Cryptanalytic attacks on the multiplicative knapsack cryptosystem and on Shamir's fast signature scheme

A. M. Odlyzko

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

The basic Merkle-Hellman additive trapdoor knapsack public-key cryptosystem was recently shown to be insecure, and attacks have also been developed on stronger variants of it, such as the Graham-Shamir system and the iterated knapsack cryptosystem. This paper shows that some simple variants of another Merkle-Hellman system, the multiplicative knapsack cryptosystem, are insecure. It is also shown that the Shamir fast signature scheme can be broken quickly. Similar attacks can also be used to break the Schöbi-Massey authentication scheme. These attacks have not been rigorously proved to succeed, but heuristic arguments and empirical evidence indicate that they work on systems of practical size.

Cryptanalytic attacks on the multiplicative knapsack cryptosystem and on Shamir's fast signature scheme

A. M. Odlyzko

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

1. Introduction

One of the best-known public-key cryptosystems, the basic Merkle-Hellman additive trapdoor knapsack system [18], was recently shown to be easy to break by Shamir [25]. Subsequently, Adleman [2] proposed attacks on the Graham-Shamir scheme and on the multiply iterated Merkle-Hellman system. Adleman's attack on the general multiply-iterated knapsack systems does not seem to succeed [3,7], but other attacks on the doubly iterated knapsack schemes have been proposed by Adleman and Lagarias (see [7,14]). Furthermore, Brickell [6] and Lagarias and the author [15] have developed attacks on low-density knapsack cryptosystems. This paper develops attacks on several other public-key cryptosystems. We show that it is easy to break the Shamir fast signature public-key scheme [24], which is related to the Merkle-Hellman additive knapsack system, but is designed for authentication. Another scheme we show how to break, but only under certain circumstances, is the Merkle-Hellman multiplicative knapsack public-key cryptosystem [18]. Both of these schemes are described in detail below. We also indicate briefly how to adapt our methods to attack the Schöbi-Massey knapsack-based signature scheme.

Both the multiplicative and the additive knapsack systems as well as the Shamir signature scheme relied for their presumed security on the difficulty of the general knapsack problem:

Given $n+2$ integers $a_1, \ldots, a_n$, M, and m, find a solution $c_1, \ldots, c_n$ (if it exists) for the modular equation

$$m \equiv \sum_{j=1}^{n} c_j a_j \pmod{M} ,$$

in which each c_j is an integer in the range $0 \le c_j \le 1$ (or else in the range

$0 \le c_j \le \log M$).

This problem is known to be NP-complete [24] and so is presumed to be hard, at least in the worst case. (There is some empirical evidence, backed by heuristics and in some cases by theoretical analyses, which indicates that many instances of this problem may be solvable in polynomial time [6,15].) In order to have a usable cryptosystem, however, it is necessary that the intended user should be able to utilize it, which means that he must be able to rapidly decode messages or sign them (in a signature scheme). In order to make this possible, the designers of the systems mentioned above have built into them *trapdoors*, which enable the intended user to transform a seemingly very hard general knapsack problem into a very easy one. For instance, in the basic Merkle-Hellman additive knapsack system, the designer publishes n keys, $a_1, \ldots, a_n$, which are all large integers of about $2n$ bits each, and anyone wishing to send him a message $(\varepsilon_1, \ldots, \varepsilon_n)$, $\varepsilon_i = 0$ or 1, computes the integer

$$m = \sum_{i=1}^{n} \varepsilon_i a_i$$

and transmits it over a regular communication channel. An eavesdropper faces the apparently intractable problem of determining the ε_i knowing only the a_i and m. The recipient, however, possesses secret keys W and M such that the sequence $Wa_1, \ldots, Wa_n$, when reduced modulo M, becomes a superincreasing sequence [18], and the transformed knapsack problem is thus very easy to solve. In other variations of this scheme, the recipient might have to perform several such modular multiplications using several sets of secret keys. In all cases, however, there is some (secret) trapdoor which allows the recipient to transform the seemingly hard knapsack problem into a very tractable one. It is this presence of trapdoors which makes some of the attacks on the additive knapsack cryptosystems feasible [2,25]. In those attacks the cryptanalyst carries out an initial polynomial-time but relatively long precomputation which with high probability finds some trapdoor (usually not the one built into the system) which then enables him to decrypt each cyphertext about as rapidly as the intended recipient can do it. Other attacks, such as those of [15] and to some extent of [6], do not depend on the existence of specific trapdoors, but on the public knapsack of the cryptosystem being sparse. In those attacks, decrypting each message is still a polynomial-time process for the cryptanalyst, but takes substantially longer than for the possessor of the secret keys (typically on the order of n^4 operations versus n^3 in the case of [15]).

