A Knapsack-Type Public Key Cryptosystem Based on Arithmetic in Finite Fields

BENNY CHOR AND RONALD L. RIVEST

Abstract —**A new knapsack-type public key cryptosystem is introduced. The system is based on a novel application of arithmetic in finite fields, following a construction by Bose and Chowla. By appropriately choosing the parameters, one can control the density of the resulting knapsack, which is the ratio between the number of elements in the knapsack and their size in bits. In particular, the density can be made high enough to foil "low-density" attacks against our system. At the moment, no attacks capable of "breaking" this system in a reasonable amount of time are known.**

I. Introduction

In 1976, Diffie and Hellman [11] introduced the idea of public key cryptography, in which two different keys are used: one for encryption and one for decryption. Each user keeps his decryption key secret while making the encryption key public, so it can be used by everyone wishing to send messages to him. A few months later, the first two implementations of public key cryptosystems (PKC's) were discovered: the Merkle–Hellman scheme [21] and the Rivest–Shamir–Adelman (RSA) scheme [26]. More PKC's have been proposed since that time. Most of these implementations[1] can be put into two categories:

a) PKC's based on hard number-theoretic problems (e.g., RSA [26], Rabin [24], Williams [31], Goldwasser–Micali [13]);
b) PKC's related to the knapsack problem (e.g., Merkle–Hellman [21], Shamir [30]).

While no efficient attacks against number-theoretic PKC's are known, several knapsack-type PKC's have been shown to be insecure. Most of those systems have a concealed "superincreasing" sequence. Shamir made the first successful attack on the basic Merkle–Hellman system [29]. Following his attack, other attacks against more complicated systems were proposed. In particular, Brickell [5] found a way to break the general Merkle–Hellman scheme. A different attack is the "low-density" attack of Lagarias and Odlyzko [17]. The density of a knapsack is defined as the ratio of the number of elements in it to the size (in bits) of these elements. The most interesting point about the Lagarias–Odlyzko attack is that it does not make any assumption about how the system was constructed, and thus could be applicable to any knapsack-type cryptosystem whose density is low (unlike, say, Shamir's attack which relies heavily on the superincreasing underlying sequence). A different low-density attack was proposed by Brickell [4], although it appears to be less effective in practice. As a result of these attacks, knapsack-type PKC's which either are based on superincreasing sequences or have very low density seem to be vulnerable.

Here we propose a new knapsack-type PKC which has high density and a completely different basis. The underlying construction makes use of a result due to Bose and Chowla [3] on unique representations of sums in "dense" finite sequences. To create the encryption–decryption keys in this construction, discrete logarithms in finite fields have to be computed. Once this is done, encryption is very fast (linear time) and decryption is reasonably fast (comparable to RSA). Hence creating the keys is the hard part. While there are no polynomial time algorithms known for taking discrete logarithms, practical algorithms (most notably the ones due to Pohlig and Hellman [23] and Coppersmith [9]) are known for some special cases. To demonstrate the feasibility of such cases, we have constructed a real-life instance of our cryptosystem in the finite field GF(197^{24}). (Readers who wish to experiment with the new cryptosystem can find this specific public key in [7].) We believe that a system of that size will foil both low-density and exhaustive search attacks. The running time for constructing the system was a few hours on a minicomputer. This time-consuming task is done only once by each user, so it is acceptable from a practical point of view.

We would like to remark that all known number-theoretic PKC's are at most as hard as factoring and hence are all reducible to the problem of taking discrete logarithms in composite moduli (see [2], [7], [19]). Should this discrete logarithm problem become tractable (thus rendering all "number-theoretic" PKC's insecure), our system will become easier to create for even larger size knapsacks.

The remainder of this paper is organized as follows: In Section II we discuss the knapsack problem and its

Manuscript received May 27, 1986; revised August 11, 1987. This work was supported in part by National Science Foundation Grant MCS-8006938. This paper was presented at Crypto '84, Santa Barbara, CA, August 1984.

B. Chor was with Bell Laboratories, Murray Hill, NJ. He is now with the Department of Computer Science, Technion–Israel Institute of Technology, Haifa 32000, Israel.

R. L. Rivest is with the Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139.

IEEE Log Number 8823612.

[1] With the exception of the McEliece system [20], which is based on error correcting codes.

0018-9448/88/0900-0901$01.00 ©1988 IEEE

use in cryptosystems. Section III is a description of the Bose–Chowla theorem and its proof. In Section IV we give the details of our new cryptosystem. In Section V the system performance is examined, and Section VI describes the actual parameters for implementing our PKC. Finally, some possible attacks against the new system are analyzed in Section VII.

II. Knapsack-Type Cryptosystems

The 0–1 knapsack problem is the following NP-complete [12] decision problem. given a set $A = \{a_i | 0 \leq i \leq n-1\}$ of nonnegative integers and a nonnegative integer S, is there an integer solution to $\sum x_i a_i = S$ where all x_i are 0 or 1? A variant of the problem is to remove the 0–1 restriction on the x_i (while requiring them to be nonnegative integers) and bounding their *weight* $\sum x_i \leq h$.

Knapsack-type public-key cryptosystems are based on the intractability of finding a solution to $S = \sum x_i a_i$ even when a solution is known to exist. In such systems each user publishes a set A of a_i and a bound h. A plain text message consisting of an integer vector $M = (x_0, x_1, \cdots, x_{n-1})$ with weight $\leq h$ is encrypted by setting

$$E(M) = \sum x_i a_i.$$

The knapsack elements a_i are chosen in such way that the equation is easily solved if certain secret *trapdoor* information is known. The exact nature of this information depends on the particular system in question. A general property of knapsack-type PKC's is that encryption is easy —all you have to do is to add.

