lenstra and Lovász [18], but more powerful modifications of it have emerged recently in e.g. [33].

According to [9] the exact complexity of finding short vectors is not known, it may be very hard in worst cases but easy on average. Indeed, none of the current algorithms are quaranteed to find the shortest vector, but Odlyzko writes in [27]: "Thus, in judging the security of knapsack cryptosystems, it is probably prudent to assume that shortest nonzero vectors in lattices can be found efficiently."

Let us turn to the probability that $\hat{e}$ is the shortest vector in the lattice L. Before citing the rigorous results we will try to make it intuitively understandable why the vector $\hat{e}$ is likely to be among the shortest ones when the knapsack has low density. First note that the Euclidean norm $\|\hat{e}\|$ of $\hat{e}$ is at most $\sqrt{n}$, and it is likely to be about $\sqrt{n/2}$, because approximately one half of the e_i are expected to be 1's and the rest 0's.

Assume $\bar{x} = (x_1, \ldots, x_{n+1}) \in L$. If x_{n+1} is not 0, then it is likely to be quite large in comparison to $\sqrt{n}$, like the last components of the $\bar{b}_i$. Actually this could be guaranteed by choosing the last components of the $\bar{b}_i$ to be Na_i with $N > \sqrt{n}$ (instead of $N = 1$) as suggested in [9].

Suppose then that $\bar{x} \neq \hat{e}$, but $x_{n+1} = 0$ and the length of $\bar{x}$ is near the length of $\hat{e}$. Then the nonzero components of $\bar{x}$ have small absolute values. They cannot however all be ± 1, if the knapsack A is injective, because now

$$\sum_{i=1}^{n} x_i a_i = 0. \tag{2}$$

The lower the density of the knapsack, the less likely become this kind of dependencies among the elements a_i with coefficients near 0.

As an example assume that A is a superincreasing knapsack in which the minimal "marginal" d of the superincreasing construction is fairly large. More formally we define this d as the smallest of the differences $a_i - \sum_{j=1}^{i-1} a_j$, for $i = 1, \ldots, n$. Here $i = 1$ gives $a_1 - 0$, hence also $d \geq a_1$.

Denote by S the ordered set of all nonzero subset sums of A. The set S is sparse in the sense that its successive members differ by at least d. The set S is also a knapsack and we will next consider its subset sums.

Suppose a vector $\bar{x} \in L$ satisfies (2) and is short: $\|\bar{x}\| \approx \|\hat{e}\| \leq \sqrt{n}$. Then each component of $\bar{x}$ is at most $\sqrt{n}$ (and $x_{n+1} = 0$). Such a vector $\bar{x}$ corresponds to an equation of two different subset sums of the set S, each having at most $\sqrt{n}$ addends. To see this move all negative addends of the sum in (2) to the right hand side of the equation. Then split the sums on both sides to (as few as possible) such sums in which each element of A appears at most once. These are subset sums of A and thus elements of S, and there are obviously at most $\sqrt{n}$ of them on each side.

Since the gaps in the set S are large, it may well happen that no two subset sums of S with fairly small number of addends coincide. This conclusion is of course only heuristic, but it would mean that there are no dependencies (2) with small $\|\bar{x}\|$. Notice that non-existence of such dependencies in a superincreasing knapsack A does not imply that there would not be any in a knapsack that is obtained from A by strong modular multiplication(s).

Critical densities

Let us assume for the next sentence that $\|\hat{e}\| \leq \sqrt{n/2}$. The result of Lagarias and Odlyzko in [17] says that for almost all knapsacks with density lower than the critical value 0.645, $\hat{e}$ will be the shortest non-zero vector in the corresponding lattice.

Since $\hat{e}$ corresponds to the subset sum s of A, the vector $\hat{e}' = (1-e_1, \ldots, 1-e_n, 0)$ corresponds to the subset sum $s_A - s$. One of the vectors $\hat{e}$ and $\hat{e}'$ has a length at most $\sqrt{n/2}$. Thus constructing two lattices, one for the subset sum s and one for $s_A - s$, we know that we have one lattice where there is a nonzero vector with length at most $\sqrt{n/2}$. And by the result cited above we know that almost surely that vector is the shortest nonzero vector in the lattice, provided that $\rho(A) < 0.645$.

In [9] the lattice (1) has been replaced by another, that reduces the effect of the small dependencies (2). As a result the critical density is improved to the value 0.9408 and this applies regardless of the (nonzero) length of the vector $\hat{e}$. This means that we need construct only one lattice, and if $\rho(A) < 0.9408$, then almost surely $\hat{e}$ will be the shortest nonzero vector in the lattice.

Are there "safe" densities?

The empirical tests in [33] indicate that it is not only low density that makes the knapsack vulnerable. With the improved lattice mentioned above and various efficient basis reduction algorithms it was possible to decipher also knapsacks with very high density. This phenomenon was not explained, but it was mentioned that "The hardest subset sum problems turn out to be those that have a density that is slightly larger than 1, i.e. a density about $1 + \frac{\log_2(n/2)}{n}$." This density is closer to the upper bound for injective density (Lemma 2.1) than to $\rho_2(n)$. The experiments concerned knapsacks of sizes 42, 50 and 58. The best algorithm solved nearly all of the problems. Even at the mentioned density with $n = 58$ it failed only in 5 cases out of 20.

We will next try to support heuristically the claim that there are many small dependencies (2) if the knapsack is dense. This would then support the belief in security of such a knapsack against the low density attack, at least in the original form of Lagarias and Odlyzko [17]. In the improved lattice of [9] the small dependencies correspond to slightly longer vectors than in the lattice (1).

Consider the knapsack $A = \{1, 2, 4, \ldots, 2^{n-1}\}$. There are several short vectors in the associated lattice (1) regardless of the subset sum involved. For example the $n+1$-dimensional vectors

$$\bar{v}(t,u) = (\underbrace{0,\ldots,0}_{t}, 2, \underbrace{1,\ldots,1}_{u}, -1, 0, \ldots, 0, 0).$$

belong to L for $t, u \geq 0$ and $t + u \leq n - 2$. The length of $\bar{v}(t,u)$ is $\sqrt{u+5}$. Assuming $n \geq 11$ we have $\|\bar{v}(t,u)\| < \sqrt{n/2}$ when $0 \leq u < n/2 - 5$ and $0 \leq t \leq n - u - 2$.

Summing $\sum_{u=0}^{\lfloor n/2 \rfloor - 6}(n - u - 1)$ and estimating downwards we get that there are at least $\frac{3}{8}n^2 - 7n + 22$ such vectors $\bar{v}(t,u)$ whose length is smaller than $\sqrt{n/2}$.

If $n \geq 100$ we know that there are more than 3072 vectors shorter than $\sqrt{n/2}$. Several hundreds of these vectors are even much shorter than $\sqrt{n/2}$. As an intuitive generalization it would seem that there are fewer but still many short vectors also in lattices corresponding to other knapsacks that have approximately the same density as A, which is $\rho_2(n)$. We expect to see fewer short vectors because the subset sums of such knapsacks are spread over a wider interval and do not necessarily form a sequence of successive integers as in case of A.

2.3 Decryption by lowering the density?

We know now that almost all knapsacks of density lower than some critical limit ρ_c can be deciphered by the lattice reduction approach, provided that the lattice reduction algorithms are efficient enough. Furthermore we have seen that these algorithms are considered quite efficient.

Assume A is a knapsack and s is a subset sum of A. Choose such a constant w that the density of the knapsack wA is below ρ_c. Use the low density approach to decipher the knapsack wA with subset sum ws.

Alternatively choose such a constant d that the density of the knapsack $A + d$ is below ρ_c. Using the low density attack try to decipher the knapsack $A + d$ with a candidate subset sum $s + hd$ for $h = 1, 2, \ldots, n$. If a solution is found, check whether it also gives the subset sum s for A. If not, continue until this condition is met. This will eventually happen, because $s + hd$ is the corresponding subset sum of $A + d$, if there are h addends in s.

Question. Does this mean that we have found an algorithm for the general knapsack problem, that would run in polynomial time if only the lattice reduction algorithm did the same? The multiplication approach namely seems to require the same time as the lattice reduction and the addition approach at most n times this.

Of course it is only almost all low density knapsacks that the lattice approach will decipher and the phrase 'almost all' refers to all such knapsacks of size n where the elements are chosen randomly from the interval $[1, 2^{n/\rho_c}]$.

Among these there are proportionally very few knapsacks where the elements either have a large common factor or are clustered near $2^{n/\rho_c}$. Obviously the low density algorithms do not work so well for these knapsacks. Otherwise there really would be chances to decipher any knapsack by multiplying or by adding a huge constant. Seeing this from the other side, it is not difficult to imagine that constructions (like those in Chapter 9) where dense knapsacks are joined together after multiplications and/or additions, could possess some immunity against the low density attack, even if they lead to knapsacks with not very high density.

A little less imagination is needed for the following conclusion.

Proposition. Increasing the density of a knapsack by subtracting a large integer is not likely to increase the security against the low density attack.

A more optimistic view of this is of course, that even when an increase of density is possible by subtraction, we perhaps need not do it. Then we maintain the injectivity and need not guess (or request from the sender) the number of addends in the subset sums that we have to decipher.

Subtraction can however be helpful in (partially) concealing some vulnerable structure of the knapsack as for example in the system of Section 8.1.

2.4 An attempt to avoid the low density attacks

The lattice basis reduction must be carried out separately for every sum instance that we want to decipher with the low density approach. In addition the reduction algorithms do not always find the shortest vectors even if the lattice were good for all sum instances in the sense that there are no small dependencies in the equation (2).

In the following system encryption is chained in order to exploit the above difficulties of the low density attack.

Assume that the public key consists of a knapsack A of size n and a collection of functions $f_1, \ldots, f_r$, that map n-dimensional binary vectors to n-dimensional vectors with integer components. The f_i can be for example permutations.

To encipher a message choose random vectors $\bar{e}_1, \ldots, \bar{e}_r \in \{0,1\}^n$ with roughly $n/2$ ones in each.

Denote the binary message vector by $\bar{e}_{r+1}$. Regard also the set A as a vector. Compute and send the following sum instances. Notice that they are not necessarily subset sums, and there may also be negative addends if the f_i produce negative components.

$$\begin{aligned}
s_1 &= \bar{e}_1 \cdot A \\
s_2 &= (\bar{e}_2 + f_1(\bar{e}_1)) \cdot A \\
s_3 &= (\bar{e}_3 + f_1(\bar{e}_2) + f_2(\bar{e}_1)) \cdot A \\
&\vdots \\
s_{r+1} &= (\bar{e}_{r+1} + f_1(\bar{e}_r) + f_2(\bar{e}_{r-1}) + \ldots + f_r(\bar{e}_1)) \cdot A.
\end{aligned}$$

Legal decryption starts from the subset sum s_1. After $\bar{e}_1$ is found, $f_1(\bar{e}_1) \cdot A$ is subtracted from s_2. The result is $\bar{e}_2 \cdot A$, from which $\bar{e}_2$ can be found. Then $(f_1(\bar{e}_2) + f_2(\bar{e}_1)) \cdot A$ is subtracted from s_3 and the decryption continues in the obvious way.

Also the enemy must know all $\bar{e}_1, \ldots, \bar{e}_r$ in order to find the vector $\bar{e}_{r+1}$. He cannot even verify his guess without knowing the other vectors. If he proceeds in the same order as the legal decryption he may get stuck to a difficult subset sum before reaching $\bar{e}_{r+1}$. It is not likely that decryption of any $\bar{e}_i \cdot A$ could be aided by some knowledge from s_{i+1}. The main reason for this is that s_{i+1} is a sum instance which is likely to be very difficult to decipher. This applies also to the lattice approach because the special vector (the $\hat{e}$) in the corresponding lattice is usually not binary and thus not very short. On the other hand, even knowledge of the addends of s_{i+1} and their multiplicities may not reveal $\bar{e}_{i+1}$.

A disadvantage of this method is that the enciphered message requires $r+1$ times more space than one subset sum. But the chaining can be continued for any number of message vectors and no new extra space is required. For example if $\bar{e}_{r+2}$ is a second message vector, the next sum will be

$$s_{r+2} = (\bar{e}_{r+2} + f_1(\bar{e}_{r+1}) + f_2(\bar{e}_r) + \ldots + f_r(\bar{e}_2)) \cdot A.$$

Further message vectors can be enciphered accordingly. If this kind of chaining really helps against the low density attack then it can be advantageous to create the message vectors in a special way. Suppose the bits of each block of the original message are spread over the whole chain of message vectors. If the cryptanalysis fails at some point of the chain then only fragments of the message have been disclosed. Which kind of partial disclosure can be tolerated best is a more general information theoretic question and lies outside the scope of this study. It must be noted also, that the whole idea of chaining is by no means restricted to knapsack cryptosystems.

3 Small injective knapsacks

The first table summarizes our results concerning the search for the densest injective knapsacks of a given small size.

n	A = the densest found knapsacks of size n	$\frac{\max(A)}{2^n-1}$	$\rho(A)$
1	$\{1\}$	1	—
2	$\{1,2\}$	1	2
3	$\{1,2,4\}$ $\{2,3,4\}$	1	1.5
4	$\{3,5,6,7\}$	0.875	1.4248
5	$\{3,6,11,12,13\}$ $\{6,9,11,12,13\}$	0.8125	1.3512
6	$\{11,17,20,22,23,24\}$	0.75	1.3086
7	$\{20,31,37,40,42,43,44\}$	0.6875	1.2822
8	$\{39,59,70,77,78,79,81,84\}$	0.6563	1.2515
9	$\{76,115,135,149,152,154,155,156,163\}$	0.6367	1.2247
10	$\{148,227,266,286,297,304,305,306,308,311\}$	0.6074	1.2076

For each $n = 1, \ldots, 10$ the table gives the densest injective knapsack of size n, that we found. The third column relates its largest element to 2^{n-1}, which represents the densest superincreasing knapsack of size n. The density is computed in the last column. For the sizes $n = 1, \ldots, 7$ the search was exhaustive and thus the results are exact. For these values all the knapsacks of the highest density are given.

Injective knapsacks of size 6

The next table deals with injective knapsacks $\{a_1, \ldots, a_6\}$ that have a density at least $\rho_2(6)$. The number of injective knapsacks with a_6 as their largest element is given in the second column. The next column '$\frac{inj}{all}$' relates this number to the number of all knapsacks of size 6 with a_6 as their largest element. The unit is 10^{-3}. The fourth column gives the maximal difference between the two largest elements. The column 'ave' is the average size of the elements. The last column s_{min} contains the smallest of the sums of the knapsacks.

a_6	inj	$\frac{inj}{all}$ /10^{-3}	ave	$\max_{a_6 - a_5}$	s_{min}
24	1	0.03	19.50	1	117
25	11	0.26	18.48	3	93
26	41	0.77	18.52	3	91
27	52	0.79	19.62	5	94
28	264	3.28	19.14	8	87
29	246	2.51	20.19	7	90
30	530	4.48	20.66	11	93
31	715	5.05	21.16	11	93
32	2275	13.39	20.30	16	63

There is a clear change in the data when $a_6 = 32$. In this row there are also many other knapsacks than the superincreasing one, that have $a_6 - a_1 = 16$ and a sum smaller than 90. Out of the 2275 knapsacks 1024 can be constructed starting from $A_1 = \{1\}$ and using Lemma 2.2. Similarly we can construct 26 knapsacks of the row $a_6 = 26$ from the two densest 5-element knapsacks, 13 from each. The row $a_6 = 28$ inherits 98 knapsacks from the densest 4-element knapsack.

The largest sum for the knapsacks in the i'th row is $117 + 6(i - 1)$. Only one knapsack in each row has this sum, and that knapsack equals $A + i - 1$ where A is the only knapsack with $a_6 = 24$. For this A also all the other shifted knapsacks $A + d$ are injective, when d is positive.

The maximal a_1 in each row is $11 + i - 1$, but in most rows there are also a couple of other cases than the shifted knapsacks. The minimal a_1 is 1 for other rows except the first two.

The 11 knapsacks with $a_6 = 25$ are

$$\begin{array}{llllll}
\{11, & 18, & 19, & 20, & 22, & 25\} \\
\{11, & 17, & 21, & 22, & 23, & ''\} \\
\{\ 3, & \ 6, & 12, & 23, & 24, & ''\} \\
\{\ 6, & \ 9, & '', & '', & '', & ''\} \\
\{\ 6, & 12, & 15, & '', & '', & ''\} \\
\{\ 3, & '', & '', & '', & '', & ''\} \\
\{\ 9, & '', & '', & '', & '', & ''\} \\
\{12, & 15, & '', & '', & '', & ''\} \\
\{\ 6, & 12, & 21, & '', & '', & ''\} \\
\{11, & 18, & '', & '', & '', & ''\} \\
\{12, & '', & '', & '', & '', & ''\}
\end{array}$$

Here '' indicates that the element is the same as above. It can be seen that these knapsacks are really dense in the sense that half of the elements are packed together at the top. The two first and two last knapsacks have the special property that all their positive shifts are injective.

It seems that when we are dealing with integers, there are always special cases to expect. Now we could say that there are relatively few knapsacks in the row $a_6 = 27$ and many in the row $a_6 = 28$. As a consequence the other data do not show any clear tendency, either. To investigate the distribution of elements we should go for larger knapsack sizes, but already the size 7 means quite an effort. We will discuss a sample of knapsacks of size 10, but we first need to describe how we have been looking for them.

Search for knapsacks

In order to find an injective knapsack of size n with a as the largest element we used the following algorithm.

1. Choose randomly an element a' satisfying $a - 10 \leq a' < a$. Set

$$A := \{a', a\}, \qquad S := \{0, a', a, a' + a\}, \qquad D := \{a - a', a'\}.$$

2. Choose randomly an element $x \in \{1, \ldots, a-1\} \setminus D$.

3. If $|S \cup (x + S)| = 2\,|S|$, set

$$A := A \cup \{x\}, \qquad S := S \cup (x + S),$$

$$D := \Big[D \cup \{x\} \cup (D + x) \cup (D - x)\Big] \quad \cap \{1, \ldots, a-1\}.$$

   otherwise,

   If $|A| < n - 1$, restart from step 1.

   If $|A| = n - 1$, try again with step 2, at most a couple of times (we used a limit of 3).

4. If $|A| < n$ go to 2.

At every stage A is an injective knapsack, S contains its subset sums, and D is a set of such integers that cannot be introduced into A without loosing injectivity.

The claim concerning D is clear at step 1. To see why no elements from $[(D+x) \cup (D-x)] \cap \{1,\ldots,a-1\}$ can be introduced into $A \cup \{x\}$, consider a $d \in D$. It is in D because there are such subset sums s_1 and s_2 of A that $s_1 + d = s_2$. Now $s_1 + x$ and $s_2 + x$ are subset sums of the knapsack $A \cup \{x\}$ and the answer can be seen from the equations $s_1 + x + d = s_2$ and $s_1 + d = s_2 + x$. If we do not restrict the set D to the range $1,\ldots,a-1$, we get the set D_A of Notation 7.1. See also Theorem 7.2.

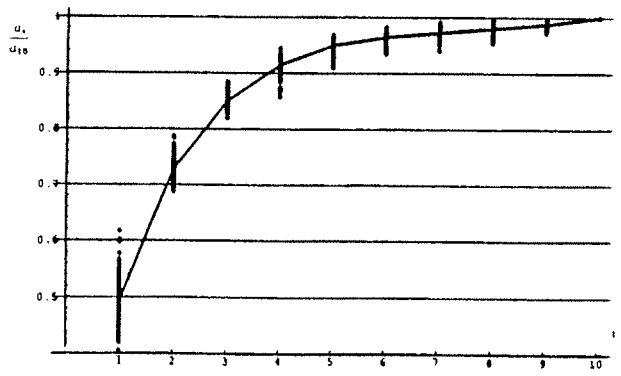
A sample of knapsacks of size 10

We used the above algorithm for $n = 10$ starting from $a = 500$ and decrementing a one by one. We proceeded until $a = 315$ where it took over 19500 rounds (step 1) to find a knapsack. The next 30000 rounds did not produce a knapsack with $a = 314$, but we found two of them, two with $a = 312$ and one with $a = 311$ by shifting down the previous knapsacks (for $a = 311$ the shift was -9).

Our sample of knapsacks with $a = 315,\ldots,500$ is not completely representative, because in each knapsack at least one element must be in the range $a-10,\ldots,a-1$. The reason to choose such a strategy was that otherwise the algorithm was very slow.

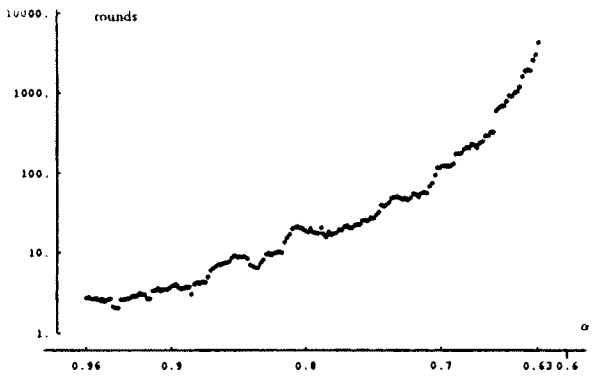
When $a = 491,\ldots,500$ the present algorithm finds a knapsack in 3.5 rounds on average. If we require that no elements are in the range $a-10,\ldots,a-1$, we need some 600 rounds to find a knapsack for these a. This means that such knapsacks are very rare among all knapsacks already at such large values of a. Below $a = 412$ we have not yet found any knapsacks satisfying the alternative requirement. When $a = 421,\ldots,430$, it seems to require at least 25 000 rounds to find one such knapsack.

On the above grounds we believe that our sample represents average behaviour of dense knapsacks of size 10. The whole sample of 186 knapsacks is depicted in the first figure below. For every knapsack a dot is plotted for $\frac{a_i}{a_{10}}$ against $i = 1,\ldots,10$. For each i we computed the average and joined them with a line in the figure.



The average size of elements in these knapsacks is clearly very large. It turned out that the average is between $0.864\,a_{10}$ and $0.913\,a_{10}$ in every case.

For each a we recorded the amount of rounds needed to find the knapsack. We then computed the average of 15 of these numbers around each $a = 322, \ldots, 493$ and plotted them against $\alpha = \frac{a}{2^9}$. By using such an abscissa we want to suggest that the figure tells also something general about the difficulty of finding injective knapsack and hence also about the number of them, when the maximal element is $\alpha 2^{n-1}$.



