Working Paper Series
ISSN 1170-487X

# The Subset Sum Problem and Arithmetic Coding

**by Sean A. Irvine, John G. Cleary, Ingrid Rinsma-Melchert**

Working Paper 95/7
March 1995


Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

# The Subset Sum Problem and Arithmetic Coding

Sean A. Irvine
John G. Cleary
Ingrid Rinsma-Melchert
School of Computing and Mathematical Sciences,
University of Waikato,
Private Bag 3105,
Hamilton,
New Zealand.
Electronic mail: {sirvine,jcleary,ingrid}@waikato.ac.nz

March 23, 1995

**Abstract**

The security offered by symmetric cryptosystems based on the arithmetic coding algorithm is examined. It is shown that this can be reduced naturally to the subset sum problem. The subset sum problem is **NP**-complete, however, the cases which arise in practical cryptosystems based on this problem tend to be solvable in polynomial time because the sums formed are either superincreasing or of low density. Our attack is therefore similar to attacks on public-key cryptosystems based on the subset sum problem (knapsack systems).

# 1 Introduction

It is well known that compression of plaintext prior to encryption offers better protection than encryption alone. In particular it reduces redundancy making statistical attacks more difficult and prevents dictionary attacks [9, 26]. It is interesting to consider whether or not compression techniques alone are sufficient for encryption. Most modern compression techniques consist of a modeler which attempts to capture statistical regularities and a coder which codes the message with respect to the model. This paper examines the security offered by arithmetic coding; an optimal coding technique used by many compression methods. A good explanation and implementation of arithmetic coding is given by Witten, Neal, and Cleary [27].

A method of using key information in the arithmetic coding algorithm is considered. The resulting cryptosystems will be shown to have a close relationship with the subset sum problem.

Section 2 introduces the subset sum problem and discusses its complexity, showing that in general it is a hard problem. It is proved that solving the subset sum problem is at least as hard as solving any other **NP**-complete problem. Two classes of subset sums which have polynomial-time solutions are then examined. It is shown that the subset sums produced by the arithmetic coders discussed in Section 3 happen to form a polynomial-time solvable class of subset sum problems. In particular, they frequently form superincreasing sums, or sums which can be solved by the short-vector algorithm [16].

# 2 The Subset Sum Problem

The **subset sum problem** is a special case of the **knapsack problem** and in the cryptology literature is often referred to as the knapsack problem. The subset sum problem is hard, its decision problem was shown to be **NP**-complete by Karp [14]. It will be shown that the related search problem of actually finding a solution, even when a solution is known to exist, is at least as hard as *any* **NP**-complete problem.

DEFINITION 1: Let $K = \{\kappa_1, \kappa_2, \ldots, \kappa_k\}$ be a finite set and $t$ be a positive integer called the **target**. Associated with each $\kappa_i \in K$ is a positive integer $s(\kappa_i)$ called its **size**. The **subset sum decision problem** asks the question: Is there any subset $C$ of $K$ such that the sum of the sizes of the elements in

$C$ is precisely $t$; that is, does there exists a $C$ such that

$$C \subseteq K \text{ and } \sum_{j \in C} s(\kappa_j) = t?$$

The problem can also be stated as a $\{0,1\}$-integer programming problem. Does there exist a solution to

$$\sum_{i=1}^{k} s(\kappa_i) x_i = t \qquad \text{where } x_i = 0 \text{ or } x_i = 1?$$

In addition to the decision problem there are the related search problems of finding a particular solution and of finding all solutions.

As defined the sizes can only be positive integers, but the problem can be generalized to allow for any integer sizes. Any element with size zero does not affect the decision problem, but if a solution does exist then the presence of size zero elements implies the existence of additional solutions. Allowing negative sizes cannot result in a simpler problem since an algorithm that could solve this problem over all the integers could definitely solve the problem over the positive integers.

THEOREM 1: The subset sum problem is **NP**-complete [14].

THEOREM 2: Finding a solution to subset sum problems, even when a solution is known to exist, cannot be done in polynomial-time unless **P=NP**.