The trapdoor present in the Merkle-Hellman multiplicative knapsack public-key cryptosystem makes that system vulnerable to our attack. This system's design was motivated by the presumed difficulty of both the general knapsack problem and the discrete logarithm problem. Here the designer chooses n relatively prime numbers $p_1, \ldots, p_n$ (which would usually be the first n primes, or else random n-bit primes) and a prime q such that

$$q > \prod_{i=1}^{n} p_i \ , \tag{1.1}$$

together with a primitive root b modulo q. He then finds integers a_i (the discrete logarithms of the p_i to base b), $1 \leq a_i \leq q-1$, such that

$$p_i \equiv b^{a_i} \pmod{q} \ , \tag{1.2}$$

and publishes $a_1, \ldots, a_n$, keeping b and q secret. Anyone wishing to send him a message $(\varepsilon_1, \ldots, \varepsilon_n)$, $\varepsilon_i = 0$ or 1, computes the integer

$$k = \sum_{i=1}^{n} \varepsilon_i a_i \tag{1.3}$$

and transmits it. The intended receipient, knowing b and q, then computes $m \equiv b^k$ (modulo q), $1 \leq m \leq q-1$. Since

$$b^k = \prod_{i=1}^{n} (b^{a_i})^{\varepsilon_i} \equiv \prod_{i=1}^{n} p_i^{\varepsilon_i} \pmod{q} \ ,$$

and q satisfies (1.1), we have

$$m = \prod_{i=1}^{n} p_i^{\varepsilon_i} ,$$

and therefore $\varepsilon_i = 1$ if and only if $p_i \mid m$. Thus the message is easy to recover for the intended recipient. The cryptanalyst, on the other hand, faces the task of either trying to solve for the ε_i knowing only the k in (1.3) and the a_i, or else of trying to discover the trapdoor keys b and q. The first of these attacks (which applies to any knapsack system) is discussed in [15]. Based on actual tests and heuristics, it appears to work in many cases. Its disadvantage is that it requires a separate run of the lattice reduction algorithm (which takes at least on the order of n^4 operations) to attack each n-bit message. Here we discuss an attack that is likely to discover the secret keys b and q in polynomial time with high probability, provided that the p_i (or at least a large subset of them) are known to the cryptanalyst, and are not too large, so that q is $O(n \log n)$ bits long. This attack does not apply if the p_i are not known, or even when they are known but their order is given by a permutation unknown to the cryptanalyst.

A typical value for n might be 100. In that case, if the p_i are the initial n primes, (1.1) forces q to be on the order of 730 bits in length. If the p_i are random 100-bit primes, (1.1) forces q to be on the order of 10,000 bits in length, which might be regarded as prohibitive, both from the standpoint of waste of transmission facilities and because of the difficulty of computing m. Thus the temptation might be to use only small primes p_i, and for ease of implementation perhaps fixed primes all the time, which would make the system vulnerable to our attack.

The multiplicative knapsack system has another weakness which the attack presented here does not exploit. This comes from the fact that the system designer must be able to compute the discrete logarithms a_i in (1.2). The discrete logarithm problem modulo a prime q is widely thought to be very hard, and the best general algorithm that is known for it, due to Western and Miller [28] (see also [19]) and analyzed by Adleman [1], has running time of about

$$\exp(O((\log q)^{1/2} (\log \log q)^{1/2})) .$$

However, relatively efficient algorithms are known when $q-1$ is divisible only by relatively small primes [20]. Hence in the multiplicative knapsack system, q is chosen to have this property. This fact provides additional information to the cryptanalyst, which might make it possible to break the system even when the p_i are not known.

The Shamir fast signature scheme [24] is similar in many ways to the basic Merkle-Hellman additive knapsack system. It was designed to make up for a significant disadvantage of the additive knapsack, namely that it could not be used for authentication. In many applications, what is required is not the ability for *A* to transmit to *B* without anyone else reading the message, but rather a way for *A* to send a message to *B* in such a way that *B* can be sure the message comes from *A* and not from anyone else. The RSA system [22] is very effective in this role, whereas the additive and multiplicative knapsack schemes are not very suitable, because only a very small fraction of the messages that can be transmitted are actually valid cyphertext messages. The Shamir scheme [24] overcomes this difficulty, and has the additional very desirable feature that it is very fast, even when implemented in software.

In the Shamir scheme with parameter *n* ($n = 100$ is suggested in [24]), Party A, who wishes to provide electronic signatures, publishes a prime p of about *n* bits and $2n$ integers $a_1 ,..., a_{2n}$, each of about *n* bits. Then, in order to certify that an *n*-bit message $(\delta_1 ,..., \delta_n)$ comes from *A*, *A* provides a signature consisting of $2n$ integers $\varepsilon_1 ,..., \varepsilon_{2n}$, $0 \le \varepsilon_i \le n$ for all *i*, which he computes using his secret trapdoor information, such that

$$\sum_{i=1}^{n} \delta_i 2^{i-1} \equiv \sum_{j=1}^{2n} \varepsilon_j a_j \pmod{p} . \tag{1.4}$$

Any recipient of *A*'s message can easily check whether (1.4) holds. The presumed security of this scheme stemmed from the fact that solving (1.4) for the ε_j seemed hard without *A*'s secret information. This information consists of a 0-1 matrix $E = (e_{ij})$ of size n by $2n$, such that for $1 \le i \le n$,

$$2^{i-1} \equiv \sum_{j=1}^{2n} e_{ij} a_j \qquad (\mathrm{mod}\ p) \ . \tag{1.5}$$

This matrix is constructed by choosing the e_{ij} independently to be 0 or 1 with equal probability. Then, with very high probability, the leading $n \times n$ submatrix of E will be nonsingular modulo the randomly chosen prime p. (One can even choose p after one chooses the matrix E so that p will not divide the determinant of the leading $n \times n$ submatrix of E.) Next, $a_{n+1}, \ldots, a_{2n}$ are chosen to be random n-bit integers, and the system (1.5), $1 \le i \le n$, is solved to obtain $a_1, \ldots, a_n$.