Conjectures

The above empirical results and some others together with daring intuition lead us to following conjectures concerning dense injective knapsacks. Denote by $\mathrm{Max}(n)$ the largest element in the densest injective knapsacks of size n.

- $\mathrm{Max}(n+1) < 2\,\mathrm{Max}(n)$. Notice that by Lemma 2.2 we have '$\leq$'. Actually, we believe that $\frac{\mathrm{Max}(n)}{2^{n-1}} \to 0$ as $n \to \infty$.
- There are injective knapsacks A of size n with $\max(A) = \mathrm{Max}(n) + d$, for all $d = 1, 2, \ldots$, and the number of these knapsacks grows exponentially in d.
- In a relatively dense knapsack A of size n
  - $\frac{\max(A)}{\min(A)}$ is not very large.
  - several of the largest elements are relatively close to one another. For example out of the 330 knapsacks of size 10 with $a_{10} \leq 500$ that we know, 320 satisfy $\frac{a_9}{a_{10}} \geq 0.9$.
  - $s_A > \frac{n}{2}\max(A)$.
  - usually $\frac{2^n}{s_A} \leq 0.6$. For example 326 of our dense knapsacks of size 10 satisfy $0.26 < \frac{2^{10}}{s_A} < 0.40$. This ratio gives the probability that an arbitrary integer between 0 and s_A is a subset sum of A.

4 About the trapdoor

4.1 Why strong modular multiplication?

The most common trapdoor, the inverse of strong modular multiplication, enables easy deciphering by the following principle:

> The sum instance of a difficult knapsack is transformed back to an equivalent sum instance of an easy knapsack. (3)

As the cryptanalytic attacks show, this property is not necessary for decryption. Also the intended decryption algorithm may have a different principle. An example of a trapdoor technique for which the above property does not hold, is a sequence of strong modular multiplications applied iteratively to an initial knapsack, which only needs to be injective. Decryption of a subset sum of the final knapsack is carried out by setting up a system of linear modular equations according to the intermediate knapsacks and the multipliers and moduli. (The system is due to Shamir. Details can be found in [30, p. 216].) Also the Merkle-Hellman multiplicative trapdoor knapsack is different because the trapdoor does not lead to an easy *additive* knapsack (see Section 6.1).

We will now assume that the property (3) holds for the trapdoor technique and we will investigate what kind of further conditions lead to the strong modular multiplication.

Although the trapdoor means a way from the difficult knapsack to an easy one, it will be more convenient to use the term **trapdoor transformation** for the transformation that converts the easy knapsack to an apparently difficult one.

Suppose that the easy knapsack is $A = \{a_1, \ldots, a_n\}$ and that the trapdoor transformation does not change the number of elements. Then the difficult knapsack will be

$$B = \{f(a_1), \ldots, f(a_n)\}$$

for some integer-valued function f that is defined over some subset of the natural numbers that contains at least the set A. Clearly f restricted to the elements of A must be one-to-one.

According to (3) such an inverting operation g is required that

$$g(\sum_{i \in S} f(a_i)) = \sum_{i \in S} a_i \tag{4}$$

for all those subsets S of $\{1, \ldots, n\}$ that correspond to messages.

Suppose that $C = \{c_1, \ldots, c_n\}$ is an arbitrary injective knapsack and define $f(a_i) = c_i$. The function g can then be defined by equation (4). This kind of definition is not of much use in practice. The following requirements for f are natural:

- The rule how to compute $f(a)$ depends only on a few parameters that do not depend on a.

- Some of the parameters may depend on the whole of A, but there must also be space for some randomness.
- The rule can be applied to any integer, at least in the range $1, \ldots, \max(A)$.

The last requirement may be the most disputable of these.

It must be added that with a separate permutation we can always take care of all desired changes of order among the elements of knapsack B when it is published as a public key.

Symmetry. We assume now that the pair of transformations f and g is symmetric in the sense of equation (4). That is, f takes subset sums of the easy knapsack to corresponding subset sums of the difficult knapsack:

$$f(\sum_{i \in S} a_i) = \sum_{i \in S} f(a_i).$$

Let us use this for subsets with two elements and assume even more: the rule $f(a+b) = f(a) + f(b)$ holds for any pair $a, b \in \{1, \ldots, \max(A)\}$, $a \neq b$. Applying the rule successively we get for any a, $3 \leq a \leq \max(A)$, that

$$f(a) = f(a-1) + f(1) = f(a-2) + f(1) + f(1) = \cdots$$

$$= f(2) + (a-2)f(1) = f(1)a + f(2) - 2f(1).$$

Suppose now that $\max(A) \geq 7$ and denote $w = f(1)$ and $d = f(2) - 2f(1)$. We have $w \cdot 7 + d = f(7) = f(3) + f(4) = w \cdot 3 + d + w \cdot 4 + d$, which gives $d = 0$. We also have $f(2) = 2f(1) = w \cdot 2$.

Our assumptions have led to a multiplicative rule for f, that is, $f(a) = wa$. The required one-to-oneness of f implies that $w \neq 0$. The transformation $g(c) = w^{-1}c$ will then satisfy (4).

Fixed number of addends. Suppose that the subset sums must always contain the same amount, say h addends. If $h > 2$ we have no reason to assume $f(a+b) = f(a) + f(b)$ which finally led to the multiplicative rule for f. If there are other reasons why we have $f(a) = wa + d$, then d need not be 0. The equation (4) is satisfied by $g(s) = w^{-1}(s - hd)$. If $d \neq 0$ this g is not the inverse of f.

Modularity. The above derivations did not assume that the integers would not be modular with respect to some modulus $m > \max(A)$. This would mean that instead of every '=' we would have '$\equiv \pmod{m}$'. If m is not a prime then the requirement $w \not\equiv 0 \pmod{m}$ is not enough for the inverse w^{-1} to exist. Instead we must have $\gcd(w, m) = 1$.

Multiplication is useless as a trapdoor technique without a modulus, because computation of a few greatest common divisors would reveal the multiplier.

Although the sum instances of B are computed without reducing them, they must be distinct modulo m. Otherwise unique decryption would be impossible. This is because the result of the transformation g is modular but

it must also be a valid (non-modular) sum instance of A. This requires that every sum instance s of A equals $s \pmod m$. Sufficient and necessary for this is that m is greater than the sum of A. Thus we are led to strong modular multiplication.

4.2 Sources of insecurity

Development of density

Assume that we have a knapsack A, which is not very small. Denote $a = \max(A)$, and choose an $m > a$ and an integer w, relatively prime to m. Denote $B = wA \pmod m$.

When $1 \leq x \leq m-1$, the modular product $wx \pmod m$ obtains all the values from the range $1, \ldots, m-1$. If we choose A randomly, then the elements of B are uniformly distributed over the range $1, \ldots, m-1$. Such a distribution means that for any given B we expect the elements to be approximately evenly dispersed between 1 and $m-1$.

Assume now that w is so large that changing the value of x with a few units changes the integer k in $wx \pmod m = wx - km$. For example $w > m/2$ will satisfy this. This assumption means that the relative position of x between 1 and a has practically nothing to do with the position of $wx \pmod m$ between 1 and $m-1$.

As Lemma 2.2 shows, there is a vast choice for the knapsack A, even if it has to be injective, relatively dense and easy to decipher. Since the integers m and w are chosen independently of the construction system of A, we expect that the elements of B are distributed as if A were a random knapsack. This means that we expect to find the same number of elements of B in every subrange of $1, \ldots, m-1$ of fixed length. Especially we expect that some elements of B are also near m. Under a truly uniform distribution the probability, that all elements of B are below αm, is approximately $\alpha^{|B|}$ $(0 \leq \alpha \leq 1)$.

We summarize this discussion as a lemma.

Lemma 4.1 Assume A is a knapsack, not very small, $m > \max(A)$ and w is a relatively large integer that satisfies $w < m$ and $\gcd(w, m) = 1$. If $B = wA \pmod m$ then the elements of B are likely to be evenly dispersed in the range $1, \ldots, m-1$. Especially $\max(B)$ is likely to be very near m, in proportion. □

If we want to construct a reasonably dense knapsack A of practical size n, then there appears to be no reason to choose the largest element a_n very much larger than the second largest element a_{n-1}. The assumption $\frac{a_n}{a_{n-1}} \leq 3$ leaves enough alternatives for a_n also in superincreasing construction. All the examples of dense knapsacks in Chapter 3 satisfy this assumption with 3 replaced by 2. The assumption implies that $s_A \geq \frac{4}{3}\max(A)$.

Suppose m is a strong modulus for A. Then the above discussion implies that modular multiplication of A is very likely to result in a knapsack B that satisfies $\max(B) \geq \frac{3}{4}m > \max(A)$.

Actually we only used the two largest elements of A to bound the modulus, and thus we may conclude the following.

Lemma 4.2 The density of the knapsack resulting from modular multiplication is likely to be lower than the original density, if the modulus exceeds the sum of the two largest elements of the original knapsack. □

Remark. The following conjecture seems plausible.

Assume we have two injective knapsacks A_1 and A_2 of the same size n. For $i = 1, 2$ transform A_i with strong modular multiplication to a knapsack B_i. If A_1 is superincreasing, $\rho(A_1) \approx \rho_2(n) < \rho(A_2)$, then we are likely to have $\rho(B_2) < \rho(B_1) < \rho(A_1)$.

Inherent dangers in modular multiplication?

Assume that we know a knapsack $B = \{b_1, \ldots, b_n\}$ and the fact that it has been obtained by modular multiplication from an unknown knapsack A. Let $B = wA \pmod m$ and u be the inverse of w modulo m. Then there are such integers k_i that $a_i = ub_i - k_i m$ for $i = 1, \ldots, n$, and hence

$$\frac{u}{m} - \frac{k_i}{b_i} = \frac{a_i}{b_i m}. \tag{5}$$

If m is a strong modulus for A, it is very large in comparison to almost all of the a_i. The values of b_i typically are of the same order of magnitude as m. Hence the quotient $\frac{a_i}{b_i m}$ is very small for most values of i, and is likely to be such for all the i. Consequently almost all of the quotients $\frac{k_i}{b_i}$ are close to one another. This is a very exceptional situation, and in case where A is superincreasing it was just this that made cryptanalysis possible. According to [5] also all the other knapsack cryptosystems that are based on strong modular multiplication(s) can be broken with this approach, using the techniques of *simultaneous diophantine approximation*.

In the next chapter we introduce weak forms of injectivity where non-strong modular multiplication can be used as a trapdoor transformation. When the modulus is somewhat smaller than the sum of the knapsack it may be more difficult to exploit the equation (5), especially if already the easy knapsack A is only injective in some weak sense. It lies however outside the scope of the present study to answer the question, whether there is something inherently insecure in modular multiplication as a trapdoor transformation of all kinds of easy knapsacks. Notice that according to Lemma 4.2 modular multiplication is almost always likely to make the knapsack somewhat more vulnerable to the low density attacks.

5 What if the knapsack is not injective ?

We introduce some simple modifications of the concept of injectivity in Section 5.1. One of them is stronger than injectivity, but the others are weaker. They still allow unique decipherment in situations that are restricted accordingly. A simple example of such controllable non-injectivity is given in Section 5.2. In Section 5.3 we discuss what can be done if the knapsack is non-

injective in a way that cannot be controlled in this manner. Sections 5.4 and 5.5 deal with two special cases.

5.1 Weak forms of injectivity

From now on the term *injectivity* will have both the concrete meaning of some knapsack being injective and an abstract meaning, which may have different concretizations depending on the attributes attached to it. The abstract meaning makes it possible to discuss *what kind* of injectivity a knapsack possesses, not only whether the knapsack is injective or not.

Definition 5.1 Assume A is a knapsack and $1 \leq h \leq |A|$.

(i) If for all fixed values of k, $1 \leq k \leq |A|$, the subset sums of A that have k addends are distinct, then A is said to be **fix-injective**.

(ii) If the subset sums of A that have at most h addends are distinct, then A is ***h*-injective**.

(iii) If for all fixed values of k, $1 \leq k \leq h$, the subset sums of A that have k addends are distinct, then A is **fix-*h*-injective**.

(iv) If the subset sums of A are distinct modulo an integer $m > 1$, then A is **injective modulo *m***.

(v) If the subset sums of A satisfy the conditions of (i), (ii) or (iii) modulo an integer $m > 1$, then A is **X-injective modulo *m***, where X is the corresponding attribute of injectivity from these definitions.

Definition 5.2 If m is an integer greater than the sum of the h largest elements of a knapsack A then m is an ***h*-modulus** for A. If m is an h-modulus for A and w is relatively prime to m then the knapsack $wA \pmod m$ is obtained by ***h*-modular multiplication** from A.

Remark. If $h = |A|$, then h-injectivity is equivalent to injectivity as defined in Section 1.4 and an h-modulus is the same as a strong modulus. Hence dealing with (fix-)h-injectivity and h-moduli in the sequel will cover also the case of (fix-)injectivity and strong moduli.

All knapsacks are 1-injective.

The following theorem depicts obvious relationships between the new concepts.

Theorem 5.3 Assume A is a knapsack, $1 \leq h'' \leq h' \leq h \leq |A|$, $m > 1$ and m' is an h'-modulus for A. The following implications hold between properties of A. From the alternatives in (i) and (iii) take either the upper or the lower ones at the same time. The statement (ii) is valid also with 'modulo m' added to each property.

(i) $\left.\begin{matrix} h \\ \text{fix-}h \end{matrix}\right\}$-injective modulo m $\Rightarrow$ $\left.\begin{matrix} h \\ \text{fix-}h \end{matrix}\right\}$-injective

(ii) h-injective $\Rightarrow$ h'-injective $\Rightarrow$ fix-h'-injective $\Rightarrow$ fix-h''-injective

(iii) $\left.\begin{matrix} h \\ \text{fix-}h \end{matrix}\right\}$-injective $\Rightarrow$ $\left.\begin{matrix} h'' \\ \text{fix-}h'' \end{matrix}\right\}$-injective modulo m'

□

The lemma expresses simply that a characterization of injectivity of a knapsack remains valid under the following changes.

- Removal of modularity.
- Addition of the attribute 'fix'.
- Decrease of the value of an attribute 'h'.
- Introduction of modularity with the attribute 'h' adjusted to a value satisfied by the modulus.

An example of a useful knapsack that is (only) fix-h-injective modulo an integer is the Chor-Rivest system that will be discussed in Chapter 10. There will be one further kind of injectivity mentioned in Theorem 9.7. It strengthens the concept of fix-injectivity by requiring that whenever two subsets of the knapsack correspond to the same sum, then the numbers of elements in these subsets are not congruent modulo an integer.

The first one of the following two simple lemmas can be deduced by considering the subset sums $s_A - s$ instead of s. The second lemma expresses the fact that if some subset sums are to be distinct modulo m, then their total number cannot exceed m.

Lemma 5.4 If A is an h-injective knapsack then all the subset sums of A with at least $|A| - h$ addends are distinct.

If A is a fix-h-injective knapsack then for all fixed values of k, $|A| - h \leq k \leq |A|$, the subset sums of A that have k addends are distinct. □

Lemma 5.5 If a knapsack of size n is h-injective modulo m then $m \geq \sum_{i=0}^{h} \binom{n}{i}$. Especially if the knapsack is injective, then $m \geq 2^n$. □

The cryptographic usage of knapsacks that possess some weak form of injectivity is obvious. We need to restrict the messages to such bit sequences that contain at most h ones, or a fixed number of ones, or the number of ones must be both fixed and at most h.

In Chapter 4 we discussed trapdoor transformations of knapsacks to other knapsacks and were confined to multiplication and addition, possibly with

respect to a modulus. The next theorem and its corollary present these operations from the viewpoint of injectivity. The theorem is a direct consequence of the following lemma.

Lemma 5.6 Assume that A is a knapsack, $1 \leq h \leq |A|$, $m > 1$, $\gcd(w, m) = 1$ and d is any integer.

Let s_1 and s_2 be two subset sums of A with at most h addends in each. Assume either that

(i) m is an h-modulus for A and $s_1 \neq s_2$, or

(ii) $s_1 \not\equiv s_2 \pmod{m}$.

Then the subset sums of $wA \pmod{m}$ corresponding to s_1 and s_2 are distinct, also modulo m.

If the number of addends in s_1 is the same as in s_2, then the subset sums of $wA + d \pmod{m}$ corresponding to s_1 and s_2 are distinct, also modulo m.

Proof. The assumption (i) implies (ii). Thus we can assume only (ii). Let $s_1 = \sum a_i$ and $s_2 = \sum a_j$. Since w is relatively prime to m, the condition $\sum a_i \not\equiv \sum a_j \pmod{m}$ implies

$$\sum wa_i \equiv w \sum a_i \not\equiv w \sum a_j \equiv \sum wa_j \pmod{m}.$$

If the number of addends in s_1 and s_2 is k we also have

$$\sum(wa_i + d) \equiv w \sum a_i + kd \not\equiv w \sum a_j + kd \equiv \sum(wa_j + d) \pmod{m}.$$

We see that the claimed subset sums are distinct modulo m and hence also distinct as integers. □

Theorem 5.7 Let A be a knapsack, $1 \leq h \leq |A|$ and $m > 1$. Assume A is X-h-injective modulo m, where X stands either for nothing or 'fix'.

If $\gcd(w, m) = 1$ then also $wA \pmod{m}$ is X-h-injective modulo m. If further d is any integer, then $wA + d \pmod{m}$ is fix-h-injective modulo m. □

Taking into account Theorem 5.3 it is easy to see that Theorem 5.7 implies the following corollary. For the first statement it suffices to consider a very large modulus and discard it in the end. A more simple case of the second statement would have $h'' = h' = h$. Now these three integers appear in accordance with Theorem 5.3.

Corollary 5.8 Let A be a knapsack, $1 \leq h \leq |A|$. Assume A is X-h-injective, where X stands either for nothing or 'fix'.

(i) If $w > 0$ then also wA is X-h-injective. If further d and d' are such integers that $w \min(A) + d > 0$ and $d' - w \max(A) > 0$ then $wA + d$ and $d' - wA$ are fix-h-injective knapsacks.

(ii) Suppose $1 \leq h' \leq |A|$. Let m be an h'-modulus for A and $\gcd(w, m) = 1$. Denote $h'' = \min(h, h')$. Then $wA \pmod m$ is X-h''-injective modulo m. If d is any integer then $wA + d \pmod m$ is a fix-h''-injective knapsack modulo m. □

As a summary of these results note that multiplication preserves all kinds of injectivities, but if it is done with respect to an h-modulus then h appears as an attribute of the injectivity — if there was not already a smaller number. Secondly, if a constant is added then the attribute 'fix' must be given to the injectivity if it was not there already.

Remark. Theorem 5.7 does not follow from the claim (ii) of the corollary, because there are knapsacks that are h- or fix-h-injective modulo an m which is not an h-modulus.

For example let $A = \{2, 3, 4\}$. Write the subset sums of A as a sequence $S = (0;\ 2, 3, 4;\ 5, 6, 7;\ 9)$ where the semicolons distinguish sums with different number of addends. Now $S \pmod 8 = (0;\ 2, 3, 4;\ 5, 6, 7;\ 1)$. We see that A is 3-injective modulo 8 (i.e. injective modulo 8), although 8 is only a 2-modulus for A.

On the other hand $S \pmod 3 = (0;\ 2, 0, 1;\ 2, 0, 1;\ 0)$ which shows that A is fix-injective modulo 3, although 3 is not even a 1-modulus for A.

Recall that $A + d \pmod m$ was defined to consist of positive remainders. Fix-injectivity is a reasonable concept also for such knapsacks that are allowed to contain a zero. For example the subset sums of $B = \{0, 2, 3, 4\}$ are $(0;\ 0, 2, 3, 4;\ 2, 3, 4, 5, 6, 7;\ 5, 6, 7, 9;\ 9)$ and we see that B would be fix-injective under such an extended definition. It is clear that the second statement of Theorem 5.7 would remain valid also in case of *non-negative* remainders.

5.2 Weakening the superincrease

The following definition and theorem express the obvious way to modify superincrease to produce h-injective knapsacks. The idea is simply to choose new elements that exceed the sum of h previous elements.

Definition 5.9 Assume $h \geq 1$ and $n \geq h+2$. A knapsack $A = \{a_1, a_2, \ldots, a_n\}$ is ***h*-increasing** if $\{a_1, \ldots, a_{h+1}\}$ is a superincreasing knapsack and the elements $a_{h+2}, \ldots, a_n$ satisfy

$$a_i > \sum_{j=i-h}^{i-1} a_j. \tag{6}$$

Theorem 5.10 An h-increasing knapsack is h-injective and all subset sums with at most h addends are easy to decipher. □

Remark. When transforming an h-increasing knapsack to a public key by modular multiplication, it of course suffices to use an h-modulus.

Example. The densest possible 4-increasing knapsack of size 8 is

$$A = \{1, 2, 4, 8, 16, 31, 60, 116\}.$$

This is 4-injective by the theorem but not 5-injective, because 1+2+4+8+16 = 31. The sum of the 4 largest elements of A is 223. Thus 225 is a 4-modulus. Since $\gcd(151, 225) = 1$ we know that

$$151A \pmod{223} = \{151, 77, 154, 83, 166, 181, 60, 191\}$$

is 4-injective. In this case the knapsack is also injective. □

It is especially easy to use an h-injective knapsack in cryptography when $h = \lceil \frac{n}{2} \rceil$, as we can see with the aid of the following lemma.

Lemma 5.11 Let A be a knapsack and denote $n = |A|$. Assume that A is $\lceil \frac{n}{2} \rceil$-injective. Then

(i) All subset sums of A that have at least $\lfloor \frac{n}{2} \rfloor$ addends are distinct.

(ii) An integer can have at most two different representations as a subset sum of A.

(iii) If two different subset sums of A coincide then one has fewer addends than $\lfloor \frac{n}{2} \rfloor$ and the other has more addends than $\lceil \frac{n}{2} \rceil$.

(iv) A is fix-injective.

Proof. Lemma 5.4 directly implies (i), because $\lfloor \frac{n}{2} \rfloor + \lceil \frac{n}{2} \rceil = n$. Since $\lfloor \frac{n}{2} \rfloor \leq \lceil \frac{n}{2} \rceil$, (i) implies (ii) and (iii). Finally (iii) implies (iv). □

Cryptographic usage of an $\lceil \frac{n}{2} \rceil$-injective knapsack, which is not injective, requires one extra bit of information in addition to each subset sum. Namely one must know whether there are more than $\lceil \frac{n}{2} \rceil$ addends in the subset sum s or not. If there are more, then one deciphers $s_A - s$ (where there are at most $\lceil \frac{n}{2} \rceil$ addends) and complements the resulting bits. Of course the sender may already do this complementation, but the same extra bit of information is still needed.

