A Knapsack Type Public Key Cryptosystem Based On Arithmetic in Finite Fields

(preliminary draft)

Benny Chor Ronald L. Rivest

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Abstract—We introduce a new knapsack type public key cryptosystem. The system is based on a novel application of arithmetic in finite fields, following a construction by Bose and Chowla. Appropriately choosing the parameters, we can control the density of the resulting knapsack. In particular, the density can be made high enough to foil "low density" attacks against our system. At the moment, we do not know of any attacks capable of "breaking" this system in a reasonable amount of time.

1. INTRODUCTION

In 1976, Diffie and Hellman [7] 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 implementation of public key cryptosystems were discovered: The Merkle-Hellman scheme [13] and the Rivest-Shamir-Adelman scheme [17]. Some more PKC have been proposed since that time. Most of them can be put into two categories[1]:

a. PKC based on hard number-theoretic problems ([17],[16],[8]).

b. PKC related to the knapsack problem ([13],[2]).

While no efficient attacks against number theoretic PKC are known, some knapsack type PKC

† Research supported by NSF grant MCS-8006938. Part of this research was done while the first author was visiting Bell Laboratories, Murray Hill, NJ.

[1] with the exception of McEliece system [12], which is based on error correcting codes

G.R. Blakley and D. Chaum (Eds.): Advances in Cryptology - CRYPTO '84, LNCS 196, pp. 54-65, 1985.

were 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 (see [19]). Following his attack, other attacks against more complicated systems were proposed. The strongest of these seems to be the "low density" attack of Lagarias and Odlyzko [11]. The most interesting point about this last attack is that it does not make any assumption about how the system was constructed, and thus might be applicable to any knapsack type cryptosystem (unlike, say, Shamir's attack which relies heavily on the superincreasing underlying sequence). As a result of these attacks, knapsack type PKC which are either based on superincreasing sequences or have very low density seem to be vulnerable.

In this paper, 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 [1] about unique representation of sums in "dense" finite sequences. To implement this construction requires taking discrete logarithms in finite fields, for creating the encryption-decryption keys. 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, there are practical algorithms (most notably the ones due to Pohlig and Hellman [15] and Coppersmith [5]) in some special cases. We can demonstrate the existence of such special cases which would both yield reasonable size keys to foil the low density and exhaustive search attacks, and do so in reasonable amount of time (a few hours on a minicomputer, which is not too bad since keys are created only once per user). It should also be noticed that all known number theoretic PKC are at most as hard as factoring and hence are all reducible to the problem of taking discrete logarithms in composite moduli (see appendix 1). Should this discrete logarithm problem become tractable (thus rendering all "number-theoretic" PKC insecure), our system will become easier to create for even larger size knapsacks.

The remainder of this paper is organized as follows: In section 2 we discuss the knapsack problem and its use in cryptosystems. Section 3 describes Bose-Chowla theorem and its proof. In section 4 we give the details of our new cryptosystem. In section 5 the system performence is examined, and section 6 describes the actual parameters for implementing our PKC. Finally, some possible attacks against the new system are analyzed in section 7.

2. KNAPSACK-TYPE CRYPTOSYSTEMS

The $0-1$ knapsack problem is the following NP-complete decision problem: Given a set $A = \{a_i \mid 0 \leq i \leq n-1\}$ of non-negative integers and a non-negative integer S, is there an integer solution to $\sum x_i a_i = S$ where all x_i are 0 or 1. A different variant of the problem is

to remove the $0-1$ restriction on the x_i (but insisting they remain non-negative integers) and bounding their total 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 plaintext message consisting of an integer vector $M = (x_0, x_1, \ldots, x_{n-1})$ 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 is that encryption is easy - all you have to do is to add.