The simplest way to form the signature $(\varepsilon_1, \ldots, \varepsilon_{2n})$ for an n-bit message $(\delta_1, \ldots, \delta_n)$ is to take

$$\varepsilon_j = \sum_{i=1}^{n} e_{ij} \delta_i \ ,$$

which by (1.5) will satisfy (1.4), and will have $0 \le \varepsilon_j \le n$. This procedure, however, is very insecure, because every message yields $2n$ linear equations for the e_{ij}, so that after about n messages, an eavesdropper can discover the matrix E and thus can break the scheme. To circumvent this problem, Shamir proposed that to sign $(\delta_1, \ldots, \delta_n)$, A should first choose a random subset $a_{i_1}, \ldots, a_{i_k}$ of the a_i, form the signature $(\varepsilon'_1, \ldots, \varepsilon'_{2n})$ for the message

$$\sum_{i=1}^{n} \delta_i 2^{i-1} - \sum_{j=1}^{k} a_{i_j}$$

as above, and then form the signature $(\varepsilon_1, \ldots, \varepsilon_{2n})$ by taking $\varepsilon_{i_j} = \varepsilon'_{i_j} + 1$, $1 \le j \le k$, and $\varepsilon_i = \varepsilon'_i$ if $i \ne i_j$ for any j. This appears to sufficiently randomize the linear equations so as to make a linear algebra attack on this scheme unlikely to succeed. However, Section 4 shows that the difficulties can be overcome by using recently developed tools of diophantine approximation. We do not reconstruct the secret matrix E. Instead, using on the order of $O(n^4)$ bit operations we construct a different matrix which then allows us to sign a message using only $O(n^3)$ operations. As a comparison, we note that the secret matrix E takes on the order of n^5 bit operations to compute (using Gaussian elimination and ordinary arithmetic schemes, which are the only practical ones in the ranges of interest) even if we don't count the cost of computing random numbers, and each signature takes on the order of n^2 bit operations to generate once E is known.

The attack we describe depends on the matrix E being essentially a random $0-1$ matrix so that a signature $(\varepsilon_1, \ldots, \varepsilon_{2n})$ is recognized as valid whenever it satisfies the congruence (1.4) and has $0 \le \varepsilon_i \le n$, $1 \le i \le 2n$. However, it is possible to restrict attention to matrices E in which all column sums are $\le m$ for some $m < n$, so that valid signatures have $0 \le \varepsilon_i \le m$. (Shamir suggested in his original proposed [24] that for $n = 100$, it might be best for technical reasons to take $m = 63$, for example.) If m is taken to be very small, say $m < \sqrt{n}$, then the attack we describe here is expected to fail. However, such a system would still be vulnerable to an attack similar to the one described here, but directed at each message individually. In addition, each signature would then reveal a lot of information about the matrix E, which might be sufficient for the cryptanalyst to recover E. We do not pursue these approaches here, since our purpose is just to point out some of the basic weaknesses of these systems, which should be sufficient to discourage people from using them.

Recently Shamir and Tulpan [26] have developed another attack on the Shamir signature scheme. Unlike the attack prescribed here, it can be proved to succeed with high probability, but its running time is exponential in n. For n not too large, though, the running time is not very high. The case of $n = 100$ was tested successfully in [26], and the authors extrapolated that even $n = 500$ might be doable.

Schöbi and Massey [23] have proposed a knapsack-based signature scheme different from the Shamir one. Its advantage is that it can be used both for transmission of information and authentication. Its disadvantage is that the secret key is very large, and it is much slower than Shamir's. In Section 5 we briefly indicate how it can be attacked by methods similar to those used for breaking the multiplicative knapsack system and the Shamir scheme.

As is the case with the other attacks [2,6,15,25] on the additive knapsack schemes, we cannot prove that our attacks always succeed. In fact, while attacks on the basic Merkle-Hellman additive knapsack cryptosystem can be shown to succeed most of the time [14,25], we cannot prove even that for our attacks. The main reason is that we rely on the Lenstra, Lenstra, and Lovász lattice basis reduction algorithm [17] satisfying certain conditions (at least on average) which it has not been proved to possess. However, extensive numerical experiments by Brickell [8] and the author show that the algorithm often does satisfy those conditions (which are discussed in Section 2). Extrapolation from these numerical experiments indicate that our attack on the Shamir scheme, for example, ought to be successful for n up to at least 1000.