A large $\lceil \frac{n}{2} \rceil$-increasing knapsack does not differ very much from a super-increasing knapsack, at least if density is concerned. For example the densest possible 50-increasing knapsack of size 100 has the largest element $6.338 \cdot 10^{26}$. This differs from 2^{99} only by $1.38 \cdot 10^{16}$.

5.3 More general non-injectivity

We saw in Sections 5.1 and 5.2 that the knapsack need not be injective if the cryptosystem always produces the same and/or sufficiently small number of addends in the subset sums. Such non-injective knapsacks can be denser than

the injective ones, but there is still the requirement of injectivity at least for some fixed numbers of addends. What happens if even this does not hold?

Suppose there is a checksum that is computed from the message and sent along with the enciphered message. If the function used for the checksum is injective then the validity of decipherment may always be checked. If it is a one-way function then the security of the cryptosystem is not affected. Although there are no one-way functions in the sense that their inversion would be impossible, there are functions that are very hard to invert. For example general knapsacks are considered suitable for one-way functions in [2, p. 17]. We cannot be sure of the injectivity of an arbitrary knapsack, but the probability is very small, that we first have a wrong decipherment of the message and then a wrong confirmation of it from the checksum knapsack.

We need not generate and publish another knapsack, since we already have our trapdoor knapsack. With some prescribed transformation the sender can change it to another knapsack and generate the checksum. The transformation can be a common one or it may depend on a few small parameters that are published together with the knapsack. An easy transformation that seems likely to change any trapdoor knapsack to a genuinely difficult knapsack is the reversing of the order of digits in the knapsack elements. If our public key is not very dense we may request that a couple of digits are dropped from each element.

If we have bad luck, the knapsack checksum is easier to cryptanalyse than the original cryptotext. Since using the general knapsack checksum already admits a small chance of wrong decipherment, we may change the system a little to avoid at least the worst luck. Assume that the transformation of our enciphering knapsack depends on the message and can produce a very large number of different checksum knapsacks. It seems that the cryptanalyst of the checksum must try with a great part of these before he finds a solution that satisfies also the subset sum of the enciphering knapsack. Direct cryptanalysis of the latter is likely to be easier.

A simple way to modify the checksum is to add the binary value of the message to it, either as such or after some transformation.

Notice that usage of a fixed checksum knapsack together with an enciphering knapsack amounts to a two-dimensional knapsack, that is, a knapsack where the elements are pairs of integers.

Example. In a system with a checksum we need not tell the number of addends even if that is the requirement for unique decipherment. This can be seen as follows. The number of 1's in random binary messages is distributed almost normally. If the message has length 200 bits then the standard deviation is about 7 and with 95% probability the number of 1's in the message is 100 ± 14. Thus, with this probability we need do at most 29 trials with the number of addends before we find one that leads to a solution. □

It may happen that no uncertainty is allowed in cryptographic usage of non-injective knapsacks. This can be achieved with the following algorithm (or a protocol) that may request new messages from the sender.

1. Given a subset sum, find all possible decipherments.
If there is only one, it is the correct decipherment.
Otherwise:

If also a checksum was given for the subset sum,

2. compute the checksum for all the decipherments.
If only one decipherment is confirmed, then that is correct.
Otherwise:

3. Request either

an entirely new message to substitute the old one and continue from 1. (This may be reasonable especially if the redundancy of the messages allows to express the same things in different ways.)

or

a new checksum for a given permutation of the original message bits. This can be produced either with the enciphering key or the checksum key. Continue from 2.

If there was no checksum,

4. request one (with some prescribed technique) and go to 2.

It is also possible to ask only for some bits of the checksum in phase 4, and iterate between phases 2 and 4 until all uncertainty has been resolved. If this does not eventually succeed then an analog of phase 3 must be performed.

To be able to use this algorithm we need special kind of non-injectivity. The next definition suggests a name for it. Notice that unlike the concepts of Section 5.1 the new property cannot so naturally be classified as a "weak form of injectivity".

Definition 5.12 We call a knapsack **temperate** if it is easy to **decipher completely**, that is, to find all possible decipherments of any given subset sum of it.

Remark. The easiness in the definition refers to existence of a polynomial time algorithm which also tells explicitly when all the decipherments have been found. Actually, in our examples the question of temperateness is fairly trivial in principle. We would prefer however, that we can bound *in advance* and to a *practical level* the number of trials needed to find the decipherments.

A more involved analysis may motivate the notion of *h-temperate* knapsacks in analogy with h-injectivity, but we will not need it. The notion of fix-temperate (or fix-h-temperate) knapsacks, however, is not very interesting theoretically. If we have a reasonable bound for the number of decipherments for each number (or at most h) of addends, then summing these bounds from 1 to n (or h) still gives a reasonable bound. For a fix-injective knapsack this bound is n, but in more general cases the bound may be impracticable. Because of this and also for simplicity we will often assume that the number of

addends is known.

We give here three examples of knapsack constructions, where the temperateness appears at different stages. The first example deals with a trapdoor transformation. The second method can be applied equally well to an initial easy knapsack as to a transformed knapsack. In the third case a non-injective easy knapsack is constructed. Sections 5.4 and 5.5 discuss two further examples.

Different shift for different parts of the knapsack

Apply $f(a) = wa + d$ to one half of the elements of the knapsack and $f(a) = wa + d'$ to the rest (use a modulus if $w \neq 1$). Suppose that the number of addends in a subset sum s is known to be h. Then the extra effort caused by this transformation in deciphering s amounts to guessing such an i, $0 \leq i \leq h$ that

$$s - id - (h - i)d'$$

is a subset sum for the knapsack wA.

A similar analysis as in the above example will give bounds for the probability that at least one decipherment is found, if only some values of i near $h/2$ are tried.

Random elements

Insertion of a few new elements in the superincreasing or other injective structures increases density and possibly perturbs the construction system of the knapsack enough so that specialized attacks do not succeed so easily. If there are k exceptional elements, then generally all the 2^k combinations of these must be checked to find all the decipherments of a subset sum. To find one decipherment may be considerably easier, as will be seen in the special case of the next section. In any case, if the messages are uniformly distributed then the average number of trials needed for the first decipherment is at most 2^{k-1}.

"Too weak" increase

Suppose $h \geq 3$ and we have an $(h-1)$-increasing knapsack and a subset sum where the number of addends is at most h.

Suppose that we have been able to decipher the sum without any ambiguity down to an index i and the remaining subset sum is s. That is, we know that s is a subset sum of $\{a_1, \ldots, a_i\}$. If $i \leq h$ then this knapsack is superincreasing and the rest will be clear. Assume that $i > h$.

The matter with a_i is uncertain only if $s \geq a_i$ and s is still the original sum (i.e. there may still be h addends). Then we must try with and without a_i, but after that there is no more ambiguity. To see this denote $A' = \{a_1, \ldots, a_{i-1}\}$.

1. Suppose we try with a_i. This means that we try to decipher $s' = s - a_i$ with the knapsack A'. We know that if s' is a subset sum of A' then

there are at most $h-1$ addends and the deciphering process will proceed without any ambiguity, because A' is $(h-1)$-injective. Hence we may always subtract such an element that exceeds the remaining sum. If we do not get to 0 in the end, then $s - a_i$ was not a subset sum of A'.

2. Trying without a_i means an attempt to decipher s with the knapsack A'. We show that in this case we must proceed by subtracting a_{i-1}. After this the situation is analogous to 1.

   Suppose first that $i = h+1$. Then

$$\begin{aligned} s \geq a_i &> a_{i-1} + a_{i-2} + \ldots + a_2 \\ &> a_{i-2} + \ldots + a_2 + a_1 \end{aligned}$$

   and we see that s cannot be a subset sum of $A' \setminus \{a_{i-1}\}$.

   Suppose then that $i > h+1$. Since $h \geq 3$ we have $a_{i-1} > a_{i-h} + a_{i-h-1}$ and thus

$$\begin{aligned} s \geq a_i &> a_{i-1} + a_{i-2} + \ldots + a_{i-h+1} \\ &> a_{i-2} + \ldots + a_{i-h+1} + a_{i-h} + a_{i-h-1}. \end{aligned}$$

   We know that if s is a subset sum of A' there are at most h addends. We see from the last inequality that a_{i-1} must be one of them.

We conclude that for subset sums with at most h addends we may have at most two decipherments when the knapsack is $(h-1)$-increasing and $h \geq 3$. Furthermore at most two trials are needed to find them. This result is not valid for $h = 2$. For example $\{1,2,3,4,5\}$ is a 1-increasing knapsack. Since $1+4 = 2+3 = 5$ we see that a subset sum can have more than two decipherments with at most 2 addends.

It is obvious that further modifications can be made.

Example. The table shows the construction of a 4-increasing knapsack $A = \{a_1, \ldots, a_{10}\}$. Here s_i gives the sum that the next element a_{i+1} must exceed. The value d_i is the amount by which this happens: $a_{i+1} = s_i + d_i$. Notice that the construction is not 5-increasing after the initial superincrease (up to a_5).

i	1	2	3	4	5	6	7	8	9	10
a_i	5	7	14	31	60	115	223	433	847	1631
s_i	5	12	26	57	112	220	429	831	1618	3134
d_i	2	2	5	3	3	3	4	6	13	

The knapsack A is known to be 4-injective, but it turns out to be even 5-injective. It is not 6-injective because $847 + 433 + 223 + 115 + 60 + 5 = 1631 + 31 + 14 + 7$. In addition to this there is only one pair of subset sums contradicting with 6-injectivity: $433+223+115+60+14+7 = 847+5$. In the first pair of sums all elements are involved, the latter pair does not contain 31 and 1631. Adding 31, 1631 and $31 + 1631$ to both sides of the latter equation, we get three more pairs of coinciding subset sums. It turns out that all the

other subset sums are distinct. This means that the number of different values of the subset sums is 1019.

If we transform A with 4-modular multiplication to $2345 \cdot A \pmod{3137}$, we obtain an injective knapsack, although Corollary 5.8 guarantees only 4-injectivity. Notice that 3137 is not even a 5-modulus. Another injective knapsack, and just a little more difficult to find, is $1234 \cdot A \pmod{1723}$, where 1723 is only a 1-modulus. The use of too small a modulus will be discussed in Section 5.5.

5.4 Exceptional blocks in superincreasing structure

We will now modify the superincreasing construction in a way where we do not know about injectivity but the knapsack will be temperate. In Chapter 9 we present some injective constructions, where decryption can be based on the same idea.

Definition 5.13 Assume we have a knapsack A_0 that is easy to decipher. Choose an integer $k > 1$ and such a set X of k elements that

- each of them is greater than s_{A_0} and
- X is an injective knapsack,
- $A_0 \cup X$ is not superincreasing.

Construct the knapsack $A = A_0 \cup X \cup A_1$ by choosing the elements of A_1 with the superincreasing principle. The set X is called an **exceptional block**, or **x-block** in the knapsack A, and the elements of X are **exceptional**, or **x-elements**.

Decryption

Assume s_a is a subset sum of A. It is easy to find out which elements of A_1 are present in the sum. Subtract these from s_a and denote the result by s. If s is smaller than any of the x-elements, we can proceed in the way appropriate to the nature of A_0. If s is bigger than the smallest of the x-elements, we know that some of them are in the sum, but not necessarily which. We cannot but subtract some of the x-elements until what remains is smaller than any of them. If we cannot decipher the remaining sum according to A_0, we must try with other subtrahends. Since we did not assume any specific structure in the x-block to guide our choice, it is clear that k must not be very large.

The search for all decipherments of s can be organized in the following way.

1. Since k is not very large, we may compute all the 2^k subset sums of X. Sort these in increasing order and denote the resulting sequence by $t_0, \ldots, t_{2^k-1}$. Keep also record of the corresponding subsets of X. All this can be done in advance.

2. For a given subset sum s of $A_0 \cup X$, let p be the smallest index for which $s - t_p \leq s_{A_0}$ and q the largest index for which $s - t_p \geq 0$. For $i = p, \ldots, q$ find a decipherment of $s - t_i$ according to A_0, if there is one.

By the choice of p and q we have $0 \leq s - t_i \leq s_{A_0}$ for $i = p, \ldots, q$ and only for these. We need never scan all the sums $t_0, \ldots, t_{2^k-1}$. At least the smallest and the largest sum exclude one another. In general each sum excludes many others (if $k > 1$). For a discussion denote by S_i the subset corresponding to the sum t_i.

Let $p \leq i < j \leq q$. If $S_i \subset S_j$ and $x \in S_j \setminus S_i$, then $t_i + x \leq t_j$ and thus $s - t_i \geq s - t_j + x > 0 + s_{A_0}$, which contradicts the choice of p. Hence $S_i \not\subset S_j$. Since $t_i < t_j$ we also know that $S_i \not\supset S_j$. We conclude that none of the sets $S_p, \ldots, S_q$ is a subset of another in the collection.

Consider generally a set X of k elements. Let Γ be such a collection of subsets of X that no set in Γ includes another set in Γ. This condition is satisfied for example if all sets in Γ have the same number of elements. The maximal size of such a special collection Γ is the maximal value of $\binom{k}{h}$, $h = 0, 1, \ldots, k$. The maximum is attained when $h = \lfloor k/2 \rfloor$ or $h = \lceil k/2 \rceil$. According to [38, p. 302] $\binom{k}{\lfloor k/2 \rfloor}$ is also the maximal size of all collections Γ.

We performed experiments with $k = 9$ and $k = 10$ by constructing as large collections Γ as possible. We started with a random subset and excluded all its subsets and supersets. Among the remaining subsets we chose the next element of Γ and went on until there were no subsets available. Let us call these collections *locally maximal*. Our experiments indicate that on average the size of locally maximal Γ is considerably smaller than the claimed global maximum. For $n = 10$, the claimed maximum is 252, and during the 5512 rounds the average size was about 90.6, and there was none found with more than 157 elements. The distribution of the size looks approximately normal, the standard deviation was about 18.4. For $n = 9$ the results were very similar, over 11000 rounds gave an average of 50.2 and a standard deviation of 11.6.

In the next lemma a special assumption on the set X helps to bound the average number of trials needed in the deciphering process. In the proof we will not make use of the fact that the x-block is injective, but this is a natural assumption for practical purposes.

Lemma 5.14 Suppose we have constructed a knapsack with an x-block of k elements. Assume that

(i) all the messages have an equal probability to occur,

(ii) when there are j x-elements present in a subset sum s of A, we cannot subtract more than j of the x-elements from s before it becomes smaller than any of the remaining x-elements.

Then the average number of trials needed to find all decipherments of a subset sum is at most

$$x_k = \frac{1}{2^k}\binom{2k}{k},$$

and the average number of trials needed to find one decipherment is at most

$$y_k = \frac{1}{2}x_k + \frac{1}{2}.$$

Proof. The second assumption means that in case of j exceptional addends we need at most $\binom{k}{j}$ trials to find all decipherments. By the first assumption the average number of trials needed to find one decipherment is at most $\frac{1}{2}\binom{k}{j}+\frac{1}{2}$, when there are j exceptional addends.

By the first assumption the probability that there are j exceptional addends in the subset sum is $\frac{1}{2^k}\binom{k}{j}$.

When ignoring the values within brackets, the following calculation gives the claimed upper bound x_k for the average number of trials needed for all decipherments. If we take the bracketed parts into account, we get the claimed upper bound y_k for the average number of trials needed for one decipherment.

$$\left.\begin{matrix} x_k \\ [y_k] \end{matrix}\right\} = \sum_{j=0}^{k} \begin{pmatrix} \text{probability that } j \\ \text{addends occur} \end{pmatrix} \times \begin{pmatrix} \text{[average] number of trials} \\ \text{needed for [one] decipherment} \end{pmatrix} \quad (7)$$

$$= \frac{1}{2^{k\,[+1]}} \sum_{j=0}^{k} \binom{k}{j}^2 \quad \left[+\frac{1}{2^{k+1}} \sum_{j=1}^{k} \binom{k}{j} \right] \quad = \quad \frac{1}{2^{k[\,+1]}} \binom{2k}{k} \quad \left[+\frac{1}{2} \right].$$

The formula for the sum of squared binomial coefficients can be found in standard mathematical handbooks. □

The table shows values of x_k and y_k given by the lemma for small values of k. For the meaning of x'_k see below. We have also included the maximal number of trials $\binom{k}{\lfloor k/2 \rfloor}$.

k	x_k	x'_k	y_k	$\binom{k}{\lfloor k/2 \rfloor}$
1	1.00	1.00	1.00	1
2	1.50	1.50	1.25	2
3	2.50	2.50	1.38	3
4	4.38	4.88	2.25	6
5	7.88	8.66	3.97	10
6	14.44	16.45	7.23	20
7	26.81	29.88	13.41	35
8	50.27	57.02	25.14	70
9	94.96	105.70	47.48	126
10	180.43	203.22	90.21	252

About assumption (ii) of the lemma

The second assumption of the lemma may look difficult to satisfy. Notice that if the k x-elements are for example part of a superincreasing sequence, we can decipher them with one trial but they do not satisfy the requirement.

The assumption is satisfied only if the k elements are of nearly equal size. The exact meaning of this nearness depends also on the sum of the preceding elements and the distance from this sum to the smallest x-element.

For example the knapsack $A = \{1, 2, 4, 8, 16;\ 33, 37, 38\}$ with x-block $X = \{33, 37, 38\}$ satisfies the assumption. It suffices to show for $j = 1, 2$ that $j + 1$

exceptional addends cannot be tried if there are only j in a subset sum of A. Subset sums with one addend from X range form 33 to 69. The minimal subset sum of X with two addends is 70, which proves the claim for $j = 1$. For $j = 2$ we note that the maximal subset sum of A with two exceptional addends is 106, whereas the three elements of X give 108 together.

From the treatment of $j = 1$ we see that changing 33 to 32, or 38 to 39 in X would contradict the assumption. Notice also that in any case A is not injective: $37 = 33 + 4$ $(= 32 + 4 + 1)$.

How does the result of the lemma change, if the x-elements do not satisfy the second assumption? The elements are then not close enough to one another. Think about a couple of very big elements that (at least together) exceed the sum of many smaller elements. If the big elements are present in a sum with j exceptional addends, we will be trying the decipherment also with more than j subtrahends. This means that there are more combinations than $\binom{k}{j}$ that we have to try in the worst case.

On the other hand, if the big elements are absent from the sum, we cannot try all the $\binom{k}{j}$ combinations of j subtrahends when reducing the sum.

Intuitively it seems that these effects cancel each other to a certain extent. But it is doubtful whether it happens completely. We will next compute a more rigorous estimate.

Suppose the assumption (ii) of the lemma is changed to be

> When there are j x-elements present in a subset sum s of A, we cannot subtract more than $j + 1$ of the x-elements from s before it becomes smaller than any of the remaining x-elements.

The formula (7) for x_k in the proof will now give the following upper bound for the average number of trials needed for all decipherments:

$$x'_k = \frac{1}{2^k} + \frac{1}{2^k} \sum_{j=1}^{\lfloor k/2 \rfloor - 1} \binom{k}{j} \binom{k}{j+1} + \frac{1}{2^k} \sum_{j=\lfloor k/2 \rfloor}^{k} \binom{k}{j}^2 .$$

The first term corresponds to 0 exceptional addends, where the new assumption causes no extra effort. In the first sum term the maximal number of trials needed to find j addends has been changed to be at most $\binom{k}{j+1}$. This exceeds the actual number at least by $k-j$, because in case of the correct decipherment we can only try with j addends, not $j + 1$.

When $j = \lfloor k/2 \rfloor, \ldots, k - 1$, we have $\binom{k}{j} \geq \binom{k}{j+1}$. It is possible that we can only try with j addends for each of these j and thus the estimate for the maximal number of trials must remain $\binom{k}{j}$ in the second sum. This applies of course also for $j = k$.

Instead of attempting to simplify the formula we include the values of x'_k in the table above. We see that even for $k = 10$ the new assumption does not increase the enciphering effort very much. For $k \leq 3$ there is no change.

About the density

Suppose there is an x-block in an *injective* knapsack A of size n. Then $\rho(A)$ can exceed $\rho_2(n)$ only if the block constitutes the largest elements of the knap-

sack. This follows from Lemma 2.1 and our assumption of continuing with superincrease after the block.

Suppose we only need to decipher subset sums with at most h addends ($h \approx n/2$). If there are at least $h+1$ elements in the knapsack after joining the x-block, we may continue with new elements exceeding only the sum of h previous elements. This would not increase the average number of trials needed for decipherment.

If we weaken the superincrease in this way already before the x-block, the averages may grow a little, especially if also the x-elements only exceed the sum of h previous elements. This is because it is more difficult to satisfy the assumption (ii) of the lemma. On the other hand, growth of the averages may be inhibited by the fact that the smaller number of allowed addends may rule out some combinations.

Several x-blocks

There may be several x-blocks in a knapsack. Deciphering a subset sum then requires a number of trials that is expected to be roughly the product of the averages connected with the individual blocks. If we want to insert 20 x-elements in a knapsack, then computing according to the table we see that two blocks of ten would require about 8.4 times more effort ($x_{10}^2 \approx 32553$ trials) than four blocks of five ($x_5^4 \approx 3856$). For a single block of 20 the effort would be about 34-fold ($x_{20} \approx 131461$).

5.5 "Too small" a modulus

Let A be an injective knapsack of size n. Assume that $1 \leq k < n$, m is a k-modulus for A and w is relatively prime to m. Denote $B = wA \pmod m$.

We know that the knapsack B is k-injective, but it is possible that B is fix-h-injective or h-injective for a value of h that is larger than k. This may be difficult to show, if n is large enough for cryptographic use. We will investigate how much extra effort the possibly missing h-injectivity causes. Basically the treatment will be restricted to a fixed number of addends. Normally we are interested in cases where h is near $n/2$. This is not necessary here.

Suppose that s_b is a subset sum of B with h addends ($1 \leq h \leq n$), and s_a is the corresponding subset sum of A. Denote $s = w^{-1}s_b \pmod m$. We have $s_a \equiv s \pmod m$ and thus $s_a = s + tm$ for some $t = 0, 1, \ldots$. In the search for t we cannot but try decipherment of $s + um$ for all such values u, that $s_{min} \leq s + um \leq s_{max}$, where s_{min} is the smallest and s_{max} the largest possible subset sum of A with h addends.

Denote by z the number of possible values for u. If A is easy to decipher, then all decipherments with h addends of s_b can be found in time that is z times the maximal time required to decipher a subset sum of A.