3. BOSE-CHOWLA THEOREM

In 1936, Sidon raised the question of whether there exist "dense" sequences whose h-fold sums are unique. *Given n and h, non-negative integers, is there a sequence $A = \{a_i \mid 0 \leq i \leq n-1\}$ of non-negative 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 $\{1, h, h^2, \ldots, h^{n-1}\}$ has the above property (but does not work even for $h+1$ element sums, since $h^2 + h \cdot 1 = (h+1) \cdot h$). But can one construct such sequence with the a_i growing only polynomially fast in n? Bose and Chowla [1] found a very elegant way of constructing such sequences with $1 \leq a_i \leq n^h - 1$ (see [9,ch.2] for an overview of the subject). Here, we'll present a slightly modified version of Bose-Chowla theorem, which will fit well our cryptographic application.

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

1. $1 \leq a_i \leq p^h - 1 \quad (i = 0, 1, \ldots, p-1)$.

2. If $(x_0, x_1, \ldots x_{p-1})$ and $(y_0, y_1, \ldots y_{p-1})$ are two distinct vectors with non-negative 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 $GF(p)$ and its h-degree extension, $GF(p^h)$. 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 $GF(p^h)$. Look at an additive shift by t of the base field, $GF(p)$, namely at the set $t + GF(p)\{t + i \mid i = 0, 1, \ldots, p-1\} \subseteq GF(p^h)$.

Let $a_i = \log_g(t+i)$ $(i = 0, 1\ldots, p-1)$ the logarithm of $t+i$ to the base g in $GF(p^h)$. Then the a_i are all in the interval $[1, p^h - 1]$ and they satisfy the distinctness of h-fold sums: For suppose there are two vectors $\vec{x}$, $\vec{y}$ with

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

$$\sum_{i=0}^{p-1} x_i, \sum_{i=0}^{p-1} y_i \leq h \quad \text{, and} \quad \sum_{i=0}^{p-1} x_i a_i = \sum_{i=0}^{p-1} y_i a_i .$$

$$\text{Then also} \quad g^{\sum_{i=0}^{p-1} x_i a_i} = g^{\sum_{i=0}^{p-1} y_i a_i}$$

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

Since $g^{a_i} = t+i$, we get

$$(t+i_1)^{x_1}(t+i_2)^{x_2}\ldots(t+i_l)^{x_l} = (t+j_1)^{y_1}(t+j_2)^{y_2}\ldots(t+j_k)^{y_k},$$

where $\{i_1, i_2, \ldots, i_l\}$ and $\{j_1, j_2, \ldots, j_k\}$ are two different non-empty sets of elements from $\{0, 1, \ldots, p-1\}$, with at most h elements each. Therefore, both sides of the last equation are monic distinct polynomials of degree $\leq h$ with coefficients in $GF(p)$, so we can subtract them and get:

t is a root of a non-zero polynomial, with coefficients in $GF(p)$, of degree $\leq h-1$.

This contradicts the fact that t is algebraic of degree h over $GF(p)$. ∎

Remarks:

1. From the above proof it is clear that l sums ($l \leq h$) of A are distinct not only over $\mathbf{Z}$, 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.

4. THE NEW CRYYPTOSYSTEM

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 cyphertexts is given.

The first step is to pick p and h such that $GF(p^h)$ is amenable for discrete logarithm computations. We leave p and h as unspecified parameters in this section, and elaborate more on their exact choice in section 7 (the approximate magnitudes will be $p \approx 200$, $h \approx 25$). Once p and h are chosen, we pick $t \in GF(p^h)$ of algebraic degree h over the base field, and a primitive element

$g \in GF(p^h)$ (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 $GF(p) + t$ 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 key.

In order 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. Section 6 deals with the question of transforming "regular", unconstrained binary strings to those of the above form.

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 $GF(p)$. The h roots of this polynomial are found by successive substitutions. Applying the inverse of the original permutation, the indices of the plaintext having the bit 1 are recovered.

a. System Generation

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

2. Pick $t \in GF(p^h)$ - t algebraic of degree h over $GF(p)$ *at random.* This will be done by finding $f(t)$, a random irreducible monic polynomial of degree h in $GF(p)[t]$, and representing $GF(p^h)$ arithmetic by $GF(p)[t]/ < f(t) >$ (where $< f(t) >$ is the ideal generated by $f(t)$).