PROOF: The existence of a polynomial-time algorithm for this problem is postulated, such an algorithm could be used to solve the subset sum decision problem in polynomial time. Thus the postulated algorithm cannot exist unless **P=NP**.

In detail, let $\mathcal{A}$ be a polynomial-time algorithm that computes a solution to a subset sum problem known to have at least one solution. Since $\mathcal{A}$ is a polynomial-time algorithm, there must exist a polynomial upper bound $p$ in the cardinality of the set.

Given an arbitrary subset sum problem, $S$, run $\mathcal{A}$ on $S$. If $\mathcal{A}$ halts within the time $p$ check the output (checking takes polynomial-time since the subset sum problem is in **NP**). If the output is not a solution then $S$ does not have a solution. Alternatively, if $\mathcal{A}$ fails to halt within the time $p$ then $S$ does not have a solution. In either case we have decided in polynomial-time whether or not $S$ has a solution.

Since the subset sum decision problem is **NP**-complete it follows that if $\mathcal{A}$ exists then **P=NP**. $\triangle$

The subset sum problem has previously been suggested for use in public-key cryptosystems [11, 19, 23]. These first proposals were broken because of special structure present in the subset sums used [24, 25]. These results were subsequently summarized in [21, 22]. Since the original system was broken many variants have been proposed. Many of these have also been analyzed and broken. Overviews of these systems and their cryptanalysis can be found in [3, 4, 5, 8]. The Chor-Rivest knapsack [7] is currently unbroken, but is computationally expensive.

Theorem 2 is a worst case bound, it proves there is at least one subset sum problem which is hard to solve. Such a result is *not* sufficient grounds on which to rest security claims. Security requires nearly every case to be hard. However, complexity theory has not yet evolved far enough to obtain these stronger results. The best alternative is to show a system does not produce cases known to be solvable in polynomial time. We emphasize again the absence of such cases does not guarantee security.

Two classes of subset sums having polynomial-time algorithms are now examined: superincreasing subset sums (solvable with a simple greedy algorithm in linear time) and low-density subset sums (solvable with the short-vector algorithm in polynomial-time).

## 2.1 Superincreasing Subset Sums

In a **superincreasing subset sum** $\mathbf{s} = (s(\kappa_1), \ldots, s(\kappa_k))$ each term $s(\kappa_i)$ is bigger than the sum of all the preceding terms; that is

$$s(\kappa_i) > \sum_{j=1}^{i-1} s(\kappa_j).$$

For example $1, 2, 4, 8, 16$ is a superincreasing subset sum, as is $1, 3, 9, 30, 102$ whereas $1, 3, 9, 30, 40$ is not. Superincreasing subset sums can be solved by a simple greedy algorithm (Figure 1) in linear time. The algorithm starts with the largest number in the sum, if this number is smaller than the current target $t$ then it is in the sum, otherwise it is not. Reduce the target by the amount of this element if it is in the sum, otherwise don't change the target. Iterate on the next smallest element. If the target is reduced to zero on any iteration then the subset sum has a solution.

```
superincreasing-solve (s, k, t)

vector s;          [k-dimensional vector of sizes]
int k;             [number of elements in the vector]
int t;             [target for sum]

begin
        for (i ← k; i > 0; i − −) begin
                if (s_i < t) begin
                        t ← t − s_i;
                        output s_i is in the knapsack;
                end;
        end;
        if (t = 0)
                output solution complete;
        else
                output no solution;
        end;
end;
```

Figure 1: Pseudocode for a greedy algorithm to solve superincreasing subset sum problems.

Superincreasing knapsacks were used as the basis of the first knapsack cryptosystems, with the superincreasing structure hidden by the transformation $s'(\kappa_i) = s(\kappa_i)n \bmod m$ for integers $n$ and $m$, where $m > \max_i\{s(\kappa_i)\}$ and $\gcd(n, s(\kappa_i)) = 1$ for all $i$. However, this transformation proved insufficient to hide the superincreasing structure and the system was broken [24, 25].