If we know s and m, we can compute the number z exactly. More generally, z can be bounded by knowledge of h, k and A only. Firstly $\Delta = s_{max} - s_{min}$ depends only on h and A. If we compute $\lceil \frac{\Delta}{m} \rceil$ we obtain either z or $z+1$. If

we replace m here by the sum of the k largest elements of A we get an upper bound for z. Unless k is very small, this is a reasonably tight bound, because we may assume that m is not a $(k+1)$-modulus.

A derivation based on even dispersion

We will now derive an estimate for z under the assumptions that the knapsack A is not very small and that its elements are approximately evenly dispersed between $a_1 = \min(A)$ and $a_n = \max(A)$. Let $h = \alpha n$.

Assumption of an even dispersion implies that the ith element of A is approximately $f(i) = a_1 + \frac{i-1}{n-1}(a_n - a_1)$, $i = 1, \ldots, n$. Hence the average of the h smallest elements is approximately the same as the value of the function f in the middle of 1 and h, which is $f(\frac{1+h}{2}) = f(\frac{\alpha}{2}n + \frac{1}{2})$. Similarly the average of the h largest elements is approximately $f(\frac{n-h+1+n}{2}) = f(n - \frac{\alpha}{2}n + \frac{1}{2})$. Multiplying these averages by h and subtracting the results gives

$$\Delta := \alpha n \frac{(1-\alpha)n}{n-1}(a_n - a_1) \approx \alpha(1-\alpha)n(a_n - a_1),$$

where the approximation is justified because n is not very small.

Let $k = \beta n$. The sum of the k largest elements is approximately

$$m_k := kf(\tfrac{n-k+1+n}{2}) = \beta n f(n - \tfrac{\beta}{2}n + \tfrac{1}{2}) = \beta n\Big(a_1 + \tfrac{n-\frac{\beta}{2}n-\frac{1}{2}}{n-1}(a_n - a_1)\Big).$$

Since n is fairly large and $\beta < 1$ we do not make a large error when ignoring $-\frac{1}{2}$ in the numerator and -1 in the nominator. Hence

$$m_k \approx \beta n\Big(a_1 + (1 - \frac{\beta}{2})(a_n - a_1)\Big)$$

and

$$\frac{\Delta}{m_k} \approx \frac{\alpha(1-\alpha)n(a_n - a_1)}{\beta n\left(a_1 + (1-\frac{\beta}{2})(a_n - a_1)\right)}.$$

Denoting $\gamma = \frac{a_n}{a_1}$ we can write this in the following form.

$$\frac{\Delta}{m_k} \approx \frac{\alpha(1-\alpha)(\gamma-1)}{\beta\left(1 + (1-\frac{\beta}{2})(\gamma-1)\right)} = \frac{2\alpha(1-\alpha)(\gamma-1)}{\beta(2\gamma - \beta(\gamma-1))}.$$

We state the result as a lemma.

Theorem 5.15 Assume that the elements of an injective knapsack A are approximately evenly dispersed between $\min(A)$ and $\max(A)$. Transform A with k-modular multiplication to a knapsack B.

Let $h > k$. The maximal number of trials needed to find all decipherments with h addends of a subset sum of B is approximately

$$z(\alpha, \beta, \gamma) = \frac{2\alpha(1-\alpha)(\gamma-1)}{\beta(2\gamma - \beta(\gamma-1))}.$$

where $\alpha = \dfrac{h}{n}$, $\beta = \dfrac{k}{n}$ and $\gamma = \dfrac{\max(A)}{\min(A)}$. □

Properties of the function z. When considering $z = z(\alpha, \beta, \gamma)$ as a function of continuous variables $\alpha \in (0,1)$, $\beta \in (0,1)$ and $\gamma \in (1, \infty)$, it is easy to see that

- if only α varies then z is symmetric with respect to $\alpha = 0.5$ and obtains its maximal value at this point.
- the partial derivative of z with respect to β is negative, whence z is a decreasing function of β.
- the partial derivative of z with respect to γ is positive, whence z is an increasing function of γ. If γ grows to infinity then z tends to $\frac{2\alpha(1-\alpha)}{\beta(2-\beta)}$.
- if $\beta \geq \alpha$, then $z < 1$. This can be seen by setting $\beta = \alpha$. This gives $z < 1$ if $\gamma > 1 - \frac{2}{\alpha}$, which is always true.

The following table shows the cryptographically interesting part of the function $z(\alpha, \beta, \gamma)$. Notice that if $n = 200$, the value $\beta = 0.005$ means 1-modular multiplication. It seems that even such an extreme case can be handled when the number of addends is also known. Such a modulus has the advantage of having hardly any effect on the density of the knapsack.

$z(\alpha,\beta,\gamma)$	$\alpha = 0.5$				$\alpha = \begin{cases} 0.4 \\ 0.6 \end{cases}$
$\beta \setminus \gamma$	2	3	5	∞	∞
0.005	25.03	33.39	40.08	50.13	48.12
0.01	12.53	16.72	20.08	25.13	24.12
0.02	6.28	8.39	10.08	12.63	12.12
0.03	4.20	5.61	6.75	8.46	8.12
0.05	2.53	3.39	4.08	5.13	4.92
0.07	1.82	2.44	2.94	3.70	3.55
0.10	1.28	1.72	2.08	2.63	2.53
0.15	0.87	1.17	1.42	1.80	1.73
0.20	0.66	0.89	1.09	1.39	1.33
0.30	0.45	0.62	0.76	0.98	0.94

Remark 1. The values in the table estimate the number of trials needed to decipher completely a subset sum of B, when the number of addends is fixed and near $\frac{1}{2}|B|$. The exact number of required trials can always be computed in practical situations (for a given subset sum and number of addends, or an exact general bound). In general all these trials do not lead to a decipherment with the knapsack A. Furthermore all the succesful decipherments with A do not necessarily correspond to the subset sum of B. And finally, among the otherwise correct decipherments there may be such that do not have the right number of addends.

Especially if our system has such parameters β and γ that $z(0.5, \beta, \gamma) < 1$, then we expect to need only one trial to find a unique decipherment of every subset sum of B, where we know the number of addends. This means that the exact computations are likely to show that B is fix-injective.

Remark 2. If we want to find an estimate for the effort needed when the number of addends (h) is allowed to vary, we must determine the corresponding range $\Delta = s_{max} - s_{min}$ between the minimal and maximal subset sum of A. If there is no restriction on h, this will give us $\Delta = s_A$. As in the derivation of the lemma we have now $s_A = \frac{1}{2}n(a_1 + a_n)$ and

$$\frac{\Delta}{m_k} \approx \frac{\gamma + 1}{\beta(2\gamma - \beta(\gamma - 1))} =: z'(\beta, \gamma).$$

For a comparison we compute the ratio

$$r := \frac{z'(\beta, \gamma)}{z(\alpha, \beta, \gamma)} = \frac{\gamma + 1}{2\alpha(1 - \alpha)(\gamma - 1)}.$$

Actually r is the ratio of the two intervals (Δ's), which explains why it does not depend on β. For a fixed γ the ratio r has its minimum at $\alpha = \frac{1}{2}$. For a fixed α the ratio is a decreasing function of γ and tends to $\frac{1}{2\alpha(1-\alpha)}$ as $\gamma \to \infty$.

If all messages are equally likely to occur then almost all of them contain a number of ones that is between $0.4n$ and $0.6n$ (recall that n is not very small). Thus, when the number of addends is given, the number of trials needed for decipherment is almost always bounded by $z(\alpha, \beta, \gamma)$, where $0.4 \leq \alpha \leq 0.6$. Assuming that $\gamma \geq 2$, we obtain the largest r by plugging in $\alpha = 0.4$ and $\gamma = 2$. This gives $r = 6.25$. This is a factor that estimates the extra effort caused by not knowing the number of addends, which is allowed to vary from 0 to n.

What if the elements are not evenly dispersed?

There are two contradicting goals to pursue when k-modular multiplication is used: a small number of trials needed for decipherment and a high density of the transformed knapsack B. A small k requires many trials but gives a higher density. If we want to reduce the number of trials we may also try to distribute the elements of the initial knapsack A in some way different from evennes. Let us assume in the sequel that the number of addends in subset sums is fixed to h, which is near $n/2$ (for simplicity).

1. Consider first the superincreasing knapsacks and their simple modifications (as in Sections 5.2 or 5.4). Their elements are approximately exponentially dispersed. When h, the number of addends, is not very near n, the range $s_{max} - s_{min}$ between the smallest and largest sum is very near s_{max}. On the other hand this is already near the sum of all elements, s_A. If we divide such a range by a k-modulus we get a result that is quite near 1. As we will see next, the ratio may be smaller than 2 already for a 1-modulus.

If we use 1-modular multiplication for the knapsack $A = \{1, 2, 4, \ldots, 2^{n-1}\}$, we can decipher all subset sums of the resulting knapsack B with two trials. This is because $\frac{s_A}{m} < 2$, when the modulus m satisfies $m > 2^{n-1}$. In most cases two decipherments will be found according to A, if m is not much larger than 2^{n-1}. As we already mentioned in Remark 1, it may well happen that only one

of these carries over to B. Notice that the knapsack B without the element corresponding to 2^{n-1} is injective.

We conclude that for superincreasing knapsacks it does not make much difference either in the number of trials or the density, whether we use a 1-modulus or a strong modulus. Of course the (possible) lack of injectivity is an essential difference, but we doubt that the modulus is still too large to hide a superincreasing structure from the attacks that are based on equation (5) on page 24.

2. If all the elements of A are of the same order of magnitude and the modulus is not much larger either, then most of the quotients $\frac{a_i}{b_i m}$ in equation (5) are not very small. This would obviously cause difficulties for the cryptanalyst.

Elements of nearly equal size would mean a small value of γ, and as we infer from Theorem 5.15, this would be advantageous also for the number of trials needed for decryption.

Knapsacks with small values of γ (e.g. near 2) can be obtained especially with the congruential construction of Section 8.1. The elements can also be distributed approximately evenly, or in the way that will be described next.

3. Let us now return to reducing the number of trials by distibuting the elements suitably. We can make Δ smaller by concentrating most of the elements in the middle of $a_1 = \min(A)$ and $a_n = \max(A)$. The number of trials will then be smaller than in Theorem 5.15, at least for $k = 1$. For larger k the k-modulus m_k can be smaller than with even dispersion. This increases the quotient $\frac{\Delta}{m_k}$ but it may still not exceed the value in the lemma. Actually, a rough estimate with an extreme concentration gives $\Delta \approx a_n - a_1$ (all the other elements being almost equal) and $m_k \approx a_n + (k-1)\frac{1}{2}(a_1 + a_n)$, whence

$$\frac{\Delta}{m_k} \approx \frac{a_n - a_1}{a_n + \frac{1}{2}(k-1)(a_1 + a_n)} = \frac{\gamma - 1}{\gamma + \frac{1}{2}(k-1)(1+\gamma)}.$$

This is smaller than 1 for all values $k \geq 1$ and $\gamma > 1$.

4. Suppose then that the knapsack elements are concentrated near $\max(A)$ This seems to be typical for dense injective knapsacks. As an example consider $A = \{156, 248, 296, 319, 328, 329, 334, 339, 342, 350\}$. Choosing a small k for a k-modulus is critical for the development of density. If there are only a few very small elements then it may happen that the range Δ will not be very large in comparison to $\max(A)$. In the example $\Delta = 347 < \max(A)$, when $h = n/2 = 5$. For other values of h we get a smaller Δ (which is a general property of Δ). Hence for any given number of addends we need only one trial to decipher a knapsack, which is obtained from A by 1-modular multiplication. Especially these knapsacks are fix-injective.

We transformed the knapsack A using the 1-moduli $m = 351, \ldots, 361$ and all possible multipliers $w = \lfloor m/2 \rfloor, \ldots, m-1$. None of the resulting knapsacks were injective, not even 4-injective.

We conclude this chapter by summarizing the study of temperate knapsacks in three questions, that deserve further research.

1. What is the effort required for complete decipherment, when the number of addends is either given, arbitrary or within some limits?

   What is the bound for the number of trials a priori, i.e. when no subset sum is given?

   Can decipherment be essentially easier for some subset sums, and if so, what is the average number of trials needed?

2. What is the probability that only one (acceptable) decipherment is found (even if several trials must be made)?

3. What other ways are there to construct temperate knapsacks?

6 Modifications of some proposed knapsack systems

Almost all proposed knapsack cryptosystems have been injective. All injective systems can be generalized by using some weak form of injectivity. If no assumptions of the construction or the trapdoor transformation can be relaxed to achieve this, then adding a constant to the elements of the public key will do it.

In this chapter non-injectivity will only be in a subsidiary role and in later chapters even more so. We will present here three well-known knapsack constructions and modify slightly two of them, in Sections 6.1 and 6.2. The third system in Section 6.3 is actually an outline of several construction algorithms. We present it here because Chapters 7–9 can be viewed as generalizations of it.

6.1 Merkle-Hellman multiplicative knapsack

Merkle and Hellman presented in [24] the following algorithm to generate multiplicative trapdoor knapsacks. The result is a 'normal', i.e. additive knapsack A with n elements. Only using the trapdoor will lead to a multiplicative knapsack.

1. Choose n relatively prime numbers $p_1, p_2, \ldots, p_n$, greater than 1. These numbers form a multiplicative knapsack that is easy to decipher: for each i the element p_i is present in a subset product π if and only if $p_i \mid \pi$.

2. Choose a prime q that is greater than the product of all the p_i.

3. Choose a primitive root b modulo q, that is, a generator of the multiplicative group of the finite field $GF(q)$.

4. Compute the discrete logarithms of the p_i to base b in the finite field $GF(q)$, and denote these by a_i. Thus the a_i are integers that satisfy

$$p_i \equiv b^{a_i} \pmod{q}, \qquad 1 \leq a_i \leq q-2.$$

Denote $A = \{a_1, a_2, \ldots, a_n\}$.

Assume we have a subset sum $s = \sum a_j$ of A. Then

$$b^s = b^{\sum a_j} = \prod b^{a_j} \equiv \prod p_j \pmod{q}.$$

Since $q > \prod_{i=1}^{n} p_i$, this shows that the smallest positive remainder of b^s modulo q equals $\prod p_j$, i.e. the product of those p_j that correspond to the a_j in the subset sum. Deciphering the product then solves also the subset sum problem.

Considering the way how injectivity of A is implied by the calculation above, we see that A is also injective modulo $q-1$. This follows from the fact that $b^x \equiv b^y \pmod{q}$ if and only if $x \equiv y \pmod{q-1}$.

The cryptosystem can be broken both by finding the trapdoor information and by the low density attack, as shown by Odlyzko in [26]. If $n = 100$ and the p_i are the n smallest primes, then q must be greater than 2^{730} which means that the density becomes very low, about $\frac{100}{730}$. If we know that there are at most h addends in the sum instances, it suffices to take q larger than the product of the h largest primes among the p_i. In the above case of $n = 100$ the value 50 for h would mean that $q > 2^{426}$. This is an improvement but it still leads to an insecure density.

We may try to improve the density further by using the non-injective approach of "too small" a modulus. We may compute the number of trials needed to decipher a subset sum with h addends in the same way as in Section 5.5. The number is $\lceil \frac{1}{m}(\pi_{max} - \pi_{min}) \rceil$, where m is the modulus, π_{max} and π_{min} are the largest and the smallest product of the p_i with h factors.

Suppose that p_i are the n smallest primes in increasing order, n is not very small, and h is relatively near $n/2$. Then $\pi_{max} - \pi_{min}$ will be approximately $\pi_{max} = \prod_{i=n-h+1}^{n} p_i$. Suppose that $1 \leq k < h$ and m is slightly larger than $\prod_{i=n-k+1}^{n} p_i$, i.e. the product of the k largest numbers p_i. Then the number of trials required for decipherment is at most but nearly $\prod_{i=n-h+1}^{n-k} p_i$. This is a product of $h-k$ primes, which are not among the very smallest ones. Thus the number is small only if k is $h-1$, or perhaps $h-2$. Since we need a temperate knapsack, we cannot improve the density very much in comparison to an h-injective knapsack. For example if $n = 100$ and $h = 50$, we have 233 trials for $k = 49$ and 55687 for $k = 48$.

A modified construction

Assume that $p_1, p_2, \ldots, p_t$ are t distinct primes, $t \geq 1$. Assume that the t-dimensional vectors $\bar{e}_i = (e_{i1}, \ldots, e_{it})$, $i = 1, \ldots, n$, consist of non-negative integers. If the vectors $\bar{e}_i$ are distinct, we may consider the set $E = \{\bar{e}_1, \ldots, \bar{e}_n\}$

as a t-dimensional additive knapsack and apply to it all the other suitable terminology we have in use. For $i = 1, \ldots, n$ denote

$$r_i = \prod_{j=1}^{t} p_j^{e_{ij}}.$$

If the set E possesses some of the injectivity properties of Definition 5.1 (i)-(iii) then the set $R = \{r_1, \ldots, r_n\}$ has the corresponding properties when considered as a multiplicative knapsack. If furthermore it is easy to decipher the (relevant) subset sums of E, then the same is true of the corresponding subset products of R. We may namely easily factor each such product, and presenting the multiplicities of $p_1, \ldots, p_t$ in the factorization as a vector gives the corresponding subset sum of E.

Now we can construct a trapdoor knapsack A as in the algorithm above by starting from phase 2 (using the r_i instead of the p_i). If we can do with an h-injective or fix-h-injective knapsack then it suffices to choose q larger than the product of the h largest numbers among the r_i. Doing so we must require that E is h-injective or fix-h-injective correspondingly.

The Merkle-Hellman system uses $h = t = n$ and

$$\bar{e}_i = (0, \ldots, 0, \overset{i}{1}, 0, \ldots, 0),$$

that is, $r_i = p_i$ for $i = 1, \ldots, n$. A different choice of the set E may give trapdoor knapsacks with higher densities. No substantial improvement is to be expected, however, as we will see next.

If we want to keep the density as high as possible, we should only use the smallest primes and also small values for the components of the $\bar{e}_i$. This leaves a very limited set of numbers r_i at our disposal. This would mean that Odlyzko's attack would find the trapdoor. Sufficient for this is namely that one knows several of the numbers r_i, and that q is not too large.

As an example consider the case $n = 100$, $h = 50$. As it will be seen we can take $t = 90$. Let $p_1, p_2, \ldots, p_{90}$ be the first 90 primes in increasing order ($p_{90} = 463$).

We do not construct the vectors $\bar{e}_i$ explicitly. Instead choose the values $r_1, \ldots, r_{17}$ as the following powers of the 7 smallest primes.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
r_i	8	32	64	128	3	9	81	5	25	7	49	11	121	13	169	17	289

The exponents of 2 form the set $\{3, 5, 6, 7\}$, which is an injective knapsack. The exponents of 3 form the superincreasing knapsack $\{1, 2, 4\}$. For all the other primes included in the table there are only the exponents 1 and 2. Until now we have used the primes $p_1, \ldots, p_7$. For $i = 18, \ldots, 100$ set $r_i = p_{i-10}$. Clearly the set $\{r_1, \ldots, r_{100}\}$ is a multiplicative knapsack, whose all subset products are distinct and also easy to decipher.

Since $r_{51} = p_{41} = 179$, among the 50 largest of the r_i there is only the number 289 from the table above. Thus the lower bound for q will be $289 \cdot \prod_{i=52}^{100} r_i = 289 \cdot \prod_{i=42}^{90} p_i \approx 2^{414}$. Using such a q for phases 3 and 4 of the

construction algorithm gives us a 50-injective knapsack with density at most 0.2415.

We might think of generating a temperate multiplicative trapdoor knapsack by allowing the knapsack E to be temperate. An upper bound for the density of such a knapsack can be computed by assuming that the numbers r_i are taken to be $2, 3, 4, \ldots, n+1$. Even in such an extreme (and not very temperate) case the prime q becomes considerably larger than 2^{n-1} — also if it only exceeds the product of the $\lceil \frac{n}{2} \rceil$ largest of the r_i. This shows that knapsacks based on the multiplicative trapdoor scheme cannot be secure, at least without further modifications.

More general sum instances

Although we only deal with subset sums in this thesis we note the following generalization of the original multiplicative trapdoor scheme. Some of the elements of A can appear in sum instances with multiplicities higher than 1 and we still have unique and easy decipherment if q is sufficiently large. For example if q is chosen larger than the square of the product of all the p_i then we may decipher every *ternary* vector $T \in \{0, 1, 2\}^n$ from the product $T \cdot A$. By restrictions similar to h-injectivity we may obtain somewhat more reasonable size for q. The reason why this kind of a modification could be useful is that the special vector $\hat{e}$ searched by the low density attacks (cf. p. 11) is not so short when several components of $\hat{e}$ are greater than 1. It seems however that the density is still too low for this approach to offer any security.

6.2 Graham-Shamir knapsack

The following construction can be found in [35].

1. Construct a superincreasing knapsack A.
2. Represent each element of A in binary form. Assume that the longest sequence is t bits long.
3. Catenate zeros in front of each of these binary sequences so that they become $t + 1$ bits long.
4. Catenate arbitrary binary sequences in front of these. Denote the resulting set of integers by $\tilde{A}$.

The knapsack $\tilde{A}$ is easy to decipher, because taking only the $t + 1$ lowest order bits of a subset sum gives a subset sum of the superincreasing knapsack. Notice that if $s_A < 2^t$, then the length t suffices in phase 3 (and in decryption). The knapsack $\tilde{A}$ is finally transformed into an arbitrary looking knapsack B by the normal strong modular multiplication.

The attack by Adleman in [1] showed that strong modular multiplication fails to hide the superincreasing structure. The knapsack is insecure also because of its fairly low density. To get an impression of this, assume that the knapsack size is n and the bit sequences of phase 4 are randomly chosen from

$0, \ldots, 2^r - 1$, where r is not very small. Then the elements of $\tilde{A}$ are nearly evenly dispersed between 2^{t+1} and 2^{t+r+1}, with the exception of a few cases where the sequence is 0. The average size of the elements will be at least 2^{t+r-1} and hence the sum of $\tilde{A}$ is at least $n2^{t+r}$. Since A is superincreasing, we have $t \geq n$, and the sum of $\tilde{A}$ is at least $n2^{n+r}$. This is an estimate for the largest element of the knapsack B, whence

$$\rho(B) \approx \frac{n}{n + r + \log_2 n}. \tag{8}$$

If $n < 135$ this formula does not give for any $r > 0$ a density higher than 0.95 (cf. the critical densities on p. 13). For $n = 200$ and $r = 3$ the formula gives 0.949.