III. Bose–Chowla Theorem

In 1936, Sidon raised the question of whether "dense" sequences exist whose h-fold sums are unique. *Given n and h, nonnegative integers, is there a sequence* $A = \{a_i | 0 \leq i \leq n-1\}$ *of nonnegative integers, such that all sums of exactly h elements (repetitions allowed) out of A are distinct?* It is easy to construct such sequences if the a_i are growing exponentially in n. For example, the sequence $\{1, h, h^2, \cdots, h^{n-1}\}$ has the aforementioned property (but does not work even for $h+1$ element sums, since $h^2 + h \cdot 1 = (h+1) \cdot h$). However, can one construct such sequences with the a_i growing only polynomially fast in n? Bose and Chowla [3] found a very elegant way of constructing such sequences with $1 \leq a_i \leq n^h - 1$ for all $0 \leq i \leq n-1$. (See Halberstram and Roth [14, ch. 2] for an overview of the subject.) Here we present a slightly modified version of Bose–Chowla theorem, which fits well with our cryptographic application.

Bose–Chowla Theorem: Let p be a prime and $h \geq 2$ an integer. Then there exists a sequence $A = \{a_i | 0 \leq i \leq p-1\}$ of integers such that

1) $1 \leq a_i \leq p^h - 1, \qquad i = 0, 1, \cdots, p-1$

2) if $(x_0, x_1, \cdots, x_{p-1})$ and $(y_0, y_1, \cdots, y_{p-1})$ are two distinct vectors with nonnegative integral coordinates and $\sum_{i=0}^{p-1} x_i, \sum_{i=0}^{p-1} y_i \leq h$, then $\sum_{i=0}^{p-1} x_i a_i \neq \sum_{i=0}^{p-1} y_i a_i$.

Proof: The construction takes place in the finite field $GF(p)$ and in its h-degree extension $GF(p^h)$ (for convenience, the elements of $GF(p)$ will be indexed by their lexicographic order). Let $t \in GF(p^h)$ be algebraic of degree h over $GF(p)$ (i.e., the minimal polynomial in $GF(p)[x]$ having t as its root is of degree h). Let g be a multiplicative generator (primitive element) of $GF(p^h)$ (that is, $GF(p^h)^* = \{g^e | 0 \leq e \leq p^h - 1\}$). Look at an additive shift by t of the base field $GF(p)$, namely, at the set

$$t + GF(p) = \{t + \alpha_i | \alpha_i \in GF(p)\} \subset GF(p^h).$$

Let $a_i = \log_g(t + \alpha_i)$ ($\alpha_i \in GF(p)$) be the logarithm of $t + \alpha_i$ to the base g in $GF(p^h)$. Then the a_i are all integers in the interval $[1, p^h - 1]$, and they satisfy the distinctness of h-fold sums. Suppose there are two vectors $\vec{x}, \vec{y}$ of nonnegative integers satisfying (1), (2), and (3):

$$(x_0, x_1, \cdots, x_{p-1}) \neq (y_0, y_1, \cdots, y_{p-1}) \tag{1}$$

$$\sum_{i=0}^{p-1} x_i, \sum_{i=0}^{p-1} y_i \leq h \tag{2}$$

$$\sum_{i=0}^{p-1} x_i a_i = \sum_{i=0}^{p-1} y_i a_i. \tag{3}$$

Then the following equality holds in $GF(p^h)$:

$$g^{\sum_{i=0}^{p-1} x_i a_i} = g^{\sum_{i=0}^{p-1} y_i a_i},$$

and so

$$\prod_{i=0}^{p-1} (g^{a_i})^{x_i} = \prod_{i=0}^{p-1} (g^{a_i})^{y_i}.$$

Using the quality $g^{a_i} = t + \alpha_i$ and considering only the nonzero x_i, y_i, we get

$$(t+\beta_1)^{x_1}(t+\beta_2)^{x_2} \cdots (t+\beta_l)^{x_l} = (t+\gamma_1)^{y_1}(t+\gamma_2)^{y_2} \cdots (t+\gamma_m)^{y_m}$$

where $\{\beta_1, \beta_2, \cdots, \beta_l\}$ and $\{\gamma_1, \gamma_2, \cdots, \gamma_m\}$ are two different nonempty subsets of $GF(p)$ with at most h elements each. Both sides of the last equation are thus distinct *monic* polynomials of degree $\leq h$ with coefficients in $GF(p)$, so, subtracting them, we get that *t is a root of a nonzero polynomial of degree* $\leq h-1$ *with coefficients in* $GF(p)$. This contradicts the fact that t is algebraic of degree h over $GF(p)$.

Remarks: 1) From the proof it is clear that l sums ($l \leq h$) of A are distinct not only over Z, but also modulo $p^h - 1$. 2) The requirement "p is a prime" can be replaced by "p is a prime power" with no change in the claim or its proof.

IV. How the Cryptosystem Is Constructed and Used

In this section we describe how the new cryptosystem is created and used. We start with an informal (and slightly simplified) description. Next, a step-by-step recipe for generating the cryptosystem, encrypting messages, and decrypting ciphertexts is given. Finally, we describe how to transform "regular" unconstrained bit strings into strings with a fixed weight.

A. The Cryptosystem

The first step is to pick a prime (or a prime power) p and h such that $\mathrm{GF}(p^h)$ is amenable to discrete logarithm computations. We leave p and h as unspecified parameters in this section and elaborate more on their exact choice in Section VI (the approximate magnitudes will be $p \approx 200$, $h \approx 25$). Once p and h are chosen, we pick $t \in \mathrm{GF}(p^h)$ of algebraic degree h over the base field, and a primitive element $g \in \mathrm{GF}(p^h)$ (both t and g are picked at random from the many possible candidates). Following Bose and Chowla, logarithms (to base g) of the p elements in $\mathrm{GF}(p)+t$ are computed. These p integers are then scrambled using a randomly chosen permutation. The scrambled integers are published. Together with p and h they constitute the public encryption key, while the elements t, g, and the unscrambling permutation constitute the secret decryption key. To encrypt a binary message of length p and weight h, a user adds the knapsack elements with 1 in the corresponding message location, and sends the sum. When the legitimate receiver gets a sum, he first raises the generator g to it, and expresses the result as a degree h polynomial in t over $\mathrm{GF}(p)$. The h roots of this polynomial are found by successive substitutions. Applying the inverse of the original permutation, the indices of the plain text having the bit 1 are recovered.