In Section 2, we discuss briefly the lattice basis reduction algorithm which is the basic tool of our attacks, as well as other algorithms that could possibly be used in its place. Sections 3 and 4 present the cryptanalysis of the multiplicative knapsack system and the Shamir fast signature scheme, respectively. Section 4 also describes successful attempts to break the Shamir scheme using the approach presented there. Finally, in Section 5 we briefly discuss attacks on the Schöbi-Massey scheme.

2. Lattice basis reduction algorithm

The initial attack on the additive knapsack cryptosystems by Shamir relied on H. W. Lenstra's polynomial time algorithm for the integer programming problem when the number of variables is fixed [16]. Since then, this algorithm has been largely superseded in cryptographic applications by the Lenstra, Lenstra, and Lovász lattice reduction algorithm [17], (which we will refer to as the L^3 algoritm), which is easy to program and whose running time is polynomial in the total size of the problem. It was first applied in cryptography by Adleman [2].

Given n vectors $\underline{v}_1, \ldots, \underline{v}_n \in Z^n$, the lattice L spanned by them is the collection of all integer linear combinations

$$\sum_{i=1}^{n} c_i \underline{v}_i \ , \qquad c_i \in Z \ .$$

Many problems in computational complexity, number theory, and cryptography can be reduced to the problem of finding the shortest vector in a lattice. In many cases, it suffices just to find some short vector in a lattice. The L^3 algorithm [17] does find a fairly short vector. To be precise, given a basis $\underline{v}_1, \ldots, \underline{v}_n$ for a lattice L, the algorithm produces another basis $\underline{w}_1, \ldots, \underline{w}_n$ for L that Lenstra, Lenstra, and Lovász call *reduced.* We do not need the precise definition, but it can be shown [17] that the shortest vector in the reduced basis is relatively short:

$$\min_i \ \| \underline{w}_i \|^2 \leq 2^{n-1} \min_{\underline{v} \in L'} \ \| \underline{v} \|^2 \ , \tag{2.1}$$

where $L' = L \backslash \{ (0, \ldots, 0) \}$, and

$$\| (u_1, \ldots, u_n) \|^2 = \left[\sum_{i=1}^{n} u_i^2 \right]^{1/2}$$

is the euclidean norm of a vector. Furthermore, the other vectors of the reduced lattice basis tend to be short also. The running time of the original algorithm was shown to be $O(n^6 (\log\ B)^3)$, where B is an upper bound for the coordinates of the initial basis $\underline{v}_1, \ldots, \underline{v}_n$, and slightly faster variants have been proposed [12].

What is perhaps most striking about the L^3 algorithm is that it performs much better in practice than it is guaranteed to. Very extensive tests by Brickell [8] and independently by the author (up to dimension 102) show that in most cases the reduced basis vectors are very short, in fact much shorter than the worst case bound (2.1) guarantees. Some of the computational results are described in greater detail in [15]. The assumption underlying our attacks is that when the L^3

algorithm is applied to the lattices that arise in our problems (which are of a somewhat special nature), it will in a substantial fraction of the cases find reduced bases in which the shortest vector is at most a constant factor longer than the shortest vector in the lattice, and that the next few shortest vectors in the reduced basis are also short. This assumption is fully supported by the experiments that have been carried out so far, in dimensions up to around 100. This is sufficient for our attack on the Shamir scheme, for example, with parameter $n \lesssim 500$.

There are some indications that in higher dimensions the performance of the L^3 algorithm might deteriorate. For example, the numerical evidence seems to imply that the ratio of the product of the lengths of the vectors in a reduced basis produced by the L^3 algorithm to the determinant of the lattice grows roughly like $\exp\ (cn^2)$ for some small constant c. Except for the value of c, this is similar to the worst case bound of [17]. On the other hand, by reasoning in analogy with the reduction theory of quadratic forms [9; pp. 259-263] we might expect that for most lattices there would be some reduced basis for which the above ratio grows only as $\exp\ (c' \ n \ \log \ n)$. Thus it is quite possible that our attacks might fail in very high dimension, say on the order of several hundred. Such high dimensions, though, would be required in the case of the Shamir signature scheme (for example) only when the parameter size n in that scheme (which is about $2 \ \log_2 \ n$ times larger than the dimension of the lattice that is used in the cryptanalysis) is on the order of several thousand, which makes that scheme impractical. Furthermore, as new diophantine approximation algorithms are developed, the range of applicability of our attacks can be expected to increase.

It is possible to significantly speed up the L^3 algorithm by performing some of the crucial multiprecision operations in floating point arithmetic. The rigorous analysis of [17] then no longer applies, but since all operations on basis elements are still carried out in integers, at the end of the computation it is easy to check that the basis is in fact reduced. The running time of this modified algorithm, assuming that roundoff errors do not cause it to make incorrect decisions, can be shown to be $O(n^4 (\log B)^3)$. In practice, the running time seems to be much shorter, perhaps on the order of $n (\log B)^3$ or so. Present computational experience with lattices in dimensions up to 102 indicates that even a relatively inefficient high-level language implementation of the modified algorithm on a high-speed machine such as the CRAY-1 can find a reduced basis for a typical 50-dimensional lattice given in terms of 100-bit coordinates in a few minutes.