If we let A be any injective knapsack, we may get a B that is resistant against the Adleman's attack. However the problem of density cannot be remedied in this way, because we must now choose the t in phase 2 so that $2^t > s_A$. Since always $s_A \geq 2^n - 1$, we have again $t \geq n$.

A deterministic modification

We may try to avoid the decrease of density by choosing the highest order bits so that the modular multiplication yields a result of moderate size. This means giving up the advantages of randomness, but the extra bits may still be useful in hiding some structures that may be vulnerable if only a modular multiplication is performed. For example the congruential knapsacks of Section 8.1 may benefit from this technique.

In the following algorithm the highest order bits are joined as extra bits to elements of a knapsack A, giving a knapsack $\tilde{A}$, which is transformed to a public key B. The original Graham-Shamir scheme is actually only modified by exploiting $\lceil \frac{n}{2} \rceil$-injectivity. Assume for simplicity that n is even, and not very small.

1. Construct an $\frac{n}{2}$-injective knapsack $A = \{a_1, \ldots, a_n\}$ that is easy to decipher.

2. Compute such a t that 2^t is greater than the sum of the $\frac{n}{2}$ largest elements of A.

3. Choose an $r \geq 1$, a modulus $m > 3n2^{r+t-3}$ and a multiplier w that is relatively prime to m and not very small.

4. For each $a_i \in A$ find the number v_i, $0 \leq v_i \leq 2^r - 1$ that gives the smallest remainder of $w(v_i 2^t + a_i)$ modulo m, and denote this remainder by b_i. Denote $\tilde{a}_i = v_i 2^t + a_i$.

5. If m is an $\frac{n}{2}$-modulus for the knapsack $\tilde{A} = \{\tilde{a}_1, \ldots, \tilde{a}_n\}$, use $B = \{b_1, \ldots, b_n\}$ as the public key. Otherwise go back to phase 3 and choose another w. If this does not appear to help (in a few rounds) choose a larger m.

In the construction $2^t + a_i$ means that we first catenate zeros in front of the binary representation of a_i so that it becomes a t-bit-long sequence $\tilde{\alpha}_i$. Then we put a 1 in front of $\tilde{\alpha}_i$ as the $(t+1)$th bit. In $\tilde{a}_i = v_i 2^t + a_i$ we multiply this bit by an r-bit number v_i, which means catenation of the bits of v_i in front of $\tilde{\alpha}_i$. For each i we have chosen the v_i so that $w\tilde{a}_i \pmod m$ is the smallest possible.

As in the original Graham-Shamir knapsack, it is easy to decipher all subset sums of $\tilde{A}$ with at most $\frac{n}{2}$ addends, because the t lowest order bits in these sums give the corresponding subset sums of A. If the knapsack B is accepted in phase 5, then it was obtained from $\tilde{A}$ by $\frac{n}{2}$-modular multiplication in phase 4. Hence $\frac{n}{2}$-injectivity and the ease of decipherment is guaranteed for B. It remains to discuss the density.

Density. Let us first explain the choice of m. If the numbers v_i were chosen randomly, we would expect to get at most $n(3 \cdot 2^{r+t-3} - 2^{t-2}) + 2^t$ as the sum of the $\frac{n}{2}$ largest elements $\tilde{a}_i = v_i 2^t + a_i$ of $\tilde{A}$. The first term is $\frac{n}{2}$ times the average of the numbers $v2^t$, $v = 2^{r-1}, \ldots, 2^r - 1$. According to phase 2 the second term 2^t is an upper bound for the sum of the $\frac{n}{2}$ largest elements of A.

The numbers v_i are not random, however. Instead v_i depends on a_i. If the knapsack A were random, then the distribution of the components in the vectors $(v_1, \ldots, v_n)$ would also be random. It seems very unlikely that the injectivity and easiness of A would cause considerable deviation from this randomness. Notice that already a small change in a_i changes v_i, because w was assumed not to be very small.

Since n is not very small we have $n(3 \cdot 2^{r+t-3} - 2^{t-2}) + 2^t < 3n2^{r+t-3}$, and we expect that numbers larger than this will qualify as $\frac{n}{2}$-moduli for $\tilde{A}$. Trying with another w before increasing m is just to avoid bad luck in this respect.

Let us now turn to the density of B. For an estimate assume $w2^t \pmod m$ is not very small in comparison to m. We have chosen each b_i from a set of 2^r residuals of the form $vx + y \pmod m$, where v varies from 0 to $2^r - 1$ and x is not very small. Since such residuals are approximately uniformly distributed between 1 and $m - 1$, we expect to have $b_i \approx \frac{m}{2^r}$. Since $m \approx 3n2^{r+t-3}$, the density of the knapsack is

$$\rho(B) \approx \frac{n}{\log_2(3n2^{r+t-3}2^{-r})} = \frac{n}{t - 3 + \log_2 3n}.$$

We see that the special choice of the r highest order bits allows us to completely eliminate their effect on the density.

Suppose now that the initial knapsack A is relatively dense, $\rho(A) \approx \rho_2(n)$, and its elements are approximately evenly dispersed in the range $1, \ldots, 2^{n-1}$. Then the sum of the $\frac{n}{2}$ largest elements of A is approximately $3n2^{n-4}$. Now t must satisfy $2^t > 3n2^{n-4}$, and hence $t \approx n - 4 + \log_2 3n$. This gives

$$\rho(B) \approx \frac{n}{t - 3 + \log_2 3n} \approx \frac{n}{n - 7 + 2\log_2 3n} \approx \frac{n}{n - 3.83 + 2\log_2 n}.$$

This is an increasing function of n and approaches 1 as n grows. But for example $n = 200$ gives only 0.946. Although we now have an A with even

dispersion, the result is only slightly lower than 0.949, which we computed for the original Graham-Shamir knapsack from equation (8), with $n = 200$ and $r = 3$.

If we assume also here that A is superincreasing, we can estimate t with n and the last calculation gives

$$\rho(B) \approx \frac{n}{n - 3 + \log_2 3n} = \frac{n}{n - 1.415 + \log_2 n},$$

and we get 0.970 for $n = 200$.

For further improvement of the density we must use some non-injective technique. As we already noted, the elements of $\tilde{A}$ are approximately evenly dispersed. Hence we may use the estimate of Theorem 5.15 for the approach with "too small" a modulus. Notice that although our modification cancels the effect of the *amount* of the extra bits, they still cause the need of a large modulus. In comparison to an evenly dispersed initial knapsack A this effect lies completely in the additional zeros that are required in the construction. This is also the reason why the modification does not improve the density very much.

A "disconnecting" modification

If the initial knapsack A is not superincreasing (and not even near it), there may be nothing special to hide in its highest order bits. On the other hand, insertion of zeros *in the middle* of the elements of A may help hiding their structure, together with the subsequent modular multiplication. These zeros cause a similar decrease in density as the zeros in the Graham-Shamir scheme when applied to a non-superincreasing knapsack. There are no obvious ways to compensate this effect, except the use of "too small" a modulus. We will evaluate this approach after presenting the algorithm. Assume again that n is even and not very small.

1. Construct an $\frac{n}{2}$-injective knapsack $A = \{a_1, \ldots, a_n\}$ that is easy to decipher.

2. Choose such a $p \geq 1$ that none or not very many of the a_i satisfy $a_i < 2^p$. Denote by d_i the integer consisting of the p lowest order bits of a_i and by c_i the integer consisting of the remaining bits. Thus $a_i = c_i 2^p + d_i$, $i = 1, \ldots, n$.

3. Compute the smallest q such that 2^{p+q} is greater than the sum of the $\frac{n}{2}$ largest integers among $d_1, \ldots, d_n$. Compute $\tilde{a}_i = c_i 2^{p+q} + d_i$, which means insertion of q zero bits between the p'th and $(p+1)$th bit of a_i.

4. Transform the knapsack $\tilde{A} = \{\tilde{a}_1, \ldots, \tilde{a}_n\}$ to a public key B by t-modular multiplication, where $t = \frac{n}{2}$.

Decryption. Assume we have a subset sum $\sum b_j$ of B with at most $\frac{n}{2}$ addends. Inverting the modular multiplication gives the corresponding subset sum $\tilde{s} = \sum \tilde{a}_j$ of $\tilde{A}$. Denote by δ the integer consisting of the $p+q$ lowest order bits of $\tilde{s}$ and by γ the integer consisting of the remaining bits. Thus $\tilde{s} = \gamma 2^{p+q} + \delta$. On the other hand $\tilde{s} = 2^{p+q} \sum c_j + \sum d_j$ and by the choice of q we know that $\sum d_j$ does not interfere with bits higher than $p+q-1$. Thus $\delta = \sum d_j$ and $\gamma = \sum c_j$, and $s = \gamma 2^p + \delta = 2^p \sum c_j + \sum d_j$. That is, s is the subset sum of A that corresponds to $\sum b_j$.

Density when the modulus is "too small". Let us first show that q is at most $\lceil \log_2 n \rceil - 1$ and usually equals this number or this minus 1. Usually the p lowest order bits of the elements of an easy knapsack have no specific structure. Hence the $\frac{n}{2}$ largest elements d_i are approximately evenly dispersed between 2^{p-1} and $2^p - 1$. Their sum is roughly $3n2^{p-3}$ and in any case it is smaller than $n2^{p-1}$. Now 2^{p+q} must be larger than the sum.

Using the upper bound we get $2^{p+q} \geq n2^{p-1}$, whence $q \geq \log_2 n - 1$. This shows that q never needs to be larger than $\lceil \log_2 n \rceil - 1$. On the other hand, using the average we get $2^{q+p} > 3n2^{p-3}$, which gives

$$q > \log_2 n + \log_2 3 - 3 \approx \log_2 n - 1.4.$$

In some cases this allows $q = \lceil \log_2 n \rceil - 2$.

Suppose again that the initial knapsack A is relatively dense, $\rho(A) \approx \rho_2(n)$, and its elements are approximately evenly dispersed in the range $1, \ldots, 2^{n-1}$. Since we are interested in fairly small values of k we can approximate a k-modulus m for A by $k2^{n-1}$. For all i we have $\tilde{a}_i \leq 2^q a_i$ so that $m2^q$ will be a k-modulus for $\tilde{A}$. The k-modular multiplication gives a knapsack B with

$$\max(B) \approx 2^q k 2^{n-1} \approx 3n2^{-3}k2^{n-1} = 3nk2^{n-4}$$

and hence

$$\rho(B) \approx \frac{n}{n - 2.415 + \log_2 nk}.$$

For $n = 200$ this gives a density higher than 0.95 as long as $k < 40$. Theorem 5.15 implies that for so large values of k we need a very moderate number of trials to decipher subset sums, where the number of addends is known. In some cases we may be even able to show that the knapsack B is fix-injective as suggested in Remark 1 after the theorem.

6.3 Constructions proposed by Desmedt et al.

Desmedt, Vandewalle and Govaerts define in [10] two classes of knapsacks where it is easy to decipher one bit of the message. The class EH consists of such knapsacks where one element exceeds the sum of all the others. The class ED consists of such knapsacks where all elements except one are divisible by an integer $d > 1$. In both cases the presence of the special element in a subset sum can be directly checked and so the bit corresponding to this element can be easily deciphered.

As we mentioned in Section 1.5, Desmedt et al. showed that there are knapsacks that cannot be obtained by strong modular multiplication from any of the knapsacks in a certain class. This class is $EH \cup ED \cup ES$ where ES stands for Graham-Shamir knapsacks (before strong modular multiplication).

The following algorithm is presented in [10] for construction of public enciphering keys:

1. Start from a knapsack A that is easy to decipher, e.g. a superincreasing or a Graham-Shamir knapsack.

2. Transform A to A' with strong modular multiplication or some other secret transformation.

   If the size of A' is sufficient then stop and use A' as the public key.

3. Introduce such a number α and transform A' to such an $\tilde{A}$, that $A'' = \tilde{A} \cup \{\alpha\}$ is injective and easy to decipher. Two examples of this step follow the ideas of EH and ED:

   (i) Set $\tilde{A} := A'$ and choose α so that it exceeds the sum of A'.

   (ii) Choose an integer $d > 1$ and set $\tilde{A} := dA'$, choose α so that it is not divisible by d.

4. Set $A := A''$ and go to step 2.

If strong modular multiplication is used in step 2, the density of the resulting knapsack will be very low, regardless of which of the methods (i) and (ii) is used in 3. To see this write the knapsacks of phases 2 and 3 with a subscript j indicating the round.

Strong modular multiplication causes an approximately even dispersion of the elements of A'_j, at least if A'_j is not very small any more. This means that $s_{A'_j} \approx \frac{1}{2}|A'_j| \max(A'_j)$. As soon as $|A'_j| > 6$, this means that $s_{A'_j}$ is at least $3 \max(A'_j)$.

The modulus in step 2 during the $(j+1)$th round will be at least $6 \max(A'_j)$. Namely, after method (i) it is at least $\alpha + s_{A'_j} \geq (3+3)\max(A'_j)$ and after method (ii) at least $2s_{A'_j} \geq 2 \cdot 3 \max(A'_j)$.

Thus, the maximal element increases with a factor at least 6 as new elements are introduced. This means that the density will eventually sink well below 1.

Chapter 7 generalizes the above algorithm and Chapters 9 and 8 can be seen as generalizations of the methods (i) and (ii). Lemma 2.2 is a special case of the above algorithm, when identity transformation $A' := A$ is used in phase 2 and $d = 2$ in (ii) of phase 3.

7 Construction of injective knapsacks from smaller knapsacks

With random experiments we can fairly easily find injective knapsacks with up to, say 12 or 13 elements. Knapsacks of such sizes can also be deciphered easily by precomputed tables of all their subset sums.

In Section 7.1 we discuss conditions on obtaining easily decipherable knapsacks by iteratively joining small knapsacks to a large knapsack. The idea generalizes the algorithm of Desmedt et al. [10] that was reviewed in Section 6.3. In that algorithm new *elements* are iteratively joined to the large knapsack. We will present two types of realizations of our idea, but neither one gives anything new when applied to the restricted situation of that algorithm. One simple example of the realizations is presented in Section 7.2, and the more complicated situations will be studied in Chapters 8 and 9.

7.1 A general framework

Assume that we have a collection of small injective knapsacks B_i, $i = 1, 2, \ldots, r$.

Suppose also that A_1 is an injective knapsack that is easy to decipher. It may have been generated by superincrease or with some of the constructions presented in Chapter 6 and in this case it can also be quite large. On the other hand it could be like one of the B_i.

The idea is to construct a large knapsack A_{r+1} in the following way:

$$A_{i+1} = f_i(A_i) \cup g_i(B_i), \qquad i = 1, \ldots, r, \tag{9}$$

where for example $f_i(A_i)$ means the knapsack resulting from application of the integer function f_i to each element of A_i. The functions f_i and g_i should be such that the knapsack A_{i+1} is injective and easy to decipher. This should be possible to deduce from the corresponding properties of A_i and the fact that B_i is small.

Concerning the functions f_i and g_i it is quite obvious that we can do more complicated things only with the small knapsack B_i and not with A_i, which is typically quite large already when i is 4 or 5.

Thus, when deciphering a subset sum of A_{i+1} we should first be able to decide, which elements from $g_i(B_i)$ are involved in it. Then, after subtracting these, we obtain a subset sum of $f_i(A_i)$, where we must be able to go back to A_i and its constituents. That is, similar requirements concern f_i as the trapdoor transformation f that we discussed in Section 4.1.

Let us now state a general property concerning the injectivity of the union of two injective knapsacks. The same idea was present already in the search algorithm on page 18. We first introduce a new notation.

Notation 7.1 Denote by D_A the set of all positive pairwise differences of subset sums of a knapsack A.

Since 0 is a subset sum, the set D_A contains especially all the subset sums of A, except 0.

Theorem 7.2 Assume that A and B are injective knapsacks. For the union $A \cup B$ to be injective it is sufficient and necessary that $D_A \cap D_B = \emptyset$.

Proof. Assume first that $D_A \cap D_B = \emptyset$. Take two coinciding subset sums of $A \cup B$, and denote them by $\sum_1$ and $\sum_2$. For $i = 1, 2$ denote by $\sum_{iA}$ and $\sum_{iB}$ those parts of $\sum_i$ which stem from the sets A and B correspondingly. Thus

$$\sum_{1A} + \sum_{1B} = \quad \sum_1 = \sum_2 \quad = \sum_{2A} + \sum_{2B}. \tag{10}$$

Without restriction we may assume that $\sum_{1A} \geq \sum_{2A}$. We then have

$$\sum_{1A} - \sum_{2A} = \sum_{2B} - \sum_{1B}, \tag{11}$$

where both sides are non-negative. They cannot be positive, because this would contradict our assumption that $D_A \cap D_B = \emptyset$. Thus $\sum_{1A} = \sum_{2A}$ and $\sum_{1B} = \sum_{2B}$, which by injectivity of the knapsacks A and B gives that the sums $\sum_1$ and $\sum_2$ have the same addends. This proves injectivity of $A \cup B$.

Assume then that $A \cup B$ is injective. A common element in the sets D_A and D_B would give us an equation (11) with both sides positive. This would lead to an equation (10) with coinciding but different subset sums of $A \cup B$, which would be a contradiction. Thus $D_A \cap D_B = \emptyset$. □

Example. Let $A = \{9, 20, 26\}$ and $B = \{13, 18, 23\}$. The subset sums of A and B are

$$S_A = \{0, 9, 20, 26, 29, 35, 46, 55\} \quad \text{and} \quad S_B = \{0, 13, 18, 23, 31, 36, 41, 54\}$$

and hence

$$\begin{aligned} D_A &= \{3, 6, 9, 11, 15, 17, 20, 26, 29, 35, 37, 46, 55\} \\ D_B &= \{5, 8, 10, 13, 18, 23, 28, 31, 36, 41, 54\} \end{aligned}$$

Since $D_A \cap D_B = \emptyset$ we know that $A \cup B$ is injective. □

There are two basic ways to provide injectivity for the union. The first is based on considerations of the magnitude of the elements. The second method deals with residuals of the elements with respect to some modulus. If we choose one method to prove injectivity, it is difficult to imagine that we could use some other method to decipher. This does not mean that we could not change the method for different i in the equation (9).

Magnitude

If the knapsack A is large it is not practical or even possible to write down the set D_A. Since our basic goal is to construct relatively dense knapsacks, we must have $\rho(A)$ at least about $\rho_2(|A|)$. In such a knapsack already the subset sums form quite a dense set in the sense that there are not large gaps between successive members of the set. It is obviously very difficult to find gaps between successive members of D_A, except at the high end of this set. For example the two largest elements of D_A are $s_A - \min(A)$ and s_A.

By *deductions concerning magnitude* we mean that we are only able to restrict the subset sums of a knapsack and their differences inside or outside one single interval. In our case of joining two knapsacks A and B we thus need such an interval $[d_1, d_2]$ that

$$D_A \subseteq [d_1, d_2] \quad \text{and} \quad D_B \cap [d_1, d_2] = \emptyset. \tag{12}$$

The minimal d_2 obviously equals s_A. Note that we may interchange the names of A and B if needed — A is not necessarily a large knapsack and B a small one.

Case 1. Assume first that $d_1 = 1$. This means that all elements of D_B are greater than s_A which, on the other hand is greater than $2^{|A|} - 2$.

Suppose A is the small knapsack to be joined to the large knapsack B. What was said above about the gaps implies now that to satisfy $\min(D_B) > s_A$ we must in practice have a common factor $d > s_A$ in the elements of B. Since $\frac{1}{d}B$ is an injective knapsack we know that $\max(B)$ is about $2^{|A|}$ times larger than the injectivity of B would need. This effect determines the development of density, because the small knapsack A is joined below the elements of B. Multiplication by 2^r where r is the number of new elements amounts to roughly a similar development as with superincrease.

Suppose then that A is the large knapsack. If B is small enough we can satisfy $\min(D_B) > s_A$ without a common factor $d > s_A$ in B. In any case the elements of B must be larger than s_A and differ from each other by amounts that are larger than s_A. If there is not a common factor it seems that these gaps must amount to at least such growth of $\max(B)$ that corresponds to superincrease. The elements of B need not superincrease however, for an example see Section 7.2. If there is a common factor, the development is similar to the case where B is the larger knapsack.

Case 2. Assume now that $d_1 > 1$. With the same arguments as above we deduce that A is (at least roughly) the same as d_1 times another knapsack. What this implies to the density depends on how great a part of D_B lies below d_1. If the whole of D_B is there, then $d_1 > s_B$ and we have a similar result as above. This is the consequence also if the whole knapsack B lies above d_2, even if some small differences of it were below d_1.

If there are some elements of B below d_1, then d_1 must be larger than their sum and again, joining these small new elements cause superincreasing growth proportional to their number. The rest of B will do the same for their part.

Residuals

If residuals with respect to some modulus m are used to ensure injectivity and enable decipherment of $A \cup B$, then these deductions cannot allow very much freedom to the residuals of the larger knapsack, suppose it is A. Instead the residuals must be used to distinguish and decipher that part of a subset sum that stems from elements of B. Some elements of A not being 0 modulo m would restrict considerably the possible values for residuals that are used for this partial decipherment. Decryption would then have to happen with respect to some divisor of m. This divisor will then divide all the elements of A.

We may thus assume that A has been obtained from another knapsack by multipying it by m. In order to distinguish all subset sums of B modulo m we must have $m \geq 2^{|B|}$. Multiplication by such an m means that the largest element of the knapsack grows as if we had joined the $|B|$ new elements in the superincreasing fashion.

Other principles?

We do not know any general method, not based on residuals, of inserting new elements between the elements of an existing knapsack, so that the new knapsack would be injective. We can introduce new elements to the Merkle-Hellman trapdoor knapsack if the modulus is large enough to allow insertion of new primes in the multiplicative knapsack. But this method only works for this type of knapsacks. As it was discussed above, deductions based on magnitude seem to be too complicated if they interfere with the inside of the set D_A of a large knapsack A.

7.2 "Three elements"

As a simple construction of type (9) based on magnitude we present the following lemma. It can be used to construct non-superincreasing knapsacks of any size (≥ 2) by taking the initial set A for example one of the following: $\emptyset$, $\{1\}$ or $\{1,2\}$. The empty set is not a knapsack but that does not matter in the construction. Naturally $s_\emptyset = 0$.

Theorem 7.3 Assume A is an injective knapsack that is easy to decipher. Choose such integers a, b and c that

$$a > 2s_A, \qquad b > a + s_A, \qquad c > b + s_A, \qquad a + b > c + s_A.$$

Then $A \cup \{a, b, c\}$ is injective, easy to decipher and not superincreasing.