1) System Generation: a) Let p be a prime power, $h \leq p$ an integer such that discrete logarithms in $\mathrm{GF}(p^h)$ can be efficiently computed.

b) Pick a random $t \in \mathrm{GF}(p^h)$ that is algebraic of degree h over $\mathrm{GF}(p)$. This will be done by finding $f(t)$, a random irreducible monic polynomial of degree h in $\mathrm{GF}(p)[t]$, and representing $\mathrm{GF}(p^h)$ arithmetic by $\mathrm{GF}(p)[t]/\langle f(t)\rangle$. (That is, elements of $\mathrm{GF}(p^h)$ are polynomials of degree $\leq h-1$ with coefficients in $\mathrm{GF}(p)$, and addition/multiplication operations are done modulo p and $f(t)$.)

c) Pick $g \in \mathrm{GF}(p^h)$, g a multiplicative generator of $\mathrm{GF}(p^h)$, *at random*. This will be done by picking a random $r \in \mathrm{GF}(p^h)$ until one which satisfies $r^{(p^h-1)/s} \neq 1$ (for all prime factors s of p^h-1) is found. Note that in our system p^h-1 will have only *small* prime divisors, and so it is easy to verify that a given r passes the above test. Since the density of such generators is relatively high in all cases (regardless of any special properties of p and h), the foregoing procedure is indeed feasible.

d) Construction following Bose-Chowla Theorem: Compute $a_i = \log_g(t+\alpha_i)$ for all $\alpha_i \in \mathrm{GF}(p)$.

e) Scramble the a_i: let π: $\{0,1,\cdots,p-1\} \rightarrow \{0,1,\cdots,p-1\}$ be a randomly chosen permutation. Set $b_i = a_{\pi(i)}$.

f) Add some noise: pick $0 \leq d \leq p^h-2$ at random. Set $c_i = b_i + d$.

g) *Public Key—to be Published*: We have $c_0, c_1, \cdots, c_{p-1}$; p, h.

h) *Private Key—to be Kept Secret*: We have t, g, π^{-1}, d.

Remark: Every user can use the *same* p and h. The probability of collisions (two users having the same keys) is negligible.

2) Encryption: To encrypt a binary message $M = (x_0, \cdots, x_{p-1})$ of length p and weight (number of 1's) *exactly* h, add the c_i whose corresponding bit is 1. Send

$$E(M) = \sum_{i=0}^{p-1} x_i c_i \pmod{p^h - 1}.$$

3) Decryption: a) Let $r(t) = t^h$ modulo $f(t)$, a polynomial of degree $\leq h-1$ (computed once at system generation).

b) Given $s = E(M)$, compute $s' = s - hd$ (modulo $p^h - 1$).

c) Compute $q(t) = g^{s'}$ modulo $f(t)$, a formal polynomial of degree $h-1$ in the variable t.

d) Add $t^h - r(t)$ to $q(t)$ to get $s(t) = t^h + q(t) - r(t)$, a polynomial of degree h in $\mathrm{GF}(p)[t]$.

e) We now have

$$s(t) = (t+\alpha_{i_1})\cdot(t+\alpha_{i_2})\cdots(t+\alpha_{i_h})$$

name, $s(t)$ factors to *linear terms* over $\mathrm{GF}(p)$. By successive substitutions, we find the h roots α_{i_j} (at most p substitutions needed). Apply π^{-1} to recover the coordinates of the original M having the bit 1.

B. Transforming Unconstrained Bit Strings

We have assumed until now that the message space M contains binary vectors of length p and weight h. However, regular binary text does not have this form. This subsection describes a simple procedure for translating unconstrained binary text into the aforementioned form.

Given a binary text, we first break it into blocks of $\left\lfloor \log_2 \binom{p}{h} \right\rfloor$ bits each. Each of these blocks is viewed as the binary representation of a number n, $0 \leq n < \binom{p}{h}$. To map these numbers into weight h binary vectors, we use the order-preserving mapping induced by the lexicographic order of the vectors and the natural order of the integers. If n is larger than $\binom{p-1}{h-1}$, the first bit in the corresponding vector is set to 1. Otherwise, the first bit is set to 0. We then update p and h, and iterate p times, until all p bits are determined:

code for transforming a number n into a binary vector $\vec{y}$:

```
Input: $n, p, h$; Output: $\vec{y}$
for $i \leftarrow 1$ to $p$ do
    if $n \geq \binom{p-i}{h}$ then
        $y_i \leftarrow 1$
        $n \leftarrow n - \binom{p-i}{h-1}$
        $h \leftarrow h - 1$
    else $y_i \leftarrow 0$
return $\vec{y}$.
```

The inverse transformation, which is the last step in decryption, is just as simple:

```
code for transforming a binary vector $\vec{y}$ into a number $n$:
    Input: $\vec{y}, p, h$; Output: $n$
    $n \leftarrow 0$
    for $i \leftarrow 1$ to $p$ do
        if $y_i = 1$ then
            $n \leftarrow n + \binom{p-i}{h}$
            $h \leftarrow h - 1$
    return $n$
```

For efficient implementation, the $p \cdot h/4$ binomial coefficients preceding $\binom{p}{h}$ (in the Pascal triangle) will be precomputed and permanently stored.

Remark: The previous indexing scheme is well-known in the literature (see, e.g. [10]).

V. System Performance: Time, Space, and Information Rate

In this section we analyze three basic parameters of the cryptosystem: the time needed for encrypting and decrypting a message, the size of the keys, and the information rate in terms of clear text bits per cipher text bits. The complexity of key generation is discussed in Section VI.

Given a binary message length p and weight h, encrypting it amounts to adding h integers c_i, each smaller than p^h. The run time for decryption is much longer. It is dominated by the modular exponentiation. To raise a polynomial g to a power in the range $[1, p^h - 1]$ takes at most $2h \log p$ modular multiplications. The modulus is $f(t)$, a polynomial of degree h, with coefficients in $GF(p)$. Using the naive polynomial multiplication algorithm, $2h^2$ operations (in $GF(p)$) per modular multiplication will suffice. Therefore, overall, $4h^3 \log p$ operations in $GF(p)$ are required. For the proposed parameters $p \approx 200$, $h \approx 25$ this gives about 500000 $GF(p)$ operations and compares favorably with RSA encryption–decryption time.