We use the L^3 algorithm primarily because its basic version is the only one in this area which is guaranteed to produce reasonably short vectors in a lattice in polynomial time. It is quite possible that other algorithms, such as those of [4,5,10,11,21,27], or modifications of them, could perform just as well or better in practice on typical lattices. Therefore the running time bounds that we quote should be regarded as provisional, and capable of substantial improvement.

3. Multiplicative knapsack

In this section we present an attack on the multiplicative knapsack cryptosystem. We assume that integers $a_1 ,..., a_n$ are known and at least some of the primes (or relatively prime numbers) $p_1 ,..., p_n$ are known, and we try to find the unknown integer b and prime q such that

$$b^{a_i} \equiv p_i \pmod{q} . \tag{3.1}$$

We assume that b, q, and the a_i are of about m bits; if the p_i are the first n primes, then $m \sim n \log_2 n$, and if the p_i are n-bit primes, then $m \gtrsim n^2$, since we must have $q \gtrsim \prod p_i$. Our attack will take polynomial time only if $m = O(n \log n)$.

The basic idea behind the attack is to find a collection of vectors $\underline{\delta} = (\delta_1 ,..., \delta_n)$ such that

$$0 = \sum_{i=1}^{n} \delta_i a_i , \tag{3.2}$$

and such that the δ_i are not too large. (If not all the p_i are known, then we also require that

$\delta_i = 0$ for those i for which p_i is not known.) By (3.1), each such vector $\underline{\delta}$ gives a congruence of the form

$$1 \equiv \frac{u(\underline{\delta})}{v(\underline{\delta})} \quad (\mathrm{mod}\ q)\ , \tag{3.3}$$

where

$$u(\underline{\delta}) = \prod_{\substack{i \\ \delta_i > 0}} p_i^{\delta_i}\ , \qquad v(\underline{\delta}) = \prod_{\substack{i \\ \delta_i < 0}} p_i^{-\delta_i}\ . \tag{3.4}$$

But then

$$q \mid u(\underline{\delta}) - v(\underline{\delta})\ .$$

If we compute several of the $u(\underline{\delta}) - v(\underline{\delta})$, then their greatest common divisor is likely to be q. Once we have found q, it is easy us to compute b. We find a solution $\underline{\delta}$ to

$$1 = \sum_{i=1}^{n} \delta_i a_i \tag{3.5}$$

(with $\delta_i = 0$ for those i for which p_i is not known, again) and then compute

$$b \equiv \Pi p_i^{\delta_i} \quad (\mathrm{mod}\ q)\ .$$

Since q is already known, we do not require the δ_i in (3.5) to be small in order to compute b efficiently. In some cases it may be impossible to find a solution to (3.5), but that can happen only if $gcd(a_1 , \ldots , a_n) = d > 1$, in which case we might as well work with b^d and $a_1 / d , \ldots , a_n / d$. In most cases, however, even the first few of the a_i should be relatively prime, so there will be no difficulty.

We now consider this approach in detail. Vectors $\underline{\delta}$ that satisfy (3.2) are easy to find in general. The difficulty is in finding $\underline{\delta}$ that satisfy (3.2) with small δ_i. There are several ways to use the L^3 algorithm to find solutions to (3.2) with small δ_i. We pick a subset of k a_i's for which the p_i are known. By renumbering the a_i and p_i, we can assume that we are dealing with

$a_1,\dots,a_k$ and $p_1,\dots,p_k$. We next apply the L^3 algorithm to the lattice L spanned by the $k+1$ vectors in Z^{k+1} given by

$$\begin{aligned}
\underline{w}_1 &= (1,0,0,\dots,0,2^m a_1) \ , \\
\underline{w}_2 &= (0,1,0,\dots,0,2^m a_2) \ , \\
&\ \vdots \\
\underline{w}_k &= (0,0,\dots,0,1,2^m a_k) \ , \\
\underline{w}_{k+1} &= (0,0,\dots,0,0,2^{4m}) \ .
\end{aligned} \tag{3.6}$$

(In actual implementations one would replace 2^m by a smaller power of 2, but we will not worry about such details here.) Note that any vector in L is given by

$$\left[\delta_1,\delta_2,\dots,\delta_k,\ 2^m \left\{ \sum_{i=1}^{k} \delta_i a_i \right\} + d\ 2^{4m} \right] \tag{3.7}$$

for some $\delta_1,\dots,\delta_k$, $d \in Z$. For a vector to be short, its last coordinate has to be zero, and the δ_i should all be small. This can only happen if $d = 0$ and $\sum \delta_i a_i = 0$, since if $d \neq 0$, some of the δ_i would have to be very large, which would make the vector long.