Proof. To prove the injectivity assume that s is a subset sum of the new knapsack. The following table shows all the different cases concerning the presence of the new elements in the sum. Using the defining inequalities we

see that the intervals in which s then belongs are disjoint (the last column).

present	s lies within the interval		
none	$[0, s_A]$		
a	$[a, a+s_A]$	$\subseteq$	$(2s_A, a+s_A]$
b	$[b, b+s_A]$	$\subseteq$	$(a+s_A, b+s_A]$
c	$[c, c+s_A]$	$\subseteq$	$(b+s_A, c+s_A]$
a, b	$[a+b, a+b+s_A]$	$\subseteq$	$(c+s_A, a+b+s_A]$
a, c	$[a+c, a+c+s_A]$	$\subseteq$	$(a+b+s_A, a+c+s_A]$
b, c	$[b+c, b+c+s_A]$	$\subseteq$	$(a+c+s_A, b+c+s_A]$
a, b, c	$[a+b+c, a+b+c+s_A]$	$\subseteq$	$(b+c+2s_A, a+b+c+s_A]$

Reading the middle column from bottom up we see an obvious algorithm for first judging which of the new elements are present in the sum and then subtracting them.

Since $a < b < c$, but $c < a + b - s_A$, we see that the knapsack is not superincreasing. □

Remark. We could only require for a that $a > s_A$, but from the other inequalities it follows that $a > 2s_A$.

Joining three new elements to A in the superincreasing fashion would require the third new element to be greater than $4s_A$. Now the construction requires that $c > 4s_A$ which means that the density of the new knapsack is comparable to the superincreasing construction. However, the sum of the new knapsack is now greater than $10s_A$ which is more than $8s_A$, which is a lower bound for the fourth new element in the superincreasing continuation of A.

Example. Supposing that $s_A = 99$ we can take $a = 200$, $b = 300$, $c = 400$.

8 Constructions based on residuals

8.1 Congruential knapsacks

Lemma 8.1 Assume A and B are knapsacks that are injective modulo the integers m_A and m_A respectively. Then the knapsack $m_B A \cup B$ is injective modulo $m_A m_B$.

Proof. Assume that two subset sums $\sum_1$ and $\sum_2$ of $mA \cup B$ are congruent modulo $m_A m_B$. For $i = 1, 2$ denote by $m\sum_{iA}$ and $\sum_{iB}$ those parts of $\sum_i$ which stem from the sets mA and B correspondingly. We thus have

$$m_B \sum_{1A} + \sum_{1B} \equiv m_B \sum_{2A} + \sum_{2B} \pmod{m_A m_B}.$$

This gives

$$\sum_{1B} \equiv \sum_{2B} \pmod{m_B},$$

which implies by modular injectivity of B that the addends in $\sum_{1B}$ and $\sum_{2B}$ are the same. Hence $\sum_{1B} = \sum_{2B}$ and we have

$$m_B \sum_{1A} \equiv m_B \sum_{2A} \pmod{m_A m_B},$$

which gives

$$\sum_{1A} \equiv \sum_{2A} \pmod{m_A},$$

and by modular injectivity of A we have that also the addends in $\sum_{1A}$ and $\sum_{2A}$ are the same. □

Theorem 8.2 Assume that the knapsack $B = \{b_1, \ldots, b_k\}$ is injective modulo an integer m. Assume also that the modular subset sums are easy to decipher.

If the knapsack A is injective and easy to decipher then the same is true of the knapsack $mA \cup B'$, where $B' = \{\beta_1, \ldots, \beta_k\}$ and $\beta_i \equiv b_i \pmod{m}$, $i = 1, \ldots, k$.

Proof. From Theorem 5.3 (iii) we know that A is injective modulo $s_A + 1$. Clearly B' is injective modulo m. We can infer the injectivity of $mA \cup B'$ by applying Lemma 8.1 with $m_A = s_A + 1$ and $m_B = m$, together with Theorem 5.3 (i).

Without using the lemma we could see the injectivity from the following algorithm which clearly always gives a unique decipherment. Furthermore the algorithm shows that it is easy to find the decipherment. Given a subset sum s of $mA \cup B'$:

1. Find the subset of B giving a sum congruent to s modulo m.
2. Subtract the corresponding elements of B' from s.
3. Divide the result by m and decipher according to A. □

Definition 8.3 Following the idea of Section 7.1 perform the construction of the theorem iteratively with an initial knapsack A_1 and the knapsacks B_i, B'_i and $A_{i+1} = m_i A_i \cup B'_i$, $i = 1, \ldots, r$. The result is

$$A_{r+1} = m_r \Big(m_{r-1} \big(\ldots m_2 (m_1 A_1 \cup B'_1) \cup B'_2 \quad \ldots \quad \big) \cup B'_{r-1} \Big) \cup B'_r \qquad (13)$$

and it is called a **congruential knapsack**. In addition to this notation we denote in the sequel $A = A_{r+1}$, $n = |A|$.

Modular decipherment of an integer s refers to finding such subsets of a knapsack that the subset sums are congruent to s modulo a given integer.

Applying Lemma 8.1 inductively we obtain the following theorem.

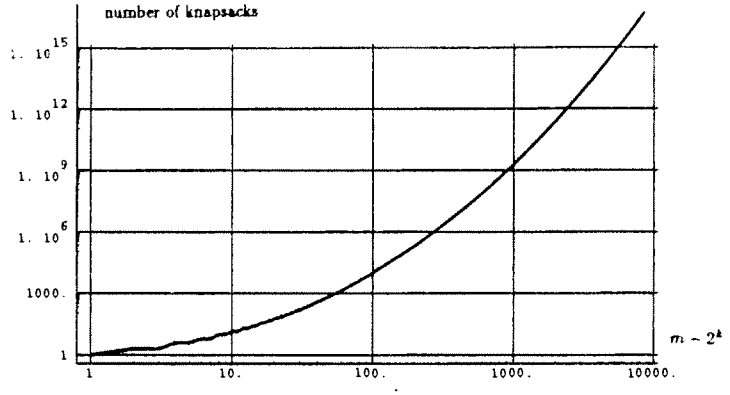
Theorem 8.4 If $A_1 = \emptyset$, then the knapsack A_{r+1} in (13) is injective modulo the product $m_1 \cdots m_r$.

Constructing the modular knapsacks B_i

Random search can be efficient only if the knapsack size k is not very large. Even then it can be difficult to find injective knapsacks modulo an m that does not exceed 2^k very much. Recall that a knapsack of size k can be injective modulo m only if $m \geq 2^k$. Suppose we have already found one knapsack.

Then we can use modular multiplication to produce new modularly injective knapsacks according to Theorem 5.7. Usually this gives a different knapsack for each multiplier, which means that there is a good choice of knapsacks.

If random search has not produced any knapsacks that are injective modulo m, one can always use an *injective* knapsack for which m is a strong modulus. The basic superincreasing knapsack $B = \{1, 2, 4, \ldots, 2^{k-1}\}$ will always be such, and if $m > 2^k$, also some other superincreasing knapsacks are possible. We computed the number of these as a function of $m - 2^k$ and plotted it in the log-log figure below.



Notice that if the modulus is 2^k and B is used, the result is the same as when applying k times Lemma 2.2.

One can also try with such injective knapsacks, whose sum exceeds the modulus, but then the modular injectivity must be checked. This will obviously succeed considerably more often than purely random search, but is not possible if k is very large.

Distributing the elements

Suppose that we always insert the elements of B_i' between $m_i \min(A_i)$ and $m_i \max(A_i)$. Choosing a b' congruent to a $b \in B_i$ modulo m amounts to adding the decimal part of $\frac{b}{m}$ to one of the integers $\min(A_i), \ldots, \max(A_i) - 1$. We see that there are $\max(A_i) - \min(A_i)$ choices.

Denote $\delta = \max(A_1) - \min(A_1)$. Then the number of choices for each element of B_i equals δ for $i = 1$ and $m_{i-1} m_{i-2} \cdots m_1 \delta$ for $i = 2, \ldots, r$.

The relative position of every new b'-element remains the same in all subsequent knapsacks A_i. Together with the degrees of freedom that every b' has, this suggests that we can distribute the elements of the final knapsack in roughly any way we want, with the exception of the elements stemming from A_1.

Density

Assume that A_1 is an injective knapsack of size n_1 and that $\max(A_1) \leq \alpha 2^{n_1}$. As we saw in Chapter 3, α can easily be for example 0.4, when $n_1 \geq 5$. Denote

$k_i = |B_i|$ and $m_i = (1+\varepsilon_i)2^{k_i}$ for $i = 1, \ldots, r$. Suppose the ε_i are small non-negative numbers, for example around 0.05.

Suppose we never insert the elements of B_i' above $m_i \max(A_i)$. Then

$$\max(A) = m_1 \cdots m_r \max(A_1) \leq \alpha 2^n \prod_{i=1}^{r}(1+\varepsilon_i)$$

and the density of A is

$$\rho(A) \geq \frac{n}{n + \log_2 \alpha + \sum_{i=1}^{r} \log_2(1+\varepsilon_i)} \geq \frac{n}{n + \log_2 \alpha + \frac{1}{\ln 2}\sum_{i=1}^{r} \varepsilon_i}, \qquad (14)$$

where the last inequality follows from

$$\log_2(1+x) = \frac{\ln(1+x)}{\ln 2} \leq \frac{x}{\ln 2}, \qquad x > -1.$$

If we want to achieve a certain density ρ we must choose the parameters so that

$$\sum_{i=1}^{r} \varepsilon_i \leq \ln 2(\frac{n}{\rho} - n) - \ln \alpha. \qquad (15)$$

For example $\rho = \rho_2(n) = \frac{n}{n-1}$ requires $\sum \varepsilon_i \leq 0.223$ when α is 0.4, regardless of n. It can be seen that ρ_2 can only be achieved if $\alpha < 0.5$.

For fixed densities larger than 1 the requirement becomes more and more strict as n grows. For densities below 1 the effect is opposite. For the density 0.95, with $\alpha = 0.4$ the bound is 1.61 when $n = 100$, but only 8.21 when $n = 200$.

The trapdoor transformation

When we perform a k-modular multiplication on the knapsack A to obtain a knapsack A', we usually suffer a loss of density, unless $k = 1$. Assume that we use a modulus $m = \kappa \max(A)$. Then the knapsack A' is likely to satisfy $\max(A') \approx \kappa \max(A)$ and we obtain the density

$$\rho(A') \approx \frac{n}{n + \log_2 \kappa\alpha + \frac{1}{\ln 2}\sum_{i=1}^{r} \varepsilon_i}$$

For a specific density ρ we must now have

$$\sum_{i=1}^{r} \varepsilon_i \leq \ln 2(\frac{n}{\rho} - n) - \ln \kappa\alpha. \qquad (16)$$

It is not likely that we could achieve the density ρ_2 if we use $k > 1$. This follows from the fact that the second largest element in the dense knapsack A_1 is relatively near the largest element. In all dense injective knapsacks with $\alpha \leq 0.4$ that we know of, a 2-modulus would have $\kappa > 1\frac{2}{3}$. To achieve ρ_2 would require $\alpha < \frac{0.5}{\kappa} = 0.3$. We do not know any injective knapsacks that would be this dense. As we mentioned in the end of Chapter 3, we believe that there are such knapsacks. The congruential knapsack would obviously benefit

if such knapsacks were found, although for small k their k-moduli would most likely be quite near $k\max(A)$.

Suppose $\alpha = 0.4$. Setting $\sum \varepsilon_i \approx 0$ in (16) shows that it is possible to achieve the density 1 if the modulus is at most 6.8 times $\max(A)$. This estimate does not depend on n.

If we want to use an $\lceil \frac{n}{2} \rceil$-modulus for a knapsack A where the elements are approximately evenly dispersed, we have $\kappa \approx \frac{3n}{8}$ and

$$\rho(A') \approx \frac{n}{n + \log_2 \frac{3n\alpha}{8} + \frac{1}{\ln 2} \sum_{i=1}^{r} \varepsilon_i}. \tag{17}$$

For example when $n = 200$, $\alpha = 0.4$ and $\sum \varepsilon_i = 1$ this gives the (hopefully) non-critical density of 0.969.

8.2 Obvious attacks and countermeasures

As such the congruential knapsack reveals its structure quite easily. Assume that we have the patience to compute all the pairwise greatest common divisors. Let m be the largest of these. Then compute the greatest common divisor of m with each element (if not known already).

The elements will fall (at least roughly) into $r + 1$ disjoint sets where the gcd with m in the ith set is of the form $m_i m_{i+1} \cdots m_{r+1}$, $i = 1, \ldots, r + 1$, where $m_{r+1} = 1$ and $m_1 m_2 \cdots m_r = m$.

The first set will be a candidate for the initial knapsack A_1 times m. For $i = 2, \ldots, r + 1$ the ith set will be a candidate for $m_i m_{i+1} \cdots m_{r+1} B'_{i-1}$. This division and the value of r may differ slightly from the original ones. We can try to adjust the division by moving some elements to sets with higher indices i (assuming that they accidentally have too large common divisors with m). This may be necessary if in some sets there are far too many elements for modular injectivity.

We may always consider a couple of elements separately, in the spirit of random elements of page 32. Even more extra elements can be handled if we use the temperate approach of the next section.

It seems that we can discover a congruential structure that enables decipherment, even if it does not coincide with the original one. It also seems that the search for m and the subsequent classification can be conducted so that one does not need to compute all the pairwise gcd's.

If a constant has been subtracted from each element, the above analysis of the knapsack structure becomes only slightly more difficult. Choose one element and subtract it from all the other elements. If it happened to stem from the initial knapsack, then the greatest common divisor method will yield the same result as above. If it did not, then trying with other elements will gradually lead to better classification as often as the element is "nearer" to the initial knapsack.

If the subtrahend remains unknown it may be impossible to use the congruences for decipherment. However, the discovered large divisors of the original

elements pose quite a lot of restrictions on the subtrahend. It may also be revealing to know that the elements of the knapsacks B_i have not been very large.

Suppose now that the congruential knapsack A has been multiplied modularly to give $A' = wA \pmod m$. The elements of A' are of the form $a'_i = wa_i + u_i m$, where all the u_i are likely to be different if w is not very small in comparison to m. The factorizations of the a'_i are likely to be completely different from those of the a_i and this is true also for the differences $a'_i - a'_j$.

Attacks exploiting the fact that many of the quotients $\frac{a_i}{a'_i m}$ are very small (cf. p. 24) are likely to have difficulties if none of the a_i are much smaller than the others. This can be achieved for example by always inserting the elements of B'_i between $m_i \min(A_i)$ and $m_i \max(A_i)$. Furthermore the modulus m and hence the elements a'_i can be relatively small, if m is a k-modulus with a fairly small k. Such a non-injective knapsack can be kept very temperate, again with a suitable dispersion of elements.

As we have seen the modularly multiplied congruential knapsack can still have a fairly high density, at least if the modulus is allowed to be "too small". To achieve high densities, the sum $\sum_{i=1}^{r} \varepsilon_i$ has to be very near zero. The smaller it must be, the fewer alternatives there are for the moduli m_i. This is the case also when r grows. There are two effects that cause this. Firstly the bound of $\sum \varepsilon_i$ must be shared with more moduli. Secondly, the knapsack sizes k_i become smaller, and with them decreases also the number of integers in the ranges $[2^{k_i}, \ldots, (1+\varepsilon_i)2^{k_i}]$. For example $\varepsilon = 0.0005$ would leave only $m = 1024$ for $k = 10$, whereas for $k = 20$ there are still 525 alternatives.

Further countermeasures. After the first one or two rounds of the construction algorithm the number of choices for the new elements is large enough also for other purposes than providing a suitable dispersion. They can be chosen to have large common factors with several of the previous moduli. This can make it more difficult to find small enough sets with the same greatest common divisor, even if this approach somehow were possible in the modularly multiplied knapsack.

For this end it may be wise to choose the moduli so that they are not primes and several of them have common factors. This can be achieved by using fairly large moduli and hence also large knapsacks. As we mentioned above, this also gives more variety for the moduli without decreasing the density. On the other hand, there are fewer moduli when the knapsacks are larger.

A different trapdoor. We can obviously scramble the congruential structure quite thoroughly by using the "disconnecting" modification of the Graham-Shamir knapsack (see p. 49). The problem of density can be cured with a small k-modulus. We will need some non-injectivity also here, but the versatility of the congruential knapsack offers also an additional remedy.

Recall that in this modification we insert q zeros after the p lowest order bits in the elements of a knapsack A. Usually we have $q = \lceil \log_2 n \rceil - 1$ or this minus 1. When $128 < n \leq 256$, this is 7 or 6. Because of the large number of alternatives for the new elements it should be possible to choose all or almost all of them so that there are 5 or 6 zeros in the high end of the p lowest order bits in the final knapsack A. This means that the sum of the $\lceil \frac{n}{2} \rceil$ largest of the p-bit numbers is determined by those numbers stemming from the initial knapsack A_1. If $|A_1| \leq 10$, then it suffices to take $q = 3$, or with bad luck $q = 4$. A smaller value of q means that we can use a k-modulus with a larger k without decreasing the density too much.

8.3 A temperate modification

In general, if x extra elements are inserted in an injective knapsack, a complete decipherment requires 2^x trials. We will show how the effort can be considerably smaller than this, when we insert the x elements in knapsacks B_i of a congruential knapsack, one in each. Let us first investigate the situation from the viewpoint of one knapsack $B = B_i$ of size k, $k > 1$.

Assume we have such an m that $2^{k-1} < m < 2^k$. Construct a knapsack B_0 of size $k - 1$ that is injective modulo m and easy to decipher modularly. Choose a positive integer β not congruent modulo m to any element of B_0. Denote $B = B_0 \cup \{\beta\}$. The knapsack B is not injective modulo m, but every integer has at most two modular decipherments according to B (i.e. modulo m). For $i = 0, 1, 2$ denote by D_i the number of integers between 0 and $m - 1$ that have i modular decipherments. We have

$$D_0 + D_1 + D_2 = m \quad \text{and} \quad D_1 + 2D_2 = 2^k.$$

Taking into account that all the D_i are non-negative we can deduce

$$2^k - m \leq D_2 \leq 2^{k-1} \quad \text{and} \quad \begin{cases} D_1 = 2^k - 2D_2 \\ D_0 = m + D_2 - 2^k \end{cases}.$$

The upper bound of D_2 would occur, if all subset sums with β were congruent to those without β (a case which we have not wittnessed). The lower bound occurs, when there are m modularly distinct subset sums. According to our experiments the lower bound can usually be achieved, but not always. The knapsack $B = \{4, 5, 11\}$ is an example of this. It is injective modulo 14, but there are never more than 13 modularly distinct subset sums of $B \cup \{\beta\}$. The value 13 is reached only when $\beta \equiv 6$ or 8.

The ratio $p_i = \frac{D_i}{m}$ gives the probability that a given integer has i modular decipherments. If the integer is known to be congruent to a subset sum of B, then the probability of i decipherments is $q_i = \frac{D_i}{D_1 + D_2}$ for $i = 1, 2$, and 0 for $i = 0$.

The expected number of modular decipherments of an arbitrary integer is

$$\pi = 1p_1 + 2p_2 = \frac{1}{m}(D_1 + 2D_2) = \frac{2^k}{m}, \tag{18}$$

which is independent of the choice of B. The expected number of modular decipherments of a subset sum of B is

$$\omega = \frac{D_1}{D_1 + D_2} + \frac{2D_2}{D_1 + D_2} = \frac{2^k}{m - D_0}. \tag{19}$$

This shows the obvious fact that we should minimize D_0, or D_2, when constructing B.

If k is not too large, we can compute the subset sums of B_0 and choose such an element β that D_0 is as small as possible. As a result of these computations we will have a table of the modular subset sums, which can be used during the deciphering process.

If k is large, we do not know how to adjust the value of β. The decipherment(s) of an integer s can be found by attempting to decipher both s and $s - \beta$ according to B_0.

An average for D_2

Consider all possible knapsacks B_0 and assume that the extra elements β generate random collections of 2^{k-1} modular integers when added to the subset sums of B_0.

Denote $u = 2^{k-1}$ and let $0 \leq j \leq u$. The equation $D_2 = j$ means that j of the new sums coincide modularly with the old ones. By the assumption of randomness the probability of this, $P(D_2 = j)$, can be computed in the following way.

Suppose we have m slots and among them a fixed subset of u slots. We allocate u tokens to the slots randomly, at most one in each. The probability that the fixed subset gets exactly j of the tokens is the same as $P(D_2 = j)$. Thus we can write

$$P(D_2{=}j) = \frac{\binom{u}{j}\binom{m-u}{u-j}}{\binom{m}{u}},$$

using the convention that $\binom{x}{y} = 0$ if $y > x$. The probability is 0, when $u - j > m - u$, that is, when $j < 2u - m$. This is the same bound for D_2 as before.

The average of D_2 can now be computed directly from the hypergeometric distribution. We get

$$\hat{D}_2 = \frac{u^2}{m} = \frac{2^{2k-2}}{m}$$

Our assumption on the randomness may be true only approximately because the modular subset sums of B_0 are not totally randomly distributed. Their distribution is however more random than that of the non-modular subset sums. We believe that $\hat{D}_2$ gives a good impression of what happens on average, when we construct the knapsack B without making any adjustments of the extra element β.

The average deciphering effort

We already computed the average number of modular decipherments according to the knapsack B. The average is π for arbitrary integers and ω for subset sums. Always $\pi \geq \omega$ and the two are equal if D_2 is at its (a priori) minimum.

Suppose now that we have a generalized congruential knapsack A as in equation (13), but with the r knapsacks B_i being like B above. Denote the decipherment parameters of B_i by π_i and ω_i.

Consider the deciphering process of an arbitrary integer. We can proceed as in the proof of Theorem 8.2, but occasionally the process is divided into two branches, and sometimes a branch must be cut off because the B_i in question provides no decipherment.

If we have x branches to decipher with the knapsack B_i, then we expect to get $\pi_i x$ decipherments and thus we give $\pi_i x$ branches to decipher with the next knapsack B_{i-1}. We infer that the initial knapsack A_1 is expected to get $\prod_{i=1}^{r} \pi_i$ branches to be processed. The probability, that an arbitrary integer in the range 0 to s_{A_1} is a subset sum of the dense knapsack A_1, can be as low as 0.5 (cf. Chapter 3). So we expect that A_1 will cut off several of the branches. Actually, there can be some cutting off already before we come back to A_1. This will happen if we only forward such branches that belong to the range of the next knapsack A_i. Our experiments show that this indeed reduces the effort, but has an effect only quite near the end.