## 2.2 Low-Density Subset Sums

DEFINITION 2: The **density** $D$ of a subset sum with sizes $s(\kappa_1), \ldots, s(\kappa_k)$ is given by

$$D = \frac{k}{\log_2(\max_i\{s(\kappa_i)\})}. \tag{1}$$

If the density is low enough ($D < 0.645$ or $D < (2-\epsilon)[\log_2(4/3)]^{-1}/k$ for any fixed $\epsilon > 0$) then the **short-vector algorithm** [16] will almost surely find a solution.

The short-vector algorithm relies heavily on being able to find the shortest

vector in a $(k+1)$-dimensional lattice. No algorithm exists which has been proven to do this in polynomial time. Such a vector is typically found by first finding a reduced basis for the lattice. The $L^3$-algorithm [15] is a popular choice for finding the reduced basis and has complexity $O(k^6[\log(\max_i\{s(\kappa_i)\})]^3)$. When more advanced techniques are used for the arithmetic operations the complexity of finding a reduced basis with the $L^3$-algorithm can be reduced to $O(k[\log(\max_i\{s(\kappa_i)\})]^3)$.

# 3 A Simple Way of Using Key Bits

Arithmetic coding is a technique which converts a given probability distribution into an optimal code and is commonly used in compression schemes. Arithmetic coding is optimal in that it can encode information arbitrarily close to the entropy of the model. It improves on other codes such as Huffman codes [13] which are restricted to integral bits per character. The security of arithmetic coding has been considered previously in [6].

There are several ways in which a key can be added to the coding process with only a small loss of compression.

Consider encoding the output of a binary source with alphabet $\{a, b\}$ using a static model. Let $p$ be the probability used to encode an $a$ and $q = 1 - p$ be the probability used to encode a $b$. Neither probability can be exactly 0 or 1 as we must always allow some probability to each symbol occurring. The current interval is expressed by $l$ (lower bound) and $h$ (upper bound). $\delta = h - l$ is the difference between the upper and lower bound. Initially $l = 0$, $h = 1$, and $\delta = 1$. Let the input sequence be $S = s_1 s_2 \cdots s_n$, where $s_i$ is either $a$ or $b$. Let $a_i$ be 1 if the $i$th symbol is an $a$ and zero otherwise. Let $\overline{a_i} = \sum_{j=1}^{i} a_i$ denote the number of $a$s and $b$s in the input up to and including the $i$th character. Let $h_i$, $l_i$ and $\delta_i$ be the upper, lower, and width bounds respectively of the arithmetic coder after encoding symbol $i$.

## Interval Narrowing

The following is a simple method of making the result of arithmetic coding dependent on $k$ key bits $x_1, \ldots, x_k$. Although insecure, this simplistic model provides insight into more complicated models. Let $n$ denote the length of a message chosen from the alphabet $\{a, b\}$.

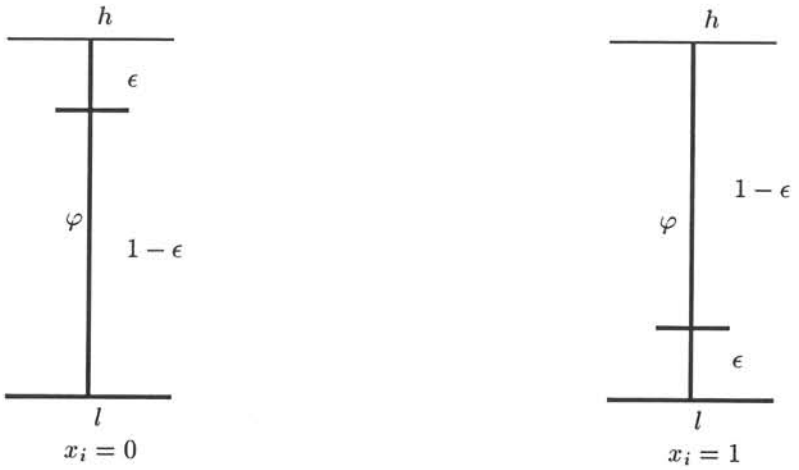


Figure 2: A simple model for using the key bits which results in polynomials having only linear terms in the key bits.