The size of the keys, and especially of the public key, is an important factor in the design of any public key system. In our system the size of the public key is that of p numbers, each in the range $[1, p^h - 1]$. In terms of bits, this is $p \log_2 p^h = ph \log_2 p$ bits. For $p \approx 200$, $h \approx 25$, the key takes less than 40000 bits. While this number is about 35 times larger than the currently proposed size for the RSA public key (600 bits for the modulus and 600 for the exponent), it is still within practical bounds.

The information rate R of a block code is defined as $R = (\log_2 |M|)/N$, where $|M|$ is the size of the message space, and N is the number of bits in a cipher text. Letting M range over all binary vectors of length p and weight h we have $|M| = \binom{p}{h}$. Also $N = \log_2 p^h$, and thus the information rate is

$$R = \frac{\log \binom{p}{h}}{\log p^h}$$

for the proposed parameters $p = 197$, $h = 24$, $R = 0.556$ (data expansion 1.798).

VI. Proposed Parameters and Implementation Details

As mentioned before, the main difficulty in implementing our cryptosystem is the computation of discrete logarithms in large finite fields $GF(p^h)$. This computational problem is considered quite hard in general. However, for some special cases the algorithms of Coppersmith [9] and Pohlig and Hellman [23] work well in practice. Coppersmith's algorithm is appropriate for fields of small characteristic, and performs best in characteristic 2. Letting $p^h = 2^n$, the run time of the algorithm is $e^{O\left(\sqrt[3]{n \log^2 n}\right)}$. For $n \leq 200$, implementation of the Coppersmith algorithm will terminate in a few hours on a mainframe computer. The Pohlig–Hellman algorithm works for any characteristic, provided $p^h - 1$ has only small prime factors. It turns out that the Pohlig–Hellman algorithm is preferable for our specific application, because of two properties: the nice factorization of several numbers $p^h - 1$ of appropriate magnitude, and the simplicity of the algorithm.

The Pohlig–Hellman algorithm has a $T \cdot S$ (time·space) complexity proportional to the largest factor of $p^h - 1$. While, in general, numbers whose order of magnitude is $\approx 200^{25}$ do not have 'small' largest factors (the expected size of the largest factor of a random number m is about $m^{0.6}$—see Knuth and Pardo [16]), things are much better when the number has the form $x^h - 1$ since we can first factor this expression as a polynomial in x and then factor each term as a number after substituting $x \leftarrow p$. Numbers h's with "good" factorization are especially effective. For example, $x^{24} - 1$ has the factors $x^8 - x^4 + 1$, $x^4 - x^2 + 1$, $x^4 + 1$, and other terms of degree not exceeding 2. Substituting $p = 197$, the largest prime factor of $197^{24} - 1$ is $10316017 \approx 10^7$. The square root of this is $3 \cdot 10^3$, so the Pohlig–Hellman algorithm can easily be implemented on a minicomputer within a few CPU hours for all the 197 logarithms.

Other possible values are (the last two values are from [6]):

- $p = 211$, $h = 24$ (largest prime factor of $211^{24} - 1$ is $216330241 \approx 2 \cdot 10^8$);
- $p = 243 = 3^5$, $h = 24$ (largest prime factor of $3^{120} - 1$ is $47763361 \approx 5 \cdot 10^7$);
- $p = 256 = 2^8$, $h = 25$ (largest prime factor of $2^{200} - 1$ is $3173389601 \approx 3 \cdot 10^9$).

The last candidate has the advantage that the field is of characteristic 2. Thus binary arithmetic can be used for key generation and decryption calculations. In addition, binary arithmetic offers easier implementation in special-purpose hardware.

We have implemented the key generation step in $GF(197^{24})$ on a Symbolics 3600 Lisp Machine. Polynomials were represented as arrays, and some preprocessing was done to speed-up the field arithmetic. In the implementation of the Pohlig–Hellman algorithm, instead of sorting the precomputed powers, we hashed them in a 197-by-197 array according to the free term and the coefficient of t in the polynomial. In this way the matching trials were simplified.

The overall run time for finding all 197 logarithms in $GF(197^{24})$ was about 8 hours. With some simple modifications we expect that the time can be reduced by 30 percent. It seems that even for $GF(256^{25})$ the computation should be feasible, taking advantage of the binary operations in the polynomial arithmetic. All these estimates can be drastically reduced if the computation is to be carried out on a faster larger computer using a programming language more suitable for numerical calculations (e.g., Fortran).

VII. Possible Attacks

In this section we examine some possible attacks on the cryptosystem. We start with specialized attacks on the cryptosystem, where the cryptanalyst is trying to reconstruct the secret key (possibly with some partial knowledge of it). We proceed by considering low-density and brute-force attacks with no prior secret information, where the goal is not to reconstruct the secret key but rather to decipher a given cipher text.

A. Specialized Attacks

We begin by assuming that certain parameters are known to the cryptanalyst. In Section VII-A5 we assume nothing is known.

1) Known g and d: Given d, compute $\{b_0, b_1, \cdots, b_{p-1}\} = \{c_0 - d, c_1 - d, \cdots, c_{p-1} - d\}$. Let $t' = g^{b_0}$. Since $g^{a_0} - g^{b_0} = t - t' \in GF(p)$, the sets $\{t + \alpha_i | \alpha_i \in GF(p)\}$ and $\{t' + \alpha_i | \alpha_i \in GF(\mathrm{p})\}$ are identical. Thus for every $\alpha_i \in GF(p)$ there is a unique $\alpha_{\sigma(i)} \in GF(p)$ so that $g^{b_{\sigma(i)}} = t' + \alpha_i$. Using t', g, σ, and d, the cryptanalyst can perform the same decryption algorithm as the legitimate receiver.

2) Known t and d: Pick an arbitrary generator g'. Compute $a_i' = \log_{g'}(t + \alpha_i)$. As sets, we have

$$\begin{aligned}\{c_0 - d, c_1 - d, \cdots, c_{p-1} - d\} &= \{a_0, a_1, \cdots, a_{p-1}\}\\ &= L\{a_0', a_1', \cdots, a_{p-1}'\}\end{aligned}$$

where equality is modulo $p^h - 1$, the numbers L, $p^h - 1$ are relatively prime, and L satisfies $g = g'^L$. Once L is recovered we are done, for then $g = g'^L$, and we can reconstruct π and have all the pieces of the private key.