3. Pick $g \in GF(p^h)$, g a multiplicative generator of $GF(p^h)$ *at random.*

4. Construction following Bose-Chowla theorem: Compute $a_i = \log_g(t+i)$ for $i = 0, 1, 2, \ldots, p-1$.

5. Scramble the a_i's: Let $\pi : \{0, 1, \ldots, p-1\} \rightarrow \{0, 1, \ldots, p-1\}$ be a randomly chosen permutation. Set $b_i = a_{\pi(i)}$.

6. Add some noise: Pick $0 \leq d \leq p^h - 2$ at random. Set $c_i = b_i + c$.

7. Public key - to be published: $c_0, c_1, \ldots, c_{p-1}$; p, h.

8. Private key - to be kept secret: t, g, π, d.

Note: Every user will have the same p and h. The probability of collisions (two users having the same keys) is negligible.

b. Encryption

To encrypt a binary message M of length p and weight (number of 1's) *exactly* h, add the c_i's whose corresponding bit is 1. Send

$$E(M) = c_{i_1} + c_{i_2} + \ldots + c_{i_h} \pmod{p^h - 1}.$$

c. Decryption

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

1. Given $s = E(M)$, compute $s' = s - hd \pmod{p^h - 1}$.

2. Compute $p(t) = g^{s'} \bmod f(t)$, a polynomial of degree $h-1$ in the formal variable t.

3. Add $t^h - r(t)$ to $p(t)$ to get $d(t) = t^h + p(t) - r(t)$, a polynomial of degree h in $GF(p)[t]$.

4. We now have

$$d(t) = (t + i_1) \cdot (t + i_2) \ldots (t + i_h)$$

namely $d(t)$ factors to *linear terms* over $GF(p)$. By successive substitutaions, we find the h roots i_j's (at most p substitutions needed). Apply π^{-1} to recover the coordinates of the original M having the bit 1.

5. SYSTEM PERFORMENCE: 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 cleartext bits per ciphertext bits.

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. So overall, $4h^3 \log p$ operations in $GF(p)$ are required. For the proposed parameters $p \approx 200$, $h \approx 25$ this gives about 500,000 $GF(p)$ 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 such system, a directory containing all public keys should be maintained such that each entry is easily accessible by every user. 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 40,000 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 $R = \frac{\log_2 |M|}{N}$, where $|M|$ is the size of the message space, and N is the number of bits in a ciphertext. Letting M range over all binary vectors of length p and weight h, $|M| = \binom{p}{h}$. $N = \log_2 p^h$, so the information rate is

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

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

6. PROPOSED PARAMETERS

As mentioned before, the main obstacle 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,the algorithms of Coppersmith [5] and Pohlig and Hellman [15] work well in practice for some special cases. Coppersmith 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 Coppersmith algorithm will terminate in a few hours on a mainframe computer. Pohlig-Hellman algorithm works for any characteristic, provided $p^h - 1$ has only small prime factors. It turns out that Pohlig-Hellman algorithm is preferable for our specific application, due to two properties: The simplicity of the algorithm, and the nice factorization of several numbers $p^h - 1$ of appropriate magnitude.

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 number m is about $m^{0.6}$), 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 $x \leftarrow p$. 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 10,316,017 $\approx 10^7$. The square root of this is $3 \cdot 10^3$, so 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 [4]):

- $p = 211$, $h = 24$ (largest prime factor of $211^{24} - 1$ is 216,330,241 $\approx 2 \cdot 10^8$)
- $p = 256 = 2^8$, $h = 25$ (largest prime factor of $2^{200} - 1$ is 3,173,389,601 $\approx 3 \cdot 10^9$). This candidate has the advantage of using binary arithmetic for decryption calculations.
- $p = 243 = 3^5$, $h = 24$ (largest prime factor of $3^{120} - 1$ is 47,763,361 $\approx 5 \cdot 10^7$).

7. POSSIBLE ATTACKS

In this section we examine some possible attacks on the cryptosystem. We start with attacks where part of the secret key is known to the cryptanalyst and he is trying to reconstruct the rest 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 ciphertext.

a. Known g and d.