To determine how short the vectors in the reduced basis might be, we apply the pigeon-hole principle. Consider the integers

$$\sum_{i=1}^{k} x_i a_i \ , \qquad 0 \le x_i \le B \ .$$

They are all $\lesssim kB2^m$ in absolute magnitude, and so if their total number, $(B+1)^k$, exceeds $kB2^m$ (which occurs for $B \approx 2^{m/(k-1)}$), two of them will have to be equal, and so we obtain an equation of the form

$$\sum_{i=1}^{k} x_i a_i = \sum_{i=1}^{k} x_i' a_i \ , \qquad 0 \le x_i,\ x_i' \le B \ ,$$

which gives, for $y_i = x_i - x_i'$,

$$\sum_{i=1}^{k} y_i a_i = 0 , \qquad |y_i| \le B .$$

In fact, it is easy to show that many such equations have to be satisfied. Therefore we can expect that there are many vectors in our lattice of length $\lesssim \sqrt{k}\, 2^{m/(k-1)}$. The basic unproved assumption underlying our attack is that the L^3 algorithm will find such vectors.

We next consider the running time of the algorithm. The L^3 algorithm takes about $O(km^3)$ bit operations (somewhat more if we use the original rigorously analyzed form of it). The integers $u(\underline{\delta})$ and $v(\underline{\delta})$ will each have on the order of $km\, 2^{m/(k-1)}$ bits, and so will the $u(\underline{\delta}) - v(\underline{\delta})$. Hence computing the $u(\underline{\delta}) - v(\underline{\delta})$ and their greatest common divisors, which will give q, takes on the order of $k^2 m^2\, 4^{m/(k-1)}$ bit operations. Finally, computing b takes $O(m^3)$ operations. Hence our estimate for the total work involved is on the order of

$$km^3 + k^2 m^2 2^{2m/(k-1)}$$

bit operations. If $m \sim n \log_2 n$ (when the p_i are the initial n primes, for example), and if we choose $k = n$, this results in about n^6 bit operations. However, if the p_i are n-bit primes, $m \sim n^2$, and the algorithm is exponential.

4. Shamir’s fast signature scheme

The cryptanalysis of the Shamir fast signature scheme is in some ways similar to the attack on the multiplicative knapsack system. Suppose that the prime p of about n bits and the n-bit numbers $a_1, \ldots, a_{2n}$ are the public keys. To be able to sign a message, we only need to be able to find, for each integer m in the range $0 \le m \le p-1$, some integers $\varepsilon_1, \ldots, \varepsilon_{2n}$ such that

$$m \equiv \sum_{i=1}^{2n} \varepsilon_i a_i \pmod{p} , \tag{4.1}$$

and such that $0 \le \varepsilon_i \le n$. We will find a way to do this fast without discovering the secret matrix $E = (e_{ij})$.

We first find solutions to

$$\sum_{i=1}^{2n} \varepsilon_i a_i \equiv 0 \quad (\mathrm{mod}\ p) \ , \tag{4.2}$$

where the ε_i are to be small but not identically zero. Initially, we work with only a subset of the a_i, which without loss of generality we may take to be $a_1, \ldots, a_k$.

By the pigeon-hole principle, a solution to

$$\sum_{i=1}^{k} \varepsilon_i a_i = 0 \tag{4.3}$$

with $|\varepsilon_i| \leq B$ is guaranteed to exist approximately when $B \gtrsim 2^{n/(k-1)}$. Each such solution will of course give a solution to

$$\sum_{i=1}^{k} \varepsilon_i a_i \equiv 0 \quad (\mathrm{mod}\ p) \ . \tag{4.4}$$

To find solutions to (4.4), we consider the lattice L spanned by

$$\begin{aligned} \underline{w}_1 &= (2^r a_1, 1, 0, \ldots, 0) \ , \\ \underline{w}_2 &= (2^r a_2, 0, 1, \ldots, 0) \ , \\ &\vdots \\ \underline{w}_k &= (2^r a_k, 0, 0, \ldots, 1) \ , \\ \underline{w}_{k+1} &= (2^r p, 0, 0, \ldots, 0) \ , \end{aligned} \tag{4.5}$$

where $r \approx 4n/k$, say. A vector in L is of the form

$$\underline{w} = (2^r \left[\sum_{i=1}^{k} \varepsilon_i a_i - dp \right], \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k) \ ,$$

and so is short only when the ε_i are small and

$$\sum_{i=1}^{k} \varepsilon_i a_i = dp \ ,$$

that is, when we have a solution to (4.4). Our assumption, supported by empirical evidence, is that when the LLL algorithm is applied to a basis of the form (4.5), it will find a reduced basis $\underline{w}_1^*, \ldots, \underline{w}_{k+1}^*$ such that the first coordinate of each of $\underline{w}_1^*, \ldots, \underline{w}_k^*$ is equal to 0, and the other coordinates are $\lesssim 2^{n/(k-1)}$ in absolute magnitude.

Once the reduced basis $\underline{w}_1^*, \ldots, \underline{w}_{k+1}^*$ has been found with the above properties, vectors $\underline{v}_i$, $1 \le i \le k$, are formed by deleting the first coordinate of $\underline{w}_i^*$. The $\underline{v}_i$ generate a k-dimensional lattice of solutions to (4.4). We then form the matrix whose rows are the $\underline{v}_i$, and compute its inverse. This completes the basic stage of our attack.