If one of the a_i' (a_0', say) is relatively prime to $p^h - 1$, then L is one of $a_j a_0'^{-1}$ (mod $p^h - 1$) for some $0 \leq j \leq p - 1$. Otherwise, the cryptanalyst can compute L modulo each of the prime power factors of $p^h - 1$ (which, by choice of p and h, are all small and, therefore, easy to find), and then combine them using the Chinese remainder theorem.

3) Known t (Attack Due to Oded Goldreich): Pick an arbitrary generator g'. Compute $a_i' = \log_{g'}(t + \alpha_i)$. As sets we have

$$\begin{aligned}\{c_0 - c_0, c_1 - c_0, \cdots, c_{p-1} - c_0\} &= \{a_0 - a_0, a_1 - a_0, \cdots, a_{p-1} - a_0\}\\ &= L\{a_0' - a_0', a_1' - a_0', \cdots, a_{p-1}' - a_0'\}\end{aligned}$$

and now it is possible to proceed as in (b).

4) Known Permutation π (Attack Due to Andrew Odlyzko): Subtracting as in 3), we get the integers $a_i - a_0$ for $i = 1, \cdots, p - 1$. Since the knapsack is dense, there are small integral coefficients x_i (some of which may be negative) such that

$$\sum_{i=1}^{p-1} x_i(a_i - a_0) = 0$$

(see the Appendix and Section VII-A5 for a justification of this claim). The x_i can be efficiently found by applying the Lenstra–Lenstra–Lovasz (LLL) basis reduction algorithm [18] to the truncated Lagarias–Odlyzko lattice (see the Appendix and [22] for a similar attack on other knapsack schemes). Raising g to both sides of the last equality, we get

$$g^{\sum_{i=1}^{p-1} x_i(a_i - a_0)} = 1$$

i.e.,

$$\prod_{i=1}^{p-1} (t + i)^{x_i} = t^{\sum_{i=1}^{p-1} x_i}.$$

Let $m_1 = |\sum x_i^+|$ ($m_2 = \sum x_i^-|$) denote the sum of positive (negative) x_i, and $m = \max(m_1, m_2)$. The left side of the last equality is a rational function of t, while the right side has the form $t^{m_1 - m_2}$. The generator g, which is still unknown, is not a part of the equality. Multiplying through, we get a polynomial equation $r(t) = 0$ of degree $m - 1$ in t, with coefficients from $GF(p)$. Since the x_i are small, m is also not too large. All roots (in $GF(p^h)$) of this polynomial can be found using a fast probabilistic algorithm. The element t is necessarily one of these roots, so attack 3) can now be used.

The most efficient way for root finding which we know of (Rabin [25]) requires finding the gcd of $r(t)$ and $t^{p^h - 1}$. With $p^h - 1 \gg m$, this polynomial gcd computation is performed by raising t to the power $p^h - 1$ and reducing modulo $r(t)$. Therefore, we basically have to perform $h \log_2 p$ multiplications of m degree polynomials with coefficients in $GF(p)$, reducing modulo $r(t)$ each time. Assuming standard arithmetic, each polynomial multiplication will take $m^2 GF(p)$ operations (FFT arithmetic (see, e.g. [1, ch. 7]) will introduce a large constant and will probably be less efficient in practice). Thus the root-finding algorithm will require at least $m^2 h \log_2 p$ operations in $GF(p)$ (assuming that a single root is found).

Remark: If π is not known, this attack does not seem to work since even though the x_i can be found, they give rise to an 'unknown' polynomial. If $m_1 + m_2$ is very small, then one can try all $\binom{p}{m_1 + m_2}$ possibilities even without knowing π. However, with π unknown and $m_1 + m_2$ exceeding 10, this brute force approach becomes unfeasible. A more refined method for dealing with unknown π is presented next.

5) Nothing Known (Attack Due to Ernest Brickell): This attack is a strengthening of Odlyzko's attack. The goal is again to find a small degree equation satisfied by g. Using a carefully designed lattice, it is possible to find integer coefficients x_i, many of them 0, such that both equations

$$\sum_{i=0}^{p-1} x_i c_i \equiv 0 \pmod{p^h - 1}$$

$$\sum_{i=0}^{p-1} x_i \equiv 0 \pmod{p^h - 1}$$

hold. The second equality guarantees that

$$\begin{aligned} g^{\sum_{i=0}^{p-1} x_i c_i} &= g^{\sum_{i=0}^{p-1} x_i (b_i + d)} \\ &= g^{\sum_{i=0}^{p-1} x_i b_i} \cdot g^{d \sum_{i=0}^{p-1} x_i} \\ &= g^{\sum_{i=0}^{p-1} x_i b_i} \end{aligned}$$

and thus by the first equality,

$$g^{\sum_{i=0}^{p-1} x_i b_i} = 1$$

that is,

$$g^{\sum_{i=0}^{p-1} x_i a_{\pi(i)}} = 1.$$

With the permutation π unknown, this equality now means that

$$\prod_{i=0}^{p-1} \left(t + \alpha_{\pi(i)}\right)^{x_i} = 1.$$

Let x_i^+ denote x_i if $x_i \geq 0$; 0, otherwise (similarly, x_i^- denote $-x_i$ if $x_i \leq 0$; 0, otherwise). Let $r_\pi(t) = \prod_{i=0}^{p-1}(t + \alpha_{\pi(i)})^{x_i^+} - \prod_{i=0}^{p-1}(t + \alpha_{\pi(i)})^{x_i^-}$, and l be the number of nonzero x_i. Every one-to-one mapping from the set $\{i | x_i \neq 0\}$ into GF(p) gives rise to a different polynomial in $r_\pi(t)$. Only the "correct" polynomial will have the right t as one of its roots. Thus on the average we have to try $(1/2)(p!/(p-l)!)$ mappings to come up with the right polynomial. (In fact, it suffices to consider the mappings of only $l-1$ elements, since having $t + \alpha_j$ for some $\alpha_j \in$ GF(p) is as good as having t itself, and so $(1/2)(p!/(p-l+1)!)$ mappings are to be checked.)