Let $t' = g^{c_0}$, then $t - t' \in GF(p)$, so the sets $\{t+i | i \in GF(p)\}$ and $\{t'+i | i \in GF(p)\}$ are identical. Therefore, by using t', the cryptanalyst can determine π and has all needed information for decryption.

b. Known t and d.

Pick arbitrary generator g'. Compute $a'_i = \log_{g'}(t+i)$. As sets, we have

$$\{a_0, a_1, \ldots, a_{p-1}\} = L\{a'_0, a'_1, \ldots, a'_{p-1}\}$$

where equality is modulo $p^h - 1$ and $L, p^h - 1$ are relatively prime, L satisfying $g = g'^L$. 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[1] to $p^h - 1$, then L is one of $a_j a_0'^{-1} \pmod{p^h - 1}$, for some $0 \le j \le p-1$. Otherwise, the cryptanalyst can compute L modulo each of the prime power factors of $p^h - 1$ (which are all small and therefore easy to find, by the choice of p and h), and then combine them together using the Chinese remainder theorem.

c. Known permutation π and d (attack due to Andrew Odlyzko).

Since the knapsack is dense, there are small integral coefficients x_i (some of which may be negative) such that

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

(for details see [14]). Furthermore, the LLL algorithm can find these x_i's. The last equality implies

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

i.e.

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

The left hand side of the last equality is a rational function of t, and g (which is still unknown) is not a part of it. If $m_1 = \left|\sum x_i^+\right|$ ($m_2 = \left|\sum x_i^+\right|$) denotes the sum of positive (negative) x_i's, and $m = \max(m_1, m_2)$, then we get a polynomial equation of degree $m-1$ in t, with coefficients from $GF(p)$. All roots (in $GF(p^h)$) of this polynomial can be found using a fast probabilistic algorithm. t is necessarily one of these roots, so attack (b) can now be used.

[1] this means that one of the $t+i$ is itself a multiplicative generator of $GF(p^h)$, and will happen with high probability.

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 approach becomes infeasible.

d. Low density attacks

Brickel [3] and independently Lagarias and Odlyzko [11] introduced "general purpose" algorithms which can be expected to recover successfully the added elements of any "low density" knapsack system. In this subsection we briefly describe the second method, and examine its success when applied to our system.

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

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

Given a knapsack system $A = \{ a_i | \, 0 \leq i \leq p-1 \}$ and a sum instance (ciphertext) $S = \sum_{i=0}^{p-1} x_i a_i$, Lagarias and Odlyzko construct a $p+1$ dimensional lattice. The lattice construction uses the p knapsack elements and the given ciphertext. A certain vector in this lattice (which we call here the *special vector*) is defined. This vector corresponds to the solution of the given ciphertext (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 the shortest one in the lattice.

Using the last observation, what Lagarias and Odlyzko are trying to do is to find the shortest vactor in the lattice. The tool they use is the basis reduction algorithm of Lenstra, Lenstra and Lovasz. While this algorithm usually succeeds if the shortest vector in the lattice is much shorter than all other vectors, it does not do so well if the shortest vector is relatively close in length to other vectors.

In our specific case, the knapsack has high density. The length (square of Euclidean norm) of the specific vector will not be much shorter than the length of many other vectors (24 vs. 40 for $p = 197$, $h = 24$), and so the LLL algorithm cannot be expected to find it. Experiments, done by Andrew Odlyzko, on a smaller knapsack created by us ($p = 103$, $h = 12$, a system with density 1.271,where the calculations imply that all vectors other than the specific one have length at least 17, but the LLL algorithm did not find the specific vector even when its length was only 5), support this claim. So, for Lagarias-Odlyzko attack to be successful against our system, it must use a better shortest vector algorithm. Currently, the best (exact) shortest vector algorithm known is the one of Kanaan [10], and its performance is no better, in our application, than the brute force attack sketched in the next subsection.