Once the steps outlined above have been completed, we can easily generate solutions to (4.1), which give the desired forged signatures. Suppose that m is given. We choose $d_1, \ldots, d_{2n}$ to be integers in the range $0 \le d_i \le n$, subject to the constraint that $d_1, \ldots, d_k$ should be close to $n/2$. We next select $x \in Z$ such that

$$x a_1 \equiv m - \sum_{i=1}^{2n} d_i a_i \pmod{p} \ . \tag{4.6}$$

Using the precomputed inverse of the matrix of the $\underline{v}_i$, we express

$$(x, 0, \ldots, 0) = \sum_{i=1}^{k} c_i \underline{v}_i \ , \qquad c_i \in R \ .$$

(Since we only need to know the c_i to within $\pm 1/10$, say, the inverse of the matrix can be computed in floating point arithmetic.) We then form

$$\underline{y} = (y_1, \ldots, y_k) = (x, 0, \ldots, 0) - \sum_{i=1}^{k} ((c_i)) \underline{v}_i = \sum_{i=1}^{k} \{ c_i - ((c_i)) \} \underline{v}_i \ ,$$

where $((t))$ denotes the integer nearest to t. We expect the $c_i - ((c_i))$ to behave like independent random variables distributed uniformly on $[-1/2, 1/2]$, and so we expect the coordinates of $\underline{y}$ on average to be about $\lesssim B(k/12)^{1/2}$ in absolute magnitude. Hence we can expect that the $y_i + d_i$, $1 \le i \le k$, will all be in the range $[0, n]$ if $Bk^{1/2}$ is sufficiently much smaller than n, and we will have

$$\sum_{i=1}^{k} (y_i + d_i) a_i + \sum_{i=k+1}^{2n} d_i a_i \equiv m \quad (\mathrm{mod}\ p) ,$$

which gives the desired signature.

The computational complexity of this attack and the allowable choices for k are easy to evaluate. The LLL algorithm takes on the order of $k^4 n^3$ bit operations. We are constrained by the fact that we need to have $B\sqrt{k} \lesssim n$ (actually, $\lesssim n(\log n)^{-1/2}$), where $B \approx 2^{n/(k-1)}$. If we choose k slightly larger than $2n/(\log_2 n)$, the constraints will be satisfied, and the running time of the initial preprocessing stage will be on the order of n^4. Once the initial stage is completed, any signature takes about n^3 bit operations to generate.

The above attack might occasionally fail because some coefficient $y_i + d_i$ is either < 0 or $> n$. In that case, we can either try to correct this problem by adding some combination of the $\underline{v}_i$, or else can try again with a different set of d_i's. In any case, this problem is unlikely to arise, at least in the basic Shamir signature scheme described here. If the matrix E is chosen so that its column sums are all $\leq m < n$, so that valid signatures have $0 \leq \varepsilon_i \leq m$, it might be necessary to choose k larger than $2n/(\log_2 n)$, since we need $Bk^{1/2} m^{-1}$ to be quite small.

The attack presented above on the basic Shamir scheme was implemented using the Vaxima symbolic manipulation system. It was tested on about half a dozen sets of keys that were generated using $n = 30$ and $p = 2^{31} - 1$. The basic stage was carried out using $k = 12$ and $r = 10$. In about 200 examples, the absolute values of coordinates of the $\underline{y}$ vector were always ≤ 7, and were ≤ 5 most of the time. Several sets of keys with $n = 50$ and $p = 669369903482281$ were also tested using $k = 16$ and $r = 14$. In over 50 examples, the absolute values of the coordinates of the $\underline{y}$ vector were usually ≤ 7, although once a coefficient of 14 appeared.

5. The Schobi-Massey scheme

This scheme [23] is a modification of the iterated Merkle-Hellman additive knapsack cryptosystem. The designer publishes keys $a_1 ,..., a_n$, derived from the usual iterated system, which can be used to transmit n-bit messages to him (or her). To sign a message m (which we can again regard as an integer in the range $0 \le m \le \sum_1^n a_i$), the designer provides a set of integers x_i such that

$$m = \sum_{i=1}^{n} x_i a_i \ , \tag{5.1}$$

and the x_i are not too large in absolute value. The construction of the x_i is quite complicated, and we will not present it here. In their paper, Schöbi and Massey [23] suggest the use of a triply iterated knapsack with $n = 200$, in which case the x_i might satisfy $|x_i| \le 100$, say.

Our attack on the Schöbi-Massey signature scheme is very similar to those on the multiplicative knapsack and on Shamir's scheme, so we only briefly sketch it. Given a message m, to forge a signature we select $x_{k+1} ,..., x_n$ at random in the allowed range, and attempt to compute $x_1 ,..., x_k$ so that the x_i are small and (5.1) is satisfied. (The size of k would depend on n, the sizes of the a_i, and the desired sizes of the x_i.) We form the lattice (cf. [15])

$$\begin{aligned}
\underline{w}_1 &= (1,\ 0,...,\ 0,\ a_1) \ , \\
\underline{w}_2 &= (0,\ 1,...,\ 0,\ a_2) \ , \\
&\vdots \\
\underline{w}_k &= (0,\ 0,...,\ 1,\ a_k) \ , \\
\underline{w}_{k+1} &= (0,\ 0,...,\ 0,\ b) \ ,
\end{aligned}$$

where

$$b = m - \sum_{i=k+1}^{n} x_i a_i \ .$$

If $x_{k+1}, \ldots, x_n$ were chosen so that b is large (as will happen for most random choices), then vectors

$$x_1 \underline{w}_1 + x_2 \underline{w}_2 + \ldots + x_k \underline{w}_k - \underline{w}_{k+1}$$

corresponding to solutions of (5.1) with small $x_1, \ldots, x_k$ will be short, and we expect that the L^3 algorithm will find them.