Prior to encoding each symbol the current interval will be narrowed. The narrowing will be accomplished by either increasing the value of the lower bound $l$ or decreasing the value of the upper bound $h$ by an amount $\epsilon\delta$, where $0 < \epsilon < 1$. To minimize compression loss $\epsilon$ should be small. Which of these operations is applied depends on a key bit, as illustrated in Figure 2. After narrowing the current interval the actual symbol will be encoded according to the compression model. This narrowing can also be considered as a two step encoding process where we first encode a cryptosymbol $\varphi$ using a key model and then the actual message symbol using the compression model.

The following recursive relationships for encoding with this model are obtained (expressions for the upper bound $h$ are not given as the results for the upper bound are symmetric with those for the lower bound):

For encoding '$\varphi$a':

$$\begin{aligned} \delta &\leftarrow (1-\epsilon)p\delta \\ l &\leftarrow l + x_i\epsilon\delta \end{aligned}$$

For encoding '$\varphi$b':

$$\begin{aligned} \delta &\leftarrow (1-\epsilon)p\delta \\ l &\leftarrow l + x_i\epsilon\delta + p(1-\epsilon)\delta \end{aligned}$$

(By letting $\epsilon \to 0$ these recurrences degenerate to those for keyless arithmetic coding.)

For the purposes of the analysis we will assume that $p$ is a constant. If $p$ is adaptive the results will not be greatly affected since by assumption the attacker is aware of the value of $p$ even if it is changing. An important aspect of this cryptosystem is that the width of the current interval is independent of the key bits. From an analytic point of view this is desirable because the characteristic polynomial for $l$ contains only linear terms in the key bits:

$$\begin{aligned} \delta_n &= (1-\epsilon)^n p^{\overline{a_n}} (1-p)^{n-\overline{a_n}} \\ l_n &= \sum_{i=1}^{n} [x_i \epsilon \delta_{i-1} + (1-a_i) p (1-\epsilon) \delta_{i-1}] \end{aligned}$$

In any implementation of this model $\epsilon$ and $p$ will be approximated by rational numbers. The equation for $l_n$ can be normalized to integers, since if $\epsilon = e/E$ and $p = f/F$ then

$$E^n F^n l_n = \sum_{i=1}^{n} (EF)^{n-i} (E-e)^{i-1} f^{\overline{a_{i-1}}} (F-f)^{i-1-\overline{a_{i-1}}} [Fex_i + (1-a_i) f(E-e)] \in \mathbb{Z}.$$

By expanding the sum on the right the coefficients $\alpha_i$ for the key bits $x_i$, $0 \le i \le k$ can be determined. Any constant terms on the right can be moved to the left hand side and we can write:

$$L_n = \sum_{i=1}^{n} \alpha_i x_i$$

where $L_n = E^n F^n l_n - C$ (where $C$ is an integer constant) and each $\alpha_i$ is an integer. This is precisely the subset sum problem discussed in Section 2.

Given a finite key $x_1, x_2, \ldots, x_k$ and a message of more than $k$ symbols it is necessary to reuse the key information, otherwise the attacks presented in [6] can be applied.

Recall that the density of a subset sum is defined by (1). For interval narrowing the maximum size will be $O(E^n F^n)$, and the density will be given by $D \approx k/n \log_2(EF)$. Since the length of the key is fixed this density can be made arbitrarily small by sending a long enough message (large $n$). It

follows that the short-vector algorithm [16] can be used to determine the key bits.

Interval narrowing also suffers from chosen-plaintext attacks of the form $b^n$ or $a^n$. In keyless arithmetic coding $l_n$ does not change when the sequence $b^n$ is used, but $l_n$ changes in this system whenever $x_i = 1$, and this extra information can be used to determine the key. In an extreme case consider the result when $p = \frac{1}{2}$, $n = 5$, and $\epsilon = \frac{1}{2}$ with the message $b^5$. Then

$$2^9 l_5 = 2^8 + 2^6 + 2^4 + 2^2 + 1 + 2^9 x_1 + 2^7 x_2 + 2^5 x_3 + 2^3 x_4 + 2x_5$$

and it is possible to directly find the key bits from the odd numbered bits in $2^9 l_5$. Note: the coefficients here form a superincreasing sequence and the density is $0.\dot{5}$.