For every such mapping, the cryptanalyst should find the roots in GF(p^h) of $r_\pi(t)$, an m degree polynomial with coefficients from GF(p) (m denotes here the same quantity as in Section VII-A4). Combining the foregoing calculations with the running-time estimates for polynomial arithmetic, the expected running time for recovering t will thus be $(1/2)(p!/(p-l+1)!) \cdot m^2 h \log_2 p$ (GF(p) operations). There is trade-off between l and m. Consider the l nonzero x_i. How large do they have to be to have a nonzero solution to $\sum x_i a_i \equiv 0$ (modulo $p^h - 1$)? If the x_i are in the range $-B/2 < x_i < B/2$, then this gives us B^l combinations of $\sum x_i a_i$. If $B^l > p^h - 1$, then two of these combinations *must* be the same modulo $p^h - 1$. In fact, if $B^l > p^{h/2}$, then by the birthday paradox two of the B^l sums are going to be equal modulo $p^h - 1$ with probability no less than $1/2$. Assuming the latter bound, $B \approx p^{h/2l}$. To get a nonzero combination $\sum x_i a_i \equiv 0$ (modulo $p^h - 1$), we subtract two combinations with the same sum. The average absolute value of the resulting x_i is about $B/2 = p^{h/2l}/2$, and about half $(l/2)$ of them are negative. Thus the sum of negative (positive) x_i is $m \sim lp^{h/2l}/4$. The total running time for finding t is thus

$$\begin{aligned} &\frac{1}{2} \frac{p!}{(p-l+1)!} \cdot m^2 h \log_2 p \\ &\sim \frac{1}{2} \frac{p!}{(p-l+1)!} \cdot \left(\frac{lp^{h/2l}}{4}\right)^2 h \log_2 p \\ &\sim \frac{p^{l-1} p^{h/l} l^2 h \log_2 p}{32}. \end{aligned}$$

This expression is optimized with $l = \theta(\sqrt{h})$, resulting in an $O(p^{2\sqrt{h}} h^2 \log_2 p)$ algorithm for retrieving t. While this expression is asymptotically superior to all other methods already mentioned, it seems quite prohibitive in practice. For example, taking $p = 197$ and $h = 24$, $l - 1 + h/l$ is optimized at $l = 5$, yielding 8.8. However, $197^{8.8} > 2^{60}$, so the attack is impractical. (Even assuming FFT arithmetic and complexity $O(m)$ for the root finding algorithm, the resulting expression $p^{l-1+h/2l} lh \log_2 p$ is optimized at $l = 3$, yielding about 2^{52}. The extra log factors of the FFT and the hidden constant will drive the expression up to at least 2^{58}.)

B. Low-Density Attacks

The *density* $d(A)$ of a knapsack system $A = \{a_i | 0 \leq i \leq p-1\}$, is defined to be

$$d(A) = \frac{p}{\log_2 (\max a_i)}.$$

Lagarias and Odlyzko [17] have devised an attack which is effective against low-density knapsacks. Given a knapsack system $A = \{a_i | 0 \leq i \leq p-1\}$ and a sum instance (cipher text) $S = \sum_{i=0}^{p-1} x_i a_i$, they construct a $(p+1)$-dimensional lattice (see the'Appendix for details). The lattice construction uses the p knapsack elements and the given cipher text. A certain vector in this lattice (which we call here the *special vector*) is defined. This vector corresponds to the solution of the given cipher text (yields the coefficients x_i in the sum), and the goal of the cryptanalyst is to find it. Lagarias and Odlyzko have shown that if $d(A)$ is low, this special vector is likely to be the shortest one in their lattice. Using the last observation, what Lagarias and Odlyzko are trying to do is to find the shortest vector in the lattice. Specifically, the tool they use is the lattice basis reduction algorithm of Lenstra *et al.* [18].

The success of the Lagarias and Odlyzko attack, when applied to a specific knapsack instance, depends on two factors. One factor is the density of the knapsack. If the density is too high, then one expects to have many extraneous short vectors. In this case, the attack will not work regardless of which short vector subroutine it uses. Even if the density is such that the special vector is indeed the shortest lattice vector, the attack is still not certain to succeed. In such case, its success is determined by the properties of the short vector subroutine. Present day algorithms either are guaranteed to find the shortest vector, but are inefficient, or are efficient (run in polynomial time), but are guaranteed only to find a relatively short vector, not necessarily the shortest one. To find the shortest vector, short vector algorithms of the second type (e.g. [18], [27]) typically require that the shortest lattice vector be substantially shorter than other lattice vectors.

In our specific case, the knapsack has relatively high density. Thus the length (square of Euclidean norm) of the special vector will not be much shorter than the length of many other vectors (24 versus 40 for $p = 197$, $h = 24$—see the Appendix). Shortest vector algorithms of the first type will, therefore, find the special vector in this case. However, the best shortest vector algorithm known currently is the one of Kannan [15], and its performance is no better in our application than the brute-force attack sketched in Section VII-C. The proximity of the special vector length (24) to lengths of many other vectors (40) should prevent the efficient short vector algorithms of the second type from finding the special vector.

The last claim was confirmed experimentally. Odlyzko has tested the Lagarias and Odlyzko attack on a smaller knapsack created by us, using the LLL algorithm as the short vector subroutine. For the test case we generated an instance of the knapsack with parameters $p = 103$ and $h = 12$. For these parameters the density is 1.271. Using the calculations of [17], the length of the shortest nonspecial vector in the constructed lattice should be at least 17. However, the LLL algorithm did not find the special vector even when its length was only 5 (i.e., when only five knapsack elements were added together in the sum). Therefore, it seems that for the Lagarias–Odlyzko attack to be successful against our system it must use a significantly better short vector algorithm than the ones now available.

A few people (including Brickell, Coppersmith, Desmedt, and Odlyzko) have suggested a modification of the Lagarias and Odlyzko attack. It is intended to make use of the fact that in our cryptosystem the number of knapsack elements that are added together is h, which is substantially smaller than $n/2$ which is the typical number in general knapsack systems. This attack (see the Appendix for more details) increases the ratio between the lengths of the special vector and the rest of the vectors in the lattice. This gives a better chance of finding the special vector when a short vector subroutine of the second type is used.