Acknowledgement

The author thanks A. E. Brouwer, J. C. Lagarias, and H. W. Lenstra, Jr., for their comments.

References

1. L. M. Adleman, A subexponential algorithm for the discrete logarithm problem with applications to cryptography, pp. 55-60 in Proc. 20-th IEEE Symp. Found. Computer Sci. (1979).

2. L. M. Adleman, On breaking the iterated Merkle-Hellman public key cryptosystem, pp. 303-308 in *Advances in Cryptology: Proceedings of Crypto 82,* D. Chaum, R. L. Rivest, and A. T. Sherman, eds., Plenum Press, 1983.

3. L. M. Adleman, E. F. Brickell, J. C. Lagarias, and A. M. Odlyzko, paper in preparation.

4. L. Afflerbach, Minkowskische Reduktionsbedingungen für positiv definite quadratische Formen in 5 Variabeln, Monatsh. Math. *94* (1982), 1-8.

5. A. J. Brentjes, *Multi-dimensional Continued Fraction Algorithms,* Ph.D. dissertation, Leiden, 1981.

6. E. F. Brickell, Solving low density knapsacks, to appear in Proceedings of Crypto 83, to be published by Plenum Press.

7. E. F. Brickell, J. C. Lagarias, and A. M. Odlyzko, Evaluation of the Adleman attack on multiply iterated knapsack cryptosystems, extended abstract to appear in Proceedings of Crypto 83, to be published by Plenum Press.

8. E. F. Brickell and G. J. Simmons, A status report on knapsack based public key cryptosystems, Congressus Numerantum, *37* (1983), 3-72.

9. J. W. S. Cassels, *Rational Quadratic Forms,* Academic Press, 1978.

10. U. Dieter, How to calculate shortest vectors in a lattice, Math. Comp. *29* (1975), 827-833.

11. H. R. P. Ferguson and R. W. Forcade, Multidimensional Euclidean algorithms, J. reine

angew. Math. *344* (1982), 171-181.

12. E. Kaltofen, On the complexity of finding short vectors in integer lattices, pp. 236-244 in *Computer Algebra*, J. A. van Hulzen, ed., Proceedings of EUROCAL ’83, Lecture Notes in Computer Science #162, Springer-Verlag, 1983.

13. J. C. Lagarias, The computational complexity of simultaneous diophantine approximation problems, pp. 32-39 in Proc. 23-rd IEEE Symp. Found. Comp. Sci. (1982). Revised version to appear in SIAM J. Comp.

14. J. C. Lagarias, Knapsack-type public key cryptosystems and diophantine approximation, extended abstract to appear in Proceedings of Crypto 83, to be published by Plenum Press. (Full manuscript in preparation.)

15. J. C. Lagarias and A. M. Odlyzko, Solving low-density subset sum problems, pp. 1-10 in Proc. 24-th IEEE Symposium on Foundations of Computer Science (1983). Revised version to be published.

16. H. W. Lenstra, Jr., Integer programming with a fixed number of variables, Math. of Operations Research, to appear.

17. A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, Factoring polynomials with rational coefficients, Math. Annalen, *261* (1982), 515-534.

18. R. C. Merkle and M. E. Hellman, Hiding information and signatures in trapdoor knapsacks, IEEE Trans. Inform. Theory, IT-24 (1978), 525-530.

19. J. C. P. Miller, On factorization, with a suggested new approach, Math. Comp., *29* (1975), 155-172.

20. S. C. Pohlig and M. E. Hellman, An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance, IEEE Trans. Inform. Theory, IT-24 (1978), 106-

110.

21. M. Pohst, On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications, ACM SIGSAM Bulletin *15* (1981), 37-44.

22. R. L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Comm. ACM, *21* (1978), 120-126.

23. P. Schöbi and J. L. Massey, Fast authentication in a trapdoor-knapsack public key cryptosystem, pp. 289-306 in *Cryptography,* T. Beth, ed., Lecture Notes in Computer Science #149, Springer-Verlag, 1983.

24. A. Shamir, A fast signature scheme, MIT Laboratory for Computer Science report, 1978.

25. A. Shamir, A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem, pp. 145-152 in Proc. 23-rd IEEE Symp. Found. Computer Sci. (1982).

26. A. Shamir and Y. Tulpan, Fast cryptanalysis of a fast signature scheme, in preparation.

27. G. Szekeres, Multidimensional continued fractions, Ann. Univ. Sci. Budapest, Sect. Math., *13* (1970), 113-140.

28. A. E. Western and J. C. P. Miller, *Tables of Indices and Primitive Roots,* Royal Soc. Math. Tables, Vol. 9, Cambridge Univ. Press 1968.