More generally if the sequence $b^n$ is used, then

$$E^n F^n l_n = \sum_{i=1}^{n} E^{n-i} F^{n-i} (E-e)^{i-1} (F-f)^{i-1} F e x_i.$$

Ignoring the value of the key bits and considering the ratio of two consecutive terms of this sum we find

$$\frac{(EF)^{n-i}(E-e)^{i-1}(F-f)^{i-1}Fe}{E^{n-i+1}F^{n-i+1}(E-e)^i(F-f)^i F e x_i} = \frac{1}{EF(E-e)(F-f)}.$$

Since $E > e \geq 1$ and $F > f \geq 1$ we have $EF(E-e)(F-f) \geq 4$. This is a sufficient condition for the coefficients of the sum to form a superincreasing sequence.

# 4 Generalizations and Discussion

It is easy to construct more complicated arithmetic coders in which the width of the coding interval as well as the upper and lower bounds depend on the key bits. Characteristic polynomials can also be produced for these arithmetic coders. They are more general in the sense that each term of the polynomial can contain arbitrary products of the key bits $x_1^{e_1} x_2^{e_2} \cdots x_k^{e_k} \in \{0, 1\}$, rather than just linear terms. Analysis in these situations is considerably more complex.

These generalizations may lead to problems which are harder than the subset sum problem. The products can take on a wide variety of forms—although, not every polynomial is possible. Which polynomials can be produced depends on the way that the key bits are used.

Several people have looked at the solvability of polynomials. In a seminal paper Matijasevič [18] showed that solvability of Diophantine equations in several variables is in general undecidable, thus answering Hilbert's tenth problem [12]. Later it was shown [20] that Diophantine equations with only thirteen unknowns are undecidable. Further, solving one quadratic equation in two variables over $\mathbb{N}$ is **NP**-complete [17]. Other solvability results include [10, 14]. Linear equations can be solved in deterministic polynomial time. These results are worst case complexities and only recently has there been much interest in the average complexity of this type of problem.

However, none of these results apply here because a solution is known to exist (namely, the result of the encoding) and the indeterminates only have two values 0 or 1, whereas the more general problems are over infinite sets. Further, in the case where there is more than one solution, the attacker must find all (or at least the majority) of the solutions to determine which solutions lead to likely plaintexts.

In the analysis we were very lenient on the attacker, and assumed the only unknown was the key. In practice, however, it is extremely unlikely that an attacker would know exactly the state of an adaptive compression model. The best current attacks against adaptive models involve flooding the model with very long chosen-plaintexts in an attempt to reduce the adaptive model to a known state [1, 2]. More sophisticated adaptive models might not be susceptible to this type of attack.

Despite the apparent increase in the complexity that is gained by allowing more general polynomials in the coding intervals it is difficult to see how this extra complexity could translate into greater security.

Together with the results presented in [6] these results strongly suggest that if compression is going to be used for encryption then the security must come from the model rather than the arithmetic coder. To this end we are continuing our research by investigating the cryptographic properties of various kinds of models used in practical compression systems.

# References


[1] Helen A. Bergen & James M. Hogan, "Data security in a fixed-model arithmetic coding compression algorithm," *Computers & Security,* **11,** pp. 445–461, 1992.

[2] Helen A. Bergen & James M. Hogan, "A chosen plaintext attack on an adaptive arithmetic coding compression algorithm," *Computers & Security,* **12,** pp. 157–167, 1993.

[3] E. F. Brickell & A. M. Odlyzko, "Cryptanalysis: A Survey of Recent Results," *Proc. of the IEEE,* **76,** 5, pp. 578–593, 1988.

[4] E. F. Brickell & A. M. Odlyzko, "Cryptanalysis: A Survey of Recent Results," *Contemporary Cryptology: The Science of Information Integrity,* G. Simmons (ed.), IEEE PRess, pp. 501–540, 1991.