What was said above applies to an arbitrary integer that we try to decipher. Of course it may turn out that there is no decipherment at all. On the other hand, we only have the trivial upper bound 2^r for the number of decipherments. Intuitively it seems that the modular non-injectivity of the B_i only leads to moderate non-injectivity of A.

Consider now a subset sum of A. We know that for every knapsack B_i at least one of the branches that it gets corresponds to a subset sum. That is, the number of decipherments for this branch is expected to be ω. It is difficult to infer the nature of the other branches. Supposing that they all correspond to subset sums, we get similarly as above, that the average number of decipherments before A_1 equals $\prod_{i=1}^{r} \omega_i$.

In any case we know that for a subset sum of A there are on average between $\prod_{i=1}^{r} \pi_i$ and $\prod_{i=1}^{r} \omega_i$ partial decipherments that must be completed with A_1.

Connections to the density

The main motivation of non-injectivity is to increase density. The formula (14) is valid also when some or all of the ε_i are negative. This gives a possibility to relate the density and the number of decipherments.

Consider again the knapsack B and its parameters. The choice of m implies that $m = (1+\varepsilon)2^k$, where $-0.5 < \varepsilon < 0$. Using this representation for m gives

$$\pi = \frac{1}{1+\varepsilon}$$

and

$$\omega = \begin{cases} \pi & \text{when } D_2 = 2^k - m = -2^k\varepsilon \quad \text{(the minimum)} \\ 1 + \frac{1}{3+4\varepsilon} & \text{when } D_2 = \hat{D}_2 = \frac{2^{k-2}}{1+\varepsilon} \end{cases}.$$

We see especially that for an average knapsack B the parameter ω is somewhat larger than 1.333. If we have for example 10 knapsacks with $\omega = 1.4$ we know that on average there are no more than 29 decipherments.

Assume in the sequel that we deal either with the minimal or average case of D_2. Denote ω by $\hat{\omega}$, when it corresponds to $\hat{D}_2$.

If the absolute values of the ε_i are not very small, the latter inequality in (14) gives too pessimistic estimates. To avoid this let us use the original estimate

$$\rho(A) \geq \frac{n}{n + \log_2 \alpha + \sum_{i=1}^{r} \log_2(1+\varepsilon_i)}.$$

To increase the density we should have $\sum \log_2(1+\varepsilon_i)$ as far below zero as possible. On the other hand we should have either $\prod \frac{1}{1+\varepsilon_i}$ or $\prod(1 + \frac{1}{3+4\varepsilon_i})$ as small as possible. Since

$$\log_2 \prod \frac{1}{1+\varepsilon_i} = -\sum \log_2(1+\varepsilon_i),$$

we see that the density estimate is in one-to-one correspondence to the expected number of decipherments, under the assumption that either the knapsacks B_i are chosen with minimal D_2, or we are deciphering random integers.

It is easy to see by using a Lagrange multiplier that in case of $\prod(1 + \frac{1}{3+4\varepsilon_i})$ the best result is obtained, if all the ε_i are equal. In practice the ε_i (if not 0) must be at least slightly different, if we do not want that all the moduli m_i and the knapsack sizes $k_i = |B_i|$ are the same.

Assume now that $\varepsilon_i = \varepsilon$ for all i. We should choose ε and r so as to minimize both $\sum \log_2(1+\varepsilon_i)$ and the bound $\hat{\omega}^r = \left(1 + \frac{1}{3+4\varepsilon}\right)^r$. Fixing a value $s = r\log_2(1+\varepsilon)$ gives $\varepsilon = 2^{s/r} - 1$. As a function of r we have

$$\hat{\omega}^r = \left(1 + \frac{1}{4 \cdot 2^{s/r} - 1}\right)^r = \left(\frac{4}{4 - 2^{-s/r}}\right)^r, \qquad r > -s.$$

The bound for r follows from the condition $\varepsilon > -0.5$. The function appears to be increasing for all negative s, which we induced from plots of $\hat{\omega}^r$ against r. We do not enter into any further details, but illustrate the situation with an example.

Example. Suppose we do not want to cause extra ambiguity by too small a modulus and hence take an $\frac{n}{2}$-modulus.

Suppose we have an initial knapsack A_1 with 11 elements and the parameter $\alpha = 0.4$. Construct a generalized congruential knapsack A with 200 approximately evenly dispersed elements. A computation analogous to the inequality (17) gives for the transformed knapsack A'

$$\rho(A') \approx \frac{200}{204.907 + \sum_{i=1}^{r} \log_2(1+\varepsilon_i)},$$

which is actually an approximate lower bound. If $\sum \log(1+\varepsilon_i) = -6$, the formula gives 1.00550, which is slightly larger than $\rho_2(200) = \frac{200}{199} = 1.00503$.

Suppose we construct r knapsacks B_i of the same size $k = \frac{188}{r}$ with respect to moduli $m_i \approx (1+\varepsilon)2^k$, where $r\log_2(1+\varepsilon) = -6$. The following table lists the possible values of r and gives π^r and $\hat{\omega}^r$ as if all the m_i were exactly $(1+\varepsilon)2^k$.

r	k	ε	π^r	$\hat{\omega}^r$
7	27	−0.4480	64	68
9	21	−0.3700	64	95
21	9	−0.1797	64	2066
27	7	−0.1428	64	11041
63	3	−0.0639	64	$3.2\cdot10^8$

The values do not change radically if we attempt more modest densities. For example the density 1 with $r = 27$ gives $\varepsilon = -0.1183$, $\pi^r = 30$ and $\hat{\omega}^r = 8132$. On the other hand, the density $1 + \frac{\log_2 n/2}{n}$ cited on page 13 is 1.03322 for $n = 200$. To achieve this we need $\sum \log_2(1+\varepsilon_i) = -11.34$, which gives $\pi^r = 2587$. Notice that now the values 7 and 9 are not possible for r.

For large k, like 21, we cannot minimize D_2 and thus we may have to cope with $\hat{\omega}^r$, but as we see, it is not very large then. For small k we can optimize, and it seems to be also necessary because then $\hat{\omega}^r$ can be prohibitively large. Notice however that very small values of k can cause difficulties, because the modulus must be an integer. For example when $k = 3$, there is no modulus m near $(1+\varepsilon)2^k = (1-0.0639)8$. The nearest integer is 7 which would give $\varepsilon = -0.125$ and $\pi^r = 4503$.

9 Constructions based on magnitude

We present in Sections 9.1 and 9.2 two different ways to realize the idea of Section 7.1. Section 9.3 slightly generalizes the latter construction. All these methods will typically produce non-superincreasing knapsacks. According to the discussion in Section 7.1, these constructions cannot however lead to good results in density.

The presentation is oriented towards proving injectivity and does not refer to the ease of decipherment. All the systems are however of type $A \cup B$ where the elements of B are larger than s_A. If B is not very large and A is easy to decipher, then according to Lemma 5.14 and the discussion after it the decipherment is possible with a reasonable number of trials.

9.1 Union of a multiplied and a shifted knapsack

We will construct from two injective knapsacks A and B a new injective knapsack of the form $wA \cup (d+B)$. We will start with a new notation and a couple of simple lemmas.

Notation 9.1 Concerning a knapsack A denote by Δ_i the largest difference between two subset sums where the number of addends differ by i ($i = 0, 1, \ldots, |A|$).

Example. For the knapsack $\{5, 7, 10, 13, 14\}$ we have the following Δ_i

i	0	1	2	3	4	5
Δ_i	15	25	32	39	44	49

Notice that Δ_5 equals the sum of the knapsack.

Lemma 9.2 (i) If $|A| = 2k$, then Δ_i is obtained when the sum of the $k - \lceil \frac{i}{2} \rceil$ smallest elements is subtracted from the sum of $k + \lfloor \frac{i}{2} \rfloor$ largest elements.

If $|A| = 2k + 1$, then Δ_i is obtained when the sum of the $k - \lfloor \frac{i}{2} \rfloor$ smallest elements is subtracted from the sum of $k + \lceil \frac{i}{2} \rceil$ largest elements.

(ii) If $|A| \geq 3$, then $\Delta_1 > \max(A)$ and $\Delta_i < i\,\Delta_1$, for $i = 2, \ldots, |A|$.

Proof. (i) Assume $0 \leq i \leq |A|$ and let $\sum_1$ and $\sum_2$ be two subset sums in which the number of addends differ by the amount i. Denote

$$\Delta = \sum_1 - \sum_2,$$

and apply the following obvious principles to maximize Δ.

- We may remove all common addends from the sums without changing Δ.
- In case $i > 0$: If $\sum_2$ contains more addends than $\sum_1$ then moving i addends from $\sum_2$ to $\sum_1$ increases Δ.
- If there are two elements which do not appear in either of the sums, then Δ may be increased by introducing the larger one to $\sum_1$ and the smaller one to $\sum_2$.
- If an element not appearing in either of the sums is larger than some addend in $\sum_1$ then interchanging these elements will increase Δ, and the same works if the element is smaller than some addend in $\sum_2$.
- If there is a pair of elements the larger of which appears in $\sum_2$ and the smaller one in $\sum_1$ we can increase Δ by interchanging those elements between the sums.

To see that the result is as claimed, we only need the following two observations. Since $\lfloor \frac{i}{2} \rfloor + \lceil \frac{i}{2} \rceil = i$, the numbers of addends in the claimed sums differ by the amount required. Furthermore, in both cases the maximal possible amount of knapsack elements are involved in the claimed sums (i.e. all, or all but one).

(ii) Assume $|A| \geq 3$. Then Δ_1 equals $\max(A)$ plus a positive difference of two sums.

If $i = 2, \ldots, |A|$, then Δ_i equals Δ_{i-1} plus one element of A. Thus Δ_i is obtained by adding $i - 1$ elements to Δ_1. Since $\Delta_1 \geq \max(A)$, we get $\Delta_i < i\,\Delta_1$. □

Lemma 9.3 Let A be a fix-injective knapsack. Choose an integer $d > \Delta_1$. Then

(i) both the knapsacks $d + A$ and $d - A$ are injective and

(ii) the difference between two subset sums of $d \pm A$ is either smaller than Δ_0 or greater than $d - \Delta_1$.

Proof. The following treatment concerns either one of the two knapsacks $d \pm A$.

Take two subset sums $\sum_1$ and $\sum_2$. For $i = 1, 2$ let k_i be the number of addends in the sum $\sum_i$. Then $\sum_i = k_i d \pm \sum_i'$, where $\sum_i'$ is the corresponding subset sum of A.

If $k_1 = k_2$, we have

$$|\sum_1 - \sum_2| = |\sum_1' - \sum_2'| \leq \Delta_0.$$

From the equation we see that if the sums $\sum_1$ and $\sum_2$ coincide then so do the sums $\sum_1'$ and $\sum_2'$, which by the assumption on A implies that the addends are the same.

Suppose now that $|k_1 - k_2| = i \geq 1$.

$$|\sum_1 - \sum_2| = |(k_1 - k_2)d \pm (\sum_1' - \sum_2')| \geq$$

$$\geq |k_1 - k_2|d - |\sum_1' - \sum_2'| \geq i\,d - \Delta_i > i(d - \Delta_1),$$

where the last inequality follows from Lemma 9.2 (ii). Since $d > \Delta_1$ we see that the sums $\sum_1$ and $\sum_2$ cannot coincide in the case $|k_1 - k_2| \geq 1$. The proof of injectivity is thus complete. From the derived inequalities we see that also the claim (ii) is true. □

The following theorem is a direct consequence of Theorem 7.2 and corresponds to the situation of equation (12), with $d_1 = \delta_A$ and $d_2 = s_A$. It will be used in the next theorem in the special case when all the elements of B are greater than s_A.

Theorem 9.4 Assume that A and B are injective knapsacks. Denote $\delta_A = \min(D_A)$, that is, the minimum of all positive differences between two subset sums of A. Assume that any two subset sums of B differ from each other an amount that is either smaller than δ_A or greater than s_A. Then the knapsack $A \cup B$ is injective. □

Theorem 9.5 Assume that A and B are injective knapsacks, and denote $\delta_A = \min(D_A)$. Use the notation Δ_0 and Δ_1 for B. Choose such integers w and d that

$$w\delta_A > \Delta_0 \quad \text{and} \quad d > \Delta_1 + ws_A.$$

Then the knapsacks

$$wA \cup (d + B) \quad \text{and} \quad wA \cup (d - B)$$

are injective.

Proof. Firstly, $s_{wA} = ws_A$ and $\min(D_{wA}) = w\delta_A$. Secondly, the knapsack $d \pm B$ is injective by Lemma 9.3 (i). From Lemma 9.3 (ii) it follows that the difference between two subset sums of $d \pm B$ is either smaller than Δ_0 or greater than $d - \Delta_1$. By assumptions we now know that this difference is either smaller than $w\delta_A$ or greater than ws_A. Thus the knapsacks wA and $d \pm B$ satisfy the conditions of the Theorem 9.4 and the claim follows. □

Example. Let $A = \{3, 5, 6, 7\}$ and $B = \{5, 7, 10, 13, 14\}$. Now $s_A = 21$ and $\delta_A = 1$ and for B we have $\Delta_0 = 15$ and $\Delta_1 = 25$ as in a previous example.

The values $w = 16$ and $d = 362$ satisfy $w\delta_A > \Delta_0$ and $d > \Delta_1 + ws_A$, and hence the knapsacks

$$\begin{aligned} wA \cup (d + B) &= \{48, 80, 96, 112, 367, 369, 372, 375, 376\} \\ wA \cup (d - B) &= \{48, 80, 96, 112, 357, 355, 352, 349, 348\} \end{aligned}$$

are injective. Since w and d were the smallest possible, these are the densest knapsacks that can be constructed from A and B in this way and order. The densities are 1.0521 and 1.0660, and both are lower than $\rho_2(9) = 1.125$ although A and B have high densities.

If we try in the other order, and again with the smallest possible parameters we get the injective knapsacks

$$\begin{aligned} 6B \cup (305 + A) &= \{30, 42, 60, 78, 84, 308.310, 311, 312\} \\ 6B \cup (305 - A) &= \{30, 42, 60, 78, 84, 302, 300, 299, 298\}, \end{aligned}$$

which are slightly denser than the previous ones.

9.2 Union of a knapsack and its sum times another knapsack

We construct from two injective knapsacks A and B a new injective knapsack of the form $A \cup B'$, where B is roughly the same as $s_A B$.

Theorem 9.6 Assume A and B are injective knapsacks. Choose a positive integer r and put $a = s_A + r$. Then define $B' = \{b'_1, \ldots, b'_{|B|}\}$ with

$$b'_i = ab_i + d_i, \qquad i = 1, \ldots, |B|,$$

where d_i are such integers that $\sum_{k=1}^{|B|} |d_k| < r$. Then the knapsack $A \cup B'$ is injective.

Proof. Assume that two subset sums $\sum_1$ and $\sum_2$ of $A \cup B'$ coincide. For $i = 1, 2$ denote by $\sum_{iA}$ and $\sum_{iB'}$ those parts of $\sum_i$ which stem from the sets A and B' correspondingly. We thus have

$$\sum_{1A} + \sum_{1B'} = \sum_1 = \sum_2 = \sum_{2A} + \sum_{2B'}$$

and hence

$$\sum_{1A} - \sum_{2A} = \sum_{2B'} - \sum_{1B'} = \sum(ab_i + d_i) - \sum(ab_j + d_j). \qquad (20)$$

We will show that the subset sums $\sum b_i$ and $\sum b_j$ of B coincide. Assume the contrary: $\sum b_i \neq \sum b_j$. The absolute value of the RHS of 20 is

$$\left|a\left(\sum b_i - \sum b_j\right) - \left(\sum d_i - \sum d_j\right)\right| \geq$$

$$a\left|\sum b_i - \sum b_j\right| - \left|\sum d_i - \sum d_j\right| \geq$$

$$a - \sum_{k=1}^{|B|} |d_k| > s_A + r - r = s_A.$$

This is a contradiction because the absolute value of the LHS of (20) is at most s_A.

We conclude that $\sum b_i = \sum b_j$. This means, by injectivity of B, that these sums have the same addends. Hence also the sums $\sum_{2B'}$ and $\sum_{1B'}$ have the same addends and (20) is an equation of zeros. From $\sum_{1A} - \sum_{2A} = 0$ we finally deduce by injectivity of A that all the addends in the sums $\sum_1$ and $\sum_2$ are the same. This proves the injectivity of $A \cup B'$. □

Remark 1. If the knapsack B is superincreasing with $b_2 \geq 2b_1$ (and all the $d_i = 0$) then the theorem only means superincreasing continuation of A. The integers d_i could not help very much in this respect. The idea is of course that B is not superincreasing and the main purpose of the d_i's is to hide the multiplier a from being a common factor of the b'_i.

Remark 2. Some, but not all, instances of the "three elements" system of Section 7.2 can be considered as a special cases of this. For example $A \cup B'$ is a "three elements" system, when $s_A = 9$ and $B' = \{20, 30, 40\}$. We get this system now by choosing $B = \{2, 3, 4\}$, $r = 1$ and $d_1 = d_2 = d_3 = 0$.

Consider then the "three elements" system $\{10\} \cup \{21, 32, 43\}$. If it is to be of the form $A \cup B'$ according to the theorem, we must have $A = \{10\}$ and $B' = \{21, 32, 43\}$. Otherwise the smallest element of B' would not exceed the sum of A. Let us now try to find an injective knapsack $B = \{b_1, b_2, b_3\}$ for the construction. This requires that $b_3 \geq 4$, and we must have such integers r, a and d_3 that

$$r \geq 1, \quad a = 10 + r, \quad |d_3| < r \quad \text{and} \quad ab_3 + d_3 = 43.$$

Since $b_3 \geq 4$ and $a \geq 11$ we must have $d_3 < 0$. Taking into account that $a > 10 + |d_3|$ we have

$$ab_3 + d_3 > (10 + |d_3|)b_3 - |d_3| \geq 40 + |d_3|(b_3 - 1) \geq 43,$$

which is a contradiction.

Remark 3. The condition $\sum_{k=1}^{|B|} |d_k| < r$ in the theorem cannot be improved by letting for example $\sum_{k=1}^{|B|} |d_k| \leq r$. To show this take $A = \{1, 2\}$ and $r = 1$, which means that $a = s_A + r = 4$. Now let $B = \{1\}$ and take $d_1 = -1$. Then $B' = \{4 \cdot 1 - 1\} = \{3\}$, but $A \cup B' = \{1, 2, 3\}$ is not injective.

Example. Take $A = \{3,5,6,7\}$ and $B = \{2,3,4\}$. Choose $r = 3$, and $d_1 = 1$, $d_2 = 0$, $d_3 = -1$. Since $s_A = 21$, we have $a = s_A + r = 24$ and

$$B' = \{24\cdot 2 + 1,\ 24\cdot 3 + 0,\ 24\cdot 4 - 1\} = \{49, 72, 95\}.$$

The resulting knapsack

$$A = A \cup B' = \{3,5,6,7,49,72,95\}$$

is injective by the theorem, but its density is not very high: $\rho(A) = 1.065 < 1.167 = \rho_2(7)$.

9.3 Residuals "rescuing" the injectivity

When looking for (small) dense knapsacks we may find a knapsack that is nearly injective. Suppose that the knapsack is fix-injective. Then the injectivity may be "rescued" using Lemma 9.3. The following theorem generalizes the lemma with the construction of Theorem 9.6. Notice that B can be "rescued" also alone, because the theorems 9.6 and 9.7 remain valid, if we allow A to be the empty set, and use $s_A = 0$.

Theorem 9.7 Assume that a knapsack B and an integer m satisfy the following condition: Whenever two different subsets of B correspond to the same sum, then the numbers of elements in these subsets are not congruent modulo m.

Assume that A is injective and all its elements are divisible by m.

Choose a positive integer r and put $a = s_A + r$. Choose an integer c such that, $\gcd(c,m) = 1$. Define $B' = \{b'_1, \ldots, b'_{|B|}\}$ by choosing such integers d_i that

$$\sum_{k=1}^{|B|} |d_k| < r \quad \text{and} \quad b'_i = ab_i + d_i \equiv c \pmod{m}, \qquad i = 1, \ldots, |B|.$$

Then the knapsack $A \cup B'$ is injective.

Proof. The proof is identical to the proof of Theorem 9.6 up to the conclusion $\sum b_i = \sum b_j$. If we can now deduce that these sums have the same addends then the proof is completed in the same way as before, when the coinciding sums $\sum_1$ and $\sum_2$ turned out to have the same addends.

Assume on the contrary that the two sums correspond to different subsets of B. By assumption the sums have such numbers of addends, say k_1 and k_2, that $k_1 \not\equiv k_2 \pmod{m}$. The same is then true of the sums $\sum_{1B'}$ and $\sum_{2B'}$.

Now for the sums $\sum_{1B'}$ and $\sum_{2B'}$ we have

$$\sum_{1B'} = \sum(ab_i + d_i) \equiv k_1 c \pmod{m} \quad \text{and} \quad \sum_{2B'} \equiv k_2 c \pmod{m}$$

and

$$\sum_{1B'} - \sum_{2B'} \equiv (k_1 - k_2)c \not\equiv 0 \pmod{m},$$

because $k_1 - k_2 \not\equiv \pmod{m}$ and $\gcd(c,m) = 1$.

Since the elements of A are congruent to 0 modulo m, we now have that the sums $\sum_1$ and $\sum_2$ are not congruent modulo m. This is a contradiction because we assumed that these sums coincide. □

Remark. Suppose B is fix-injective. If B is dense then no element equals the sum of all the others. Consequently $m = |B| - 2$ is a possible value for the modulus in the theorem. Namely all sums with 1 or $|B| - 1$ addends are different, as we just said, and the same is of course true for sums with 0 and $|B| - 2$ as well as with 2 and $|B|$ addends.

The modulus m is essentially the "price we have to pay" in making the knapsack injective. Firstly, starting from a general A we have to multiply it by m, which means that instead of s_A we have ms_A. On the other hand the larger m is, the more space the integers d_i may need to satisfy the congruences. Allowing this space means an increase in r.

Both these effects enlarge the value of a, which is the multiplier of B and directly affects the density.

Example. The knapsack $B = \{1, 2, 3, 5\}$ satisfies the requirements of the theorem for $m = 2$. This can be seen by computing all the 16 sums. Four of them appear twice, but each time one of them has an even number of addends and the other an odd number. Use $A = \emptyset$, $s_A = 0$ and $c = 1$ in the theorem. The multiplier $a = 2$ will not do, because it would lead to $\sum |d_k| = 4$. With $a = 3$ we get the injective knapsacks $\{3, 6 \pm 1, 9, 15\}$.