It is possible to modify the cryptosystem so that several elements are used more than once. Provided that the total number of elements used (counted with their multiplicities) is h, such modification does not affect the efficiency of decryption. It has the advantage of increasing the length of the special vector. However, with the current state of short vector algorithms it looks like the last attack is not efficient enough to break the cryptosystem, and so such modification of the cryptosystem will not be necessary.

C. Brute-Force Attacks

Despite the sophistication of the previous attacks, none of them outperforms a careful brute-force attack (unless the cryptanalyst is supplied with some part of the secret decryption key). The most efficient method we know of for solving knapsack instances with h out of p items, given a specific cipher text, is the following. There are $\binom{p}{h}$ ways of choosing h out of p elements. Take a random subset S containing $p/2$ elements. The probability that a given sum contains exactly $h/2$ out of these $p/2$ elements is

$$\frac{\binom{p/2}{h/2}^2}{\binom{p}{h}} \approx \frac{1}{\sqrt{h}}.$$

Assuming that this is indeed the case, we generate all $h/2$ sums of S and of its complement, and sort them. The goal is to find a pair of sums from the two lists whose sum matches the desired target. This can be achieved by keeping two pointers on the two lists and marching linearly through each one (one in increasing order, and the other in decreasing order). If the two lists are exhausted but no matching sum found, then another random S is tried. The run time per one choice of S is dominated by sorting all $h/2$ sums of both S and its complement. This will require $2 \cdot \binom{p/2}{h/2} \ln \binom{p/2}{h/2}$ operations. On the average, about $\sqrt{h}$ choices of S have to be made. The overall expected running time will thus be

$$2 \cdot \binom{p/2}{h/2} \ln \binom{p/2}{h/2} \frac{\binom{p}{h}}{\binom{p/2}{h/2}^2} = \frac{2\binom{p}{h} \ln \binom{p/2}{h/2}}{\binom{p/2}{h/2}}.$$

For the proposed parameters $p = 197$, $h = 24$, the expected number of operations is $3.466 \cdot 10^{17} > 2^{58}$, so such brute force attack is impractical. The knapsack algorithm of Schroeppel and Shamir [28] might be used here for space efficiency. However, its run time behavior is no better than the above algorithm.

D. A Word of Caution

Even though none of these attacks seems to produce a serious threat to the system security, other attacks might be successful. We urge the reader to examine our proposal for as yet undiscovered weaknesses.

Acknowledgment

We wish to thank Ernie Brickel, Don Coppersmith, Oded Goldreich, Jeff Lagarias, and Andrew Odlyzko for many discussions concerning the system and possible attacks on it. Andrew's assistance in a first implementation of the system, and later in testing the low-density attack against it, was especially helpful. Oded was kind enough to decipher earlier versions of this manuscript, and his comments made it much clearer. We would also like to thank Scott Warner for his assistance in the final implementation of the system. Finally, thanks to Don Coppersmith and Victor Miller for acquainting us with [6] and [10].

Appendix

The Lagarias–Oklyzko Low-Density Attack

In this appendix we give a brief description of the Lagarias–Odlyzko "low-density" attack, which is based on finding short vectors in lattices. (A different "low-density" attack was proposed by Brickell [4].) Given a knapsack system $A = \{a_i | 0 \le i \le n-1\}$ and a sum instance $S = \sum_{i=0}^{n-1} x_i a_i$, the algorithm proceeds as follows.

a) Construct an $(n+1)$-dimensional integer lattice with basis vectors

$$
\begin{aligned}
\vec{v}_0 &= (1,0,0,\cdots,0,a_0)\\
\vec{v}_1 &= (0,1,0,\cdots,0,a_1)\\
\vec{v}_2 &= (0,0,1,\cdots,0,a_2)\\
&\vdots\\
\vec{v}_{n-1} &= (0,0,0,\cdots,1,a_{n-1})\\
\vec{v}_n &= (0,0,0,\cdots,0,-S).
\end{aligned}
$$

b) Look for the shortest nonzero vector $\vec{u}$ in this lattice. This step is using the LLL basis reduction algorithm which finds a relatively short vector in the lattice (not necessarily the shortest vector).

c) Check if $\vec{u} = \vec{x}$, where $\vec{x} = (x_0, x_1, x_2, \cdots, x_{n-1}, 0)$ is the special vector which deciphers S.

We call the lattice spanned by the $n+1$ basis vectors the *full Lagarias–Odlyzko lattice*, and the sublattice spanned by the first n basis vectors the *truncated* Lagarias–Odlyzko lattice. The truncated lattice does not depend on the actual sum S. The basic idea behind the algorithm is that since $S = \sum x_i a_i$, $\vec{x}$ is always a vector in the space spanned by the basis vectors. The Euclidean norm of $\vec{x}$ is $\sqrt{\sum_{i=0}^{n-1} x_i^2}$. Lagarias and Odlyzko show that for any constant $0 < \alpha \le 1/2$, there is a *critical density* d_α so that for almost all sets A (in an appropriate probability space) with density below d_α, and almost all subset sums of $\le \alpha n$ elements, there is a unique vector in the lattice spanned by $\vec{v}_0, \vec{v}_1, \cdots, \vec{v}_n$ with Euclidean norm not exceeding $\sqrt{\alpha n}$. (For example, $d_{1/2} = 0.645$.) Lagarias and Odlyzko conjectured that d_α is a "cut-off" density in the following sense: almost all knapsacks A with $d(A) > d_\alpha$ will have exponentially many vectors with Euclidean norm $\le \sqrt{\alpha \cdot n}$ in the truncated lattice. These short vectors correspond to small linear dependencies among the a_i—solutions to $\sum_{i=0}^{p-1} x_i a_i = 0$ with small integral x_i (not necessarily ± 1 or 0).