[5] E. F. Brickell, "The cryptanalysis of knapsack cryptosystems," *Applications of Discrete Mathematics,* R. D. Ringeisen & F. S. Roberts (eds.), Society for Industrial and Applied Mathematics, pp. 3–23, 1988.

[6] J. Cleary, S. Irvine, I. Rinsma-Melchert, "On the Security of Arithmetic Coding," Technical Report, University of Waikato, 1994.

[7] B. Chor & R. L. Rivest, "A knapsack type public-key cryptosystem based on arithmetic in finite fields," *Advances in Cryptology—CRYPTO '84,* Springer-Verlag, pp. 54–65, 1985.

[8] Y. Desmedt, "What happened with knapsack cryptographic schemes," *Performance Limits in Communication, Theory and Practice,* NATO ASI Series E: Applied Sciences, **142,** Kluwer Academic Publishers, pp. 113–134, 1988.

[9] C. Deavours, "Unicity Points in Cryptanalysis," *Cryptologia,* **1,** 1, pp. 46–48, 1977.

[10] Aviezri S. Fraenkel & Yacov Yesha, "Complexity of Solving Algebraic Equations," *Info. Proc. Letters,* **10,** 4 and 5, pp. 178–179, 1980.

[11] M. E. Hellman, "The mathematics of public-key cryptography," *Scientific American,* **241,** 8, pp. 146–157, 1979.

[12] D. Hilbert, "Mathematische Probleme: Vortrag gehalten auf dem internationalen Mathematiker-Kongress zu Paris, 1970," *Nachr. Akad. Wiss, Göttingen, Math.-Phys. Kl. II,* pp. 253–297, 1900.

[13] D. A. Huffman, A method for the construction of minimum-redundancy codes, *Proc. Inst. Electr. Radio. Eng.,* **40,** 9, pp. 1098–1101, 1952.

[14] Richard M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations,* Raymond E. Miller & James W. Thatcher (eds.), Plenum Press, NY, 1972.

[15] A. K. Lenstra, H. W. Lenstra, Jr. & L. Lovász, "Factoring Polynomials with Rational Coefficients," *Mathematische Annalen,* **261,** pp. 515–534, 1982.

[16] J. C. Lagarias & A. M. Odlyzko, "Solving low-density subset sum problems," *J ACM,* **32,** 1, pp. 229–246, 1985.

[17] Kenneth L. Manders & Leonard Adleman, "NP-complete decision problems for binary quadratics," *J. of Comp. & Sys. Sc.* **16,** pp. 168–184, 1978.

[18] Ju. V. Matijasevič, "Enumerable sets are Diophantine," *Dokl. Akad. Nauk SSSR* **191,** pp. 279–282 (in Russian), *Soviet Math. Dokl.* **11,** pp. 354–357 (in English), 1970.

[19] R. C. Merkle & M. E. Hellman, "Hiding information and signatures in trap-door knapsacks," *IEEE Trans. Info. Th.,* **IT-24,** pp. 525–530, 1978.

[20] Ju. V. Matijasevič & Julia Robinson, "Reduction of an arbitrary Diophantine equation to one in 13 unknowns," *Acta Arithmetica,* **XXVII,** pp. 521–553, 1975.

[21] W. Patterson, *Mathematical Cryptology for Computer Scientists and Mathematicians,* Rowman and Littlefield, 1987.

[22] C. P. Pfleeger, *Security in Computing,* Prentice-Hall, 1989.

[23] A. Shamir, "A fast signature scheme," MIT Library for Computer Science, Technical Memorandum MIT/LCS/TM-107, 1978.

[24] A. Shamir, "On the cryptocomplexity of knapsack systems," *Proc. of the 11th ACM Symposium on Theory of Computing,* pp. 118–129, 1979.

[25] A. Shamir, "A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem," *Proc. of the 23rd IEEE Symposium on the Foundations of Computer Science,* pp. 145–152, 1982.

[26] William J. Wilson, "Chinks in the armor of public key cryptosystems," University of Waikato Technical Report 94/3, March 1994.

[27] Ian H. Witten, Radford M. Neal, & John G. Cleary, "Arithmetic coding for data compression," *Communications ACM,* **30,** 6, pp. 520–540, 1987.