The knapsacks

$$B_1 = \{2, 5, 8, 14, 24, 43, 44, 45\} \quad \text{and} \quad B_2 = \{1, 22, 38, 45, 48, 50, 51, 52\}$$

are quite dense, obviously denser than any injective knapsack of this size. To use the theorem, B_1 requires that $m \geq 4$, but B_2 will do with $m = 2$.

Assuming $A = \emptyset$ and $m = 4$ it turns out that the smallest multiplier a for B_1 is 9 when $c = 1$ and 10 when $c = 3$. The smaller modulus $m = 2$ for B_2 allows us to take $a = 7$ when $c = 1$. In all of these cases the transformation results in a density below 1.

10 Chor-Rivest system and modified usages

Chor and Rivest presented in [7] a knapsack cryptosystem that is based on arithmetics in finite fields. A modular reduction of the sum can be carried out during encryption, but the trapdoor is not based on modular multiplication. Instead there is some similarity to the Merkle-Helman multiplicative trapdoor knapsack. Also here the knapsack consists of logarithms of a multiplicative knapsack, but now the elements are chosen in a special way from a large finite field. Lenstra [19] proposes a cryptosystem that uses this multiplicative knapsack and avoids the laborious computation of discrete logarithms. We will not consider this system here.

The Chor-Rivest knapsack is generally not injective. It is fix-h-injective modulo $q^h - 1$, where the value h in practical systems is approximately $\frac{1}{8}$ times the knapsack size q.

To our knowledge the Chor-Rivest system has not been broken. After noting that their low density attack does not threaten the security of the Chor-Rivest system asymptotically Coster et al. write in [9, p. 124]: "On the

other hand, for moderate sizes of the problem (such as a challenge version of the Chor-Rivest knapsack with $n = 103$ that was constructed by B. Chor) solutions can be found with nonnegligible probability. Thus to obtain secure cryptosystems, one has to use very large values of the basic parameters, which makes this system less attractive."

We review the system in Section 10.1. In Section 10.2 we present some modified ways how the system could be used either to increase security or to avoid the difficulty of constructing very large knapsacks.

We have investigated the Chor-Rivest knapsack from an algebraic point of view in [16]. Those results may have some significance in deeper analysis of the security of the cryptosystem, but in the present study we do not need them.

10.1 Chor-Rivest system

We first recall some basic facts about the finite fields. For more details we refer the reader to [21] or to the more concise treatment of [22].

If p is a prime and m a positive integer, we will denote with F_{p^m} the finite field that has p^m elements. The cryptosystem is based on a field extension between two fields $F_q \subset F_{q^h}$, where q is a power of a prime and $h \geq 2$. The number h is the *degree* of the extension and F_{q^h} is defined as the smallest field containing both F_q and the formal root x of an irreducible polynomial $f(x)$ of degree h from the polynomial ring $F_q[x]$.

The elements of F_{q^h} are represented as polynomials from $F_q[x]$ of degree at most $h-1$. Addition and multiplication are carried out between these polynomials and the remainder modulo $f(x)$ is the result. This means that only polynomial calculations need to be performed, the coefficients being from the *base field* F_q. However, if $q = p^m$ and $m > 1$, the field F_q must first be constructed similarly with respect to the *prime field* F_p, which is simply the set $\{0, 1, \ldots, p-1\}$ with modulo p arithmetic.

In the cryptosystem the number q will be the size of the knapsack and h gives the fixed number of addends in the sum instances. According to Chor and Rivest, q^h could be for example 197^{24}, or even 256^{25}. Discrete logarithms must be computed of q members of the extended field. This is the most laborious part of the whole cryptosystem. For the Pohlig-Hellman algorithm [28] to work efficiently it is advantageous if the largest divisor of $q^h - 1$ is not very large. This is why also the degree of the field extension h should have suitable divisors (for example, if $d \mid h$, then $q^d - 1$ is a divisor of $q^h - 1$).

Construction of the knapsack

Suppose the field setting (i.e. q and h) has already been chosen. We proceed by choosing an element t from F_{q^h} that is of degree h over F_q, i.e. a root of an irreducible polynomial of degree h from $F_q[x]$. Then we choose another element g that is primitive, i.e. a generator of the multiplicative group $F_{q^h}^*$ (which is always cyclic).

Since all the non-zero elements in F_{q^h} are powers of g, we can use g as a base for the logarithm. We now choose an integer d, $0 \leq d \leq q^h - 2$ and

compute the knapsack

$$A = \{\log(t + k) + d \mid k \in F_q\},$$

where the elements have been reduced modulo $q^h - 1$.

Notice that the initial definition of a knapsack is slightly extended, since zero may occur as a member of A. This does not affect decipherability because the number of addends in sum instances will always be h.

Encryption

The public key consists of the integers q and h and the elements of A in some permuted order. To simplify the notation we do not pay attention to this permutation.

To encrypt a message it must first be transformed into blocks of q bits that contain exactly h 1's. For each block the elements of A corresponding to the 1's are added and reduced modulo $q^h - 1$.

Decryption

The decryption algorithm requires that the arithmetic of the field F_{q^h} has been constructed by using just that irreducible polynomial $f(x) \in F_q[x]$ that has t as its root.

Suppose we are given an integer s that is a sum of h elements of A. The h addends or actually the corresponding elements of F_q will be found as roots of a polynomial. Suppose we have already eliminated the effect of d by subtracting $h \cdot d \pmod{q^h - 1}$.

We compute $g^s \in F_{q^h}$. As a polynomial of $F_q[x]$ g^s has at most $h - 1$ roots. We now add to it the zero of F_{q^h} in the form $x^h - r(x)$, where $r(x)$ has been computed in advance by reducing x^h modulo $f(x)$.

As a result we get a polynomial $\sigma(x) \in F_q[x]$ that is of degree h. Since we only added a zero we have $\sigma(x) = g^s$ in F_{q^h}.

For some elements $k_i \in F_q$,

$$s = \sum_{i=1}^{h} \log(t + k_i).$$

Since the polynomial representation of t as an element of F_{q^h} is $t = x$, we have

$$\sigma(x) = g^s = \prod_{i=1}^{h} (x + k_i) \in F_q[x].$$

We find the roots $-k_i$ of this polynomial by simply computing its value at the points $x \in F_q$. Knowing these roots amounts to knowing which elements of A were added to form s.

It is possible to modify the encryption algorithm to allow repeated addends. In this case we may find fewer than h elements k_i from the roots and need to divide $\sigma(x)$ repeatedly by each $x + k_i$ until we also find the multiplicities.

Density

When we construct a Chor-Rivest system in the field setting F_{q^h}, we get a knapsack of size q where the largest element is αq^h with $0 < \alpha < 1$. If q^h is not very small, α is generally very near 1. Thus the density of a real size knapsack is slightly more than

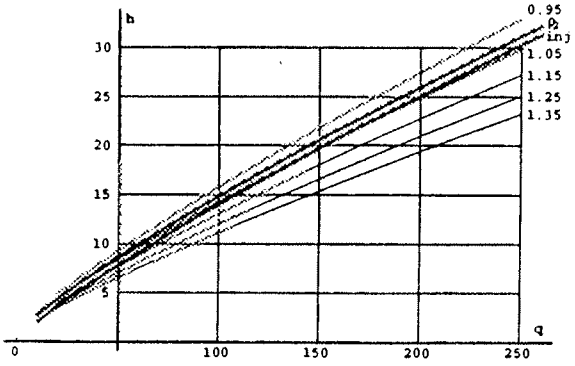
$$\rho(q,h) = \frac{q}{h \log_2 q}.$$

The figure below depicts the curves $\rho(q,h) = C$, for various constant values of C. In addition there are two other curves where C is a function of the knapsack size q. In the first curve $C = \rho_2(q)$, whence this curve delimits those pairs (q,h) for which the density is higher than the highest superincreasing density. In the last curve C is the upper bound from Lemma 2.1 for the density of an injective knapsack. Since $\rho(q,h)$ is a lower bound for the density of the Chor-Rivest knapsack with parameters q and h, we know that below the last curve there are no injective Chor-Rivest knapsacks.

For the proposed parameters $q = 197$ and $h = 24$ the density of the knapsack is at least

$$\frac{197}{\log_2 197^{24}} = 1.077$$

and for a knapsack of size 197 the injective upper bound is 1.040.



10.2 Modified usages of the Chor-Rivest knapsack

We will not try to modify the Chor-Rivest system itself. Instead we use it as a part of other knapsack constructions.

Additional trapdoors

Although the trapdoor is difficult to find from a Chor-Rivest knapsack it is in principle possible by an extremely lucky guess. If we apply some additional trapdoor transformation to the knapsack then the task will most likely require two extremely lucky guesses.

Strong modular multiplication is actually the only general trapdoor technique that is applicable to any given knapsack. In case of a Chor-Rivest knapsack it does not appear to be vulnerable in the way it is with the superincreasing knapsack, because there is no easy structure to be looked for behind the multiplied knapsack.

The problem with strong modular multiplication is that it will decrease the density, even if the initial knapsack is dense (actually, especially in this case). If we have a Chor-Rivest knapsack A with parameters q and h we may perform an h-modular multiplication. The new density will be above but near $q/\log_2(hq^h)$, which is not so much worse than before if h is not very large in proportion to q. For the proposed parameters $q = 197$ and $h = 24$ this formula gives 1.0506, whereas the original density was at least 1.077. The new density is still above the upper bound for the density of an injective knapsack.

Since A is fix-h-injective modulo $q^h - 1$ we might think of an additional trapdoor multiplication where the modulus is just $q^h - 1$. This will indeed work: if $\gcd(w, q^h - 1) = 1$, we can transform every subset sum of $wA \pmod{q^h - 1}$ with h addends to the corresponding sum of A. The only problem here is that $wA \pmod{q^h - 1}$ is just another Chor-Rivest knapsack. To show this assume that

$$A = \{\log(t + k) + d \mid k \in F_q\},$$

where the base for the logarithm is g. Denote $b = g^{w^{-1}}$, where w^{-1} is the inverse of w modulo $q^h - 1$. We have now $w = (\log b)^{-1}$ and

$$wA = \left\{\tfrac{\log(t+k)}{\log b} + wd \mid k \in F_q\right\} \pmod{q^h - 1}.$$

On the other hand $\frac{\log x}{\log b}$ is the logarithm of x to base b. Thus we see that $wA \pmod{q^h - 1}$ is a Chor-Rivest knapsack with the same element t as in A but with $g^{w^{-1}}$ instead of g and $wd \pmod{q^h - 1}$ instead of d.

Usage of Chor-Rivest as a part of a larger knapsack

We may try to exploit the high density of a Chor-Rivest knapsack by taking it as a starting point of the iterative constructions proposed in Section 7.1. In this way we may avoid the effort of constructing large Chor-Rivest knapsacks.

There is a difficulty in this approach and it is the fact that we can decipher a Chor-Rivest knapsack with parameters q and h only when there are exactly h addends in its subset sums. We sketch an attempt to solve this problem by using superincreasing construction as a simple example of the ideas of Section 7.1.

Assume A_1 is a Chor-Rivest knapsack with $|A_1| = q$ and the required number of addends is h. Now extend A_1 to a larger knapsack A by joining new elements in the superincreasing fashion. The first new element will be greater than the sum of A_1, etc.

Choose an upper bound k for the number of addends in subset sums. Assume further that the number is always given together with the subset sums. Use k-modular multiplication as a trapdoor transformation.

Decryption starts in the normal way with the superincreasing part of A. After this, if there are still more than h addends to resolve from the Chor-Rivest knapsack A_1, then the decipherment has failed and a new somewhat modified message must be requested from the sender.

If there are exactly h addends left, they can be found in the normal way from the knapsack A_1. If there are fewer, say ℓ addends left, then $h-\ell$ elements of A_1 are added to the sum and the resulting sum instance is deciphered in the normal way. Here we need the fact that decipherment succeeds also when the same addend occurs more than once.

By choosing a sufficiently small k we can make the probability of failure arbitrarily low. Indeed, if $k = h$ then the probability is 0.

Suppose we want there to be on average βh addends from A_1 in subset sums of A that have k addends. Assuming that all of these subset sums occur with the same probability, we must have $\frac{k}{|A|} \approx \frac{\beta h}{q}$. The uniform distribution namely implies that the number of addends from each part of the knapsack is proportional to the size of that part. The value of β must be somewhat smaller than 1 to ensure that decipherment succeeds at least in half of the cases. This means that k is slightly smaller than $\frac{h}{q}|A|$, which is typically $\frac{1}{8}|A|$.

We would like to have the number of addends near $\frac{1}{2}|A|$ to protect the knapsack system against low density attacks. This would require an increase of k and consequently of the ratio $\frac{h}{q}$. But this would cause the density of A_1 to become lower, because it is approximately $\frac{q}{h}\frac{1}{\log_2 q}$. Also the trapdoor multiplication decreases the density the more, the larger k is.

Parallel use of Chor-Rivest and another knapsack

The number of addends in sum instances of the Chor-Rivest system is fairly small when compared to the knapsack size. The number of addends is the same as the squared length of the special vector $\hat{e}$ searched by the lattice reduction attacks. The shorter the vector $\hat{e}$ is known to be the higher is the critical density below which knapsacks become insecure against these attacks.

One possible remedy was suggested already by Chor and Rivest in [7], and that is to produce sum instances where some elements occur with higher multiplicities than 1. This increases the length of $\hat{e}$ considerably although the sum of its components is still h. Another approach to increase the length of $\hat{e}$ is outlined in the following.

Assume A is a Chor-Rivest knapsack with parameters q and h. That is, $|A| = q$ and the required number of addends is h. We assume also that h is about $\frac{q}{8}$ or slightly less, as it is in the cases $(q,h) = (197,24)$ and $(q,h) = (256,25)$ proposed in [7]. Assume B is an injective trapdoor knapsack with $|B| = q$. Let the messages correspond to subsets $S \subseteq \{1,\ldots,q\}$ and assume that every subset is equally likely to occur.

Suppose $|S| \geq \frac{q}{2}$. Compute the corresponding subset sum of A and denote it by s. Now choose randomly such a subset $S' \subseteq S$ that $|S'| = |S| - h$. Then compute the subset sum of B corresponding to S' and denote it by s'. If $|S| < \frac{q}{2}$ then use the complement of S instead (and inform the recipient about this).

The legal decryption consists of deciphering first the sum instance s' of B and subtracting the corresponding elements of A from s and deciphering the resulting sum instance of h elements of the Chor-Rivest system A.

An enemy willing to decipher the sum instance s directly faces the following difficulties

- The number of addends in the sum instance s is at least $\frac{q}{2}$ and on average does not exceed this value very much. The number is not known.
- Even if the enemy knew the secret key of A the sum instance cannot be deciphered in the normal way.
- There may be several possible decipherments of s, because A is not necessarily injective.

If the enemy wants to use the information of s' he basicly first faces one knapsack problem where the number of addends is unknown and not very small; it is $\frac{3q}{8}$ or near above this value. After getting to know the addends of s' he knows $|S'|$ bits of the message and also that among the remaining $q - |S'|$ bits there are exactly h 1's. If redundancy does not help in finding those the only way is to solve the normal Chor-Rivest knapsack problem.

That the two knapsack problems with s and s' are related may be a weakness. If the h extra elements in the sum s are not enough to confuse the relation, then the sum s' could be enciphered according to B.

This system is an example of computing a generalized sum instance (not necessarily even a subset sum) of the Chor-Rivest system and then telling in an enciphered form how to obtain a decipherable subset sum.

References

[1] Adleman L.M. *On Breaking Generalized Knapsack Public Key Cryptosystems*, Proceedings of the Annual ACM Symposium on Theory of Computing, Boston, Mass. April 25-27, 1983, 402-412.

[2] Beth Th., Frisch M., Simmons G.J. (eds) *Public-Key Cryptography: State of the Art and Future Directions*, E.I.S.S. Workshop 1991, Springer Verlag, Berlin.

[3] Brassard G. *Modern Cryptology, a Tutorial*, Lecture Notes in Computer Science 325, Springer Verlag, 1988.

[4] Brickell E.F. *Breaking Iterated Knapsacks*, Advances in Cryptology, Proceedings of CRYPTO'84, Lecture Notes in Computer Science 196, Springer Verlag, 1985, 342-358.

[5] Brickell E.F., Odlyzko A.M. *Cryptanalysis: A Survey of Recent Results*, Proceedings of the IEEE, Vol. 76, No 5, 1988, pp. 578-593.

[6] Brickell E.F. *The cryptanalysis of knapsack cryptosystems*, in: Ringeisen R.D, Roberts F.S. (eds.), Applications of Discrete Mathematics, Proceedings of the 3rd SIAM Discrete Mathematics Conference (1986), SIAM, 1988, 3-23.

[7] Chor B., Rivest R.L. *A Knapsack-Type Public Key Cryptosystem Based on Arithmetic in Finite Fields*, IEEE Transactions on Information Theory, Vol. 34 No. 5, 1988, 901-909.

[8] Cooper R., Patterson W. *A Generalization of the Knapsack Algorithm Using Galois Fields*, Cryptologia Vol. 8 No. 4, 1984, 343-347.

[9] Coster M.J., Joux A., LaMacchia B.A., Odlyzko A.M., Schnorr C.P., Stern J. *Improved Low Density Subset Sum Algorithms*, Computational Complexity Vol. 2 No. 2 (1992), 111-128.

[10] Desmedt Y.G., Vandewalle J.P., Govaerts R.J.M. *A critical analysis of the security of the knapsack public-key cryptosystems*, IEEE Transactions on Information Theory, Vol. 30, Nr. 4, July 1984, pp. 601-611.

[11] Desmedt Y. *What happened with the knapsack cryptosystems?* Performance Limits in Communication, Theory and Practice, NATO ASI Series E: Applied sciences Vol. 142, Skwirzynski J.K. (ed.), Kluwer Academic Publishers, 1988, pp. 113-134.

[12] Diffie W. *The First Ten Years of Public-Key Cryptography*, Proceedings of the IEEE, Vol. 76, No. 5, 1988, pp. 560-576.

[13] Frieze A.M. *On the Lagarias-Odlyzko algorithm for the subset sum problems*, SIAM Journal on Computing Vol. 15 No. 2 (1986), 536-539.

[14] Goodman R.M.F., McAuley A.J. *A New Trapdoor Knapsack Public Key Cryptosystem*, Advances in Cryptology, Proceedings of EUROCRYPT'84, Lecture Notes in Computer Science 209, Springer Verlag, 1985, 150–158.

[15] Johnson D.S. *Catalog of complexity classes*, in: Leeuwen J. (ed.) Handbook of Theoretical Computer Science, Vol. A, Algorithms and Complexity, Elsevier, Amsterdam, 1990, 67–162.

[16] Koskinen J.A. *Knapsack sets for cryptography from finite fields*, Licentiate's thesis, University of Turku, 1992.

[17] Lagarias J.C, Odlyzko A.M. *Solving low density subset sum problems*, Proc. 24th IEEE Symp. on Foundations of Computer Science, 1983, pp. 1–10.

[18] Lenstra A.K., Lenstra H.W. Jr., Lovász L. *Factoring Polynomials with Rational Coefficients*, Mathematische Annalen 261 (1982), 515–534.

[19] Lenstra H.W. Jr. *On the Chor-Rivest knapsack cryptosystem*, Journal of Cryptology, Vol. 3, No. 3, 1991, 149–155.

[20] Leung C. *Some Open Problems in Cryptography*, Proceedings of the Annual Conference of ACM (1978), 471–475.

[21] Lidl R., Niederreiter H. *Finite Fields*, Encyclopedia of Mathematics and its Applications, Vol. 20, Cambridge University Press, 1985.

[22] Lidl R., Pilz G. *Applied Abstract Algebra*, Springer Verlag, New York, 1984.

[23] Lu S., Lee L. *A Simple and Effective Public Key Cryptosystem*, Comsat Technical Review, Vol. 9, No. 1, 1979.

[24] Merkle R.C., Hellman M.E. *Hiding information and signatures in trapdoor knapsacks*, IEEE Transactions on Information Theory, Vol. 24, No. 5 (1978), 525–530.

[25] Niemi V. *A New Trapdoor in Knapsacks*, Advances in Cryptology, Proceedings of EUROCRYPT'90, Lecture Notes in Computer Science 473, Springer Verlag, 1991, 405–411.

[26] Odlyzko A.M. *Cryptanalytic Attacks on the Multiplicative Knapsack Cryptosystem and on Shamir's Fast Signature Scheme*, IEEE Transactions on Information Theory, Vol. 30, Nr. 4, July 1984, pp. 594–600.

[27] Odlyzko A.M. *The Rise and Fall of Knapsack Cryptosystems*, in: Pomerance C. (ed.): Cryptology and Computational Number Theory, American Math. Soc. 1990, pp. 75–88.

[28] Pohlig R.C., Hellman M. *An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance*, IEEE Transactions on Information Theory, Vol. 24, 1978, 106–110.

[29] Rivest R.L. *Cryptography*, in: Leeuwen J. (ed.): Handbook of Theoretical Computer Science, Vol. A, Algorithms and Complexity. Elsevier, Amsterdam, 1990, 717-756.

[30] Salomaa A. *Computation and Automata*, Encyclopedia of Mathematics and its Applications, Vol. 24, Cambridge University Press, 1985.

[31] Salomaa A. *Public Key Cryptography*, Springer Verlag, Berlin, 1990.

[32] Salomaa A. *Decision problems arising from knapsack transformations*, Acta Cybernetica, Tom. 9, Fasc. 4, Szeged, 1991, 419-440.

[33] Schnorr C.P., Euchner M. *Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems*, Proceedings of Fundamentals of Computation Theory '91, Lecture Notes in Computer Science 529, Springer Verlag, New-York, 1991, 65-85.

[34] Seberry J., Pieprzyk H. *Cryptography: an introduction to computer security*, Prentice Hall, New York London 1989.

[35] Shamir A., Zippel R.E. *On the Security of the Merkle-Hellman Cryptographic Scheme*, IEEE Transactions on Information Theory, Vol. 26, No. 3 (1980), 339-340.

[36] Shamir A. *A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem*, IEEE Transactions on Information Theory, Vol. 30, No. 5, (1984), 699-704.

[37] Webb W.A. *A Public Key Cryptosystem Based on Complementing Sets*, Cryptologia Vol. 16 No. 2, (1992), 177-181.

[38] Welsh D.J.A *Matroid Theory*, Academic Press, London 1976.