The modification of the Lagarias and Odlyzko attack, which makes use of the additional information that in our ciphertexts $\sum_{i=0}^{p-1} x_i = h$, is as follows. Add one more column to the basis elements in the Lagarias–Odlyzko lattice. For $i = 0, 1, \cdots, n-1$, $\vec{v}_i$ will have a large constant s in this additional entry, while $\vec{v}_n$ will contain $-hs$ in this entry ($s \approx 10$ suffices for our application). To be shorter than the special vector, a vector $\vec{y}$ in the truncated lattice must now satisfy both $\sum_{i=0}^{p-1} y_i = 0$ and $\sum_{i=0}^{p-1} y_i a_i = 0$. This should get rid of many extraneous short vectors.

The possible modification to the cryptosystem which is aimed at increasing the length of the special vector is the following. For 0–1 knapsack problems, the Euclidean norm of $\vec{x}$ is not too big since each coordinate contributes at most 1 to the sum. However, if the same item is taken more than once, the norm of $\vec{x}$ grows substantially. Thus by taking some elements with multiplicity greater than one, we can improve on the resistance of the system to low-density attacks.

References

[1] A. Aho, J. Hopcroft, and J. Ullman, *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley, 1974.

[2] E. Bach, "Discrete logarithms and factoring," Computer Science Division, Univ. of California, Berkeley, Tech. Rep. CSD UCB 84/186, 1984.

[3] R. C. Bose and S. Chowla, "Theorems in the additive theory of numbers," *Comment. Math. Helvet.*, vol. 37, pp. 141–147, 1962.

[4] E. F. Brickell, "Are most low density knapsacks solvable in polynomial time?," in *Proc. 14th Southeastern Conf. Combinatorics, Graph Theory and Computing, Cong. Numer.*, vol. 39, 1983, pp. 145–156.

[5] ——, "Breaking iterated knapsacks," *Advances in Cryptology: Proc. Crypto84*, G. R. Blakely and D. Chaum, Eds. New York: Springer-Verlag, 1985, pp. 342-358.

[6] J. Brillhart, D. H. Lehmer, J. L. Selfridge, B. Tuckerman, and S. S. Wagstaff, Jr., "Factorization of $b^n \pm 1$," in *Contemporary Mathematics*, vol. 22. Providence, RI: AMS, 1983.

[7] B. Chor, "Two issues in public-key cryptography," Ph.D. dissertation, Mass. Inst. Technol., Cambridge, 1986.

[8] B. Chor and R. L. Rivest, "A knapsack type public key cryptosystem based on arithmetic in finite fields (preliminary report)," in *Advances in Cryptology: Proc. Crypto84*, G. R. Blakely and D. Chaum, Eds. New York: Springer-Verlag, 1985, pp. 54–65.

[9] D. Coppersmith, "Fast evaluation of logarithms in fields of characteristic two," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 587–594, 1984.

[10] T. M. Cover, "Enumerative source encoding," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 73 77, 1973.

[11] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 644–654, 1976.

[12] M. Garey and D. Johnson, *Computers and Intractability*. New York: Freeman, 1979.

[13] S. Goldwasser and S. Micali, "Probabilistic encryption," *J. Comput. Syst. Sci.*, vol. 28, no. 2, pp. 270–299, 1984.

[14] H. Halberstram and K. F. Roth, *Sequences*. New York: Springer-Verlag, 1983.

[15] R. Kannan, "Improved algorithms for integer programming and related lattice problems," in *Proc. 15th Ann. Symp. Theory of Computing*, ACM, 1983, pp. 193–206.

[16] D. E. Knuth and L. T. Pardo, "Analysis of a simple factorization algorithm," *Theoret. Comput. Sci.*, vol. 3, no. 3, pp. 321–348, 1976.

[17] J. C. Lagarias and A. M. Odlyzko, "Solving low-density subset sum problems," *J. Ass. Comput. Mach.*, vol. 32, no. 1, pp. 229–246, Jan. 1985.

[18] A. K. Lenstra, H. W. Lenstra Jr., and L. Lovasz, "Factoring polynomials with rational coefficients," *Math. Ann.*, vol. 261, pp. 515–534, 1982.

[19] D. L. Long, "Random equivalence of factorization and computation of order," *Theoret. Comput. Sci.*, to be published.

[20] R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory," *DSN Progress Rep. 42-44*, pp. 114–116, 1978.

[21] R. C. Merkle and M. Hellman, "Hiding information and signatures in trap-door knapsacks," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 525–530, 1978.

[22] A. M. Odlyzko, "Cryptanalytic attacks on the multiplicative knapsack cryptosystem and on Shamir's fast signature scheme," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 594–601, 1984.

[23] R. C. Pohlig and M. Hellman, "An improved algorithm for computing logarithms over GF(p) and its cryptographic significance," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 106–110, 1978.

[24] M. O. Rabin, "Digitalized signatures and public-key functions as intractable as factorization," Laboratory for Computer Science, Mass. Inst. Technol., Cambridge, Tech. Rep. TR-212, 1979.

[25] ____, "Probabilistic algorithms in finite fields," *SIAM J. Comput.*, vol. 9, no. 2, pp. 273–280, 1980.

[26] R. L. Rivest, A. Shamir, and L. Adleman, "On digital signatures and public key cryptosystems," *Commun. ACM*, vol. 21, pp. 120–126, 1978.

[27] C. P. Schnorr, "A hierarchy of polynomial time basis reduction algorithms," manuscript, 1984.

[28] R. Schroeppel and A. Shamir, "A $T = O(2^{n/2})$, $S = O(2^{n/4})$ algorithm for certain NP-complete problems," *SIAM J. Comput.*, vol. 10, no. 3, pp. 456–464, 1981.

[29] A. Shamir, "A polynomial time algorithm for breaking the basic Merkle–Hellman cryptosystem," in *Proc. 23rd Ann. Symp. Foundations of Computer Science*. New York: IEEE, 1982, pp. 145–152.

[30] ____, "Embedding cryptographic trapdoors in arbitrary knapsack systems," Lab. for Computer Science, Mass. Inst. Technol., Cambridge, Tech. Memo. TM230, Sept. 1982.

[31] H. C. Williams, "A modification of the RSA public-key encryption procedure," *IEEE Trans. Inform. Theory*, vol. IT-26, pp. 726–729, 1980.