We wish to remark that it is possible to make the specific vector which solves a given ciphertext *longer*, by reducing the information rate of the system, without changing its density (details are left to the full paper). In this case, no shortest vector algorithm will find this vector. However, with the current state of shortest vector algorithm, it looks like such modification to the cryptosystem is not required.

e. Brute force attacks.

The most efficient method we know of for solving knapsack instances with h out of p items, given a specific ciphertext, 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 to the two lists, and marching linearly through each (one in increasing order, and the other in decreasing order). If the two lists are exhausted but no matching sum was 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

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

For $p = 197$, $h = 24$ the expected number of operations is $3.466 \cdot 10^{17} > 2^{58}$, so such brute force attack is totally impractical[1].

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

ACKNOWLEDGEMENTS

We wish to thank 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

[1]the knapsack algorithm of Schroeppel and Shamir [18] might be used here as well for space efficiency. However, its run time behavior is no better than the above algorithm.

first implementation of the system, and later in testing the low density attack against it, was especially helpful. Oded was kind enough to decipher a few earlier versions of this manuscript, and his comments made it much clearer. Finally, thanks to Don Coppersmith and Victor Miller for acquainting us with [4] and [6].

REFERENCES

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

[2] Brickell, E.F., "A new knapsack based cryptosystem", Presented in Crypto83.

[3] Brickell, E.F., "Are most low density knapsacks solvable in polynomial time?", *Proceedings of the Fourteenth Southeastern Conference on Combinatorics, Graph Theory and Computing,* 1983.

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

[5] Coppersmith, D., "Fast Evaluation of Logarithms in Fields of Characteristic Two", to appear,*IEEE Trans. Inform. Theory*; extended abstract in *Proceedings of the Sixteenth Annual Symposium on Theory of Computing,* ACM, pp. 201–207, 1984.

[6] Cover, T.M., "Enumerative Source Encoding", *IEEE Trans. Inform. Theory,* vol IT-19, pp. 73–77, 1973.

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

[8] Goldwasser, S. and S. Micali, "Probabilistic Encryption", *Proceedings of the Fourteenth Annual Symposium on Theory of Computing,* ACM, pp. 365–377, 1982.

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

[10] Kannan, R., "Improved algorithms for integer programming and related lattice problems", *Proceedings of the Fifteenth Annual Symposium on Theory of Computing,* ACM, pp. 193–206, 1983.

[11] Lagarias, J.C. and A.M. Odlyzko, "Solving low-density subset sum problems",*Proceedings of the Twenty-Fourth Annual Symposium on Foundations of Computer Science,* IEEE, pp. 1–10, 1983.

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

[13] Merkle, R.C. and M.E. Hellman, "Hiding information and signatures in trap-door knap-

sacks",*IEEE Trans. Inform. Theory*, vol. IT-24, pp. 525-530, 1978.

[14] Odlyzko, M.O., "Cryptanalytic attacks on the multiplicative knapsack cryptosystem and on Shamir's fast signature scheme", preprint, 1983.

[15] Pohlig, R.C. 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.

[16] Rabin, M.O., "Digitalized signatures and public-key functions as intractable as factorization", Technical report MIT/LCS/TR-212, MIT, 1979.

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

[19] Shamir, A., "A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem",*Proceedings of the Twenty-Third Annual Symposium on Foundations of Computer Science*, IEEE, pp. 145–152, 1982.

[18] Schroeppel, R. 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.

Appendix 1: Discrete logarithms and factorization.

We'll show here how the problem of factoring "paired primes" $n = p \cdot q$ (p, q primes) is polynomially reducible to that of finding indices in Z_n. Let $a \in Z_n^*$. Since $a^{\varphi(n)} = 1 \pmod{n}$, we have

$$a^n = a^{n-\varphi(n)} = a^{pq-(p-1)(q-1)} = a^{p+q-1} \pmod{n}.$$

The index of a^{p+q-1} to base a is a divisor of $p+q-1$, most likely $p+q-1$ itself. Hence a discrete logarithm subroutine will output $p + q - 1$ when given $a^n \pmod{n}$ as input. Having $n = p \cdot q$ and $p + q - 1$, p and q can easily be determined.