A Polynomial-Time Algorithm for Breaking the Basic Merkle–Hellman Cryptosystem

ADI SHAMIR

Abstract—The Merkle–Hellman cryptosystem is one of the two major public-key cryptosystems proposed so far. It is shown that the basic variant of this cryptosystem, in which the elements of the public key are modular multiples of a superincreasing sequence, is breakable in polynomial time.

I. Introduction

IN 1976 Diffie and Hellman published their pioneering paper on public-key cryptography [2]. Their paper speculated that such cryptosystems exist and surveyed their potential applications but did not describe actual implementations. In late 1976 and early 1977, the first two public-key cryptosystems were discovered (see [5], [6]). Since then many variants and a few new public-key cryptosystems have been proposed, but for a variety of reasons these first two systems continue to dominate the field. They have been extensively analyzed, and a number of cryptanalytic attacks have been proposed to try to break them. However, all these attacks are unlikely to succeed unless the cryptosystems are greatly simplified or their key sizes reduced.

We describe the first cryptanalytic attack we know of that can break a full-size variant of one of these cryptosystems in reasonable time and space complexities. The variant is known as the single-iteration Merkle–Hellman cryptosystem, and it is the simplest (and presumably the least secure) in the family of public-key cryptosystems proposed in Merkle and Hellman's original paper. The cryptanalytic attack is not directly applicable to multi-iteration Merkle–Hellman cryptosystems, and thus the cryptographic security of these variants remains an open problem.

The algorithm is easy to implement, and it is efficient even on a microcomputer. It always halts after polynomially many steps, but it can sometimes fail to break a particular key. Heuristic arguments indicate that such failures are exceedingly rare, and they are supported by hundreds of tests conducted on full-size keys without a single failure.

A number of countermeasures can protect one's knapsack-based cryptosystem against the specific attack considered here. For some of these countermeasures, there are counter-countermeasures that can revitalize the attack. Cryptography is a never-ending struggle between code makers and code breakers, and this paper is not claiming to give any ultimate answers in this sense.

Section II presents an overview of the Merkle–Hellman cryptosystem. In Section III we describe the cryptanalytic attack in an informal way, and in Section IV we analyze its performance. A discussion of the results appears in Section V.

II. Basic Merkle–Hellman Cryptosystem

The public encryption key in any Merkle–Hellman cryptosystem is a sequence of n natural numbers $a_1, \cdots, a_n$. (A typical value of n is 100 and a typical size of each a_i is 200 bits.) To encrypt an n-bit cleartext, $X = x_1, \cdots, x_n (x_i \in \{0,1\})$, the sender computes a message-dependent partial sum of the a_i elements:

$$b = \sum_{i=1}^{n} x_i a_i,$$

and sends the ciphertext b to the receiver via the (insecure) communication channel. Both the receiver and the potential eavesdropper know $a_1, \cdots, a_n$ and b, and they have to find which subset of the a_i elements sums up to b. This is an instance of the knapsack problem, which is known to be nondeterministic polynomial time complete (NP-complete). To make this problem apparently difficult (for the eavesdropper) but actually easy (for the receiver), the sequence $a_1, \cdots, a_n$ is chosen in a special way. First, the receiver chooses a sequence of numbers $a'_1, \cdots, a'_n$ for which the associated knapsack instances are easy to solve. Then he scrambles the numbers in such a way that only he knows how to change them back to their easy original form. Finally, he publishes the scrambled numbers $a_1, \cdots, a_n$ as his public encryption key.

There are many ways in which the easy sequence can be chosen and then disguised. The basic scheme proposed in Merkle and Hellman's paper is based on superincreasing sequences and modular multiplications. A sequence of numbers $a'_1, \cdots, a'_n$ is *superincreasing* if each number in it is larger than the sum of its predecessors:

$$a'_i > \sum_{j=1}^{i-1} a'_j.$$

For any superincreasing sequence, there is a linear-time greedy algorithm for solving all its associated knapsack

Manuscript received October 26, 1983; revised January 16, 1984. This paper was presented at Crypto '82, The University of Santa Barbara, CA., and at the 23rd Foundations of Computer Science Conference.

The author is with the Department of Applied Mathematics, Weizmann Institute of Science, Rehovot, 76100, Israel.

instances. To hide the obvious structure of such a sequence, the receiver randomly chooses two numbers, M_0 (the *modulus*) and U_0 (the *multiplier*), such that M_0 is larger than the sum of all the a_i' and U_0 is relatively prime to M_0. Each a_i' is then transformed into a new, randomly looking number between 0 and $M_0 - 1$ by the modular multiplication

$$a_i \equiv U_0 \cdot a_i' \pmod{M_0},$$

and the new sequence, $a_1, \cdots, a_n$, is published as the encryption key.

To show that the asymptotic complexity of our cryptanalytic attack is polynomial, we have to consider a family of cryptosystems whose sizes grow to infinity. There are two basic parameters we have to consider: the number of elements in the published key and their sizes. If either one of these is kept constant, there is a trivial polynomial-time algorithm for solving the associated knapsack instances. We thus make the assumption that the size of the modulus M_0 (and therefore also the size of the a_i elements) grows linearly with n. If d is the proportionality constant ($1 < d < \infty$), we choose a_1' to be a $dn - n$ bit number, a_i' to be a $dn - n + i - 1$ bit number, and M_0 to be a dn bit number (dn is rounded to the nearest integer whenever necessary). Merkle and Hellman use this scheme with $d = 2$ and $n = 100$, so that the a_i' grow in size from 100 to 199 bits and $|M_0|$ is 200. The parameter d measures the redundancy introduced by the cryptosystem (i.e., the ratio between the sizes of the ciphertext and the cleartext). The complexity of our algorithm is a rapidly growing function of d, but for each fixed d it is polynomial in n.

III. Informal Description of the Algorithm

The algorithm proposed in this paper analyzes the given numbers $a_1, \cdots, a_n$ and attempts to find a *trapdoor pair* of natural numbers M and W such that $W \cdot a_i \pmod{M}$ is a superincreasing sequence and its sum is smaller than M. If any pair of numbers with these properties are known, then one can solve all the knapsack instances associated with $a_1, \cdots, a_n$ in linear time. Since the a_i were obtained from a superincreasing sequence by modular multiplication, we know that at least one such pair exists (with $W_0 \equiv U_0^{-1} \pmod{M_0}$). Our algorithm finds some trapdoor pair, but it is not guaranteed to find the original modulus and multiplier used in the construction of the public key.

The algorithm is divided into two parts. In the first part Lenstra's integer programming algorithm [4] is used to find a few small intervals in $[0,1]$ such that a *necessary* condition for M and W to be a trapdoor pair is that the ratio W/M is in such an interval. In the second part of the algorithm we use the fact that W/M is approximately known to carry out a finer analysis and divide each interval into smaller subintervals such that a *sufficient* condition for M and W to be a trapdoor pair is that their ratio is in such a subinterval. At least one of the subintervals must be nonempty, and by using a fast diophantine approximation algorithm [1], we can find the smallest M and W whose ratio satisfies this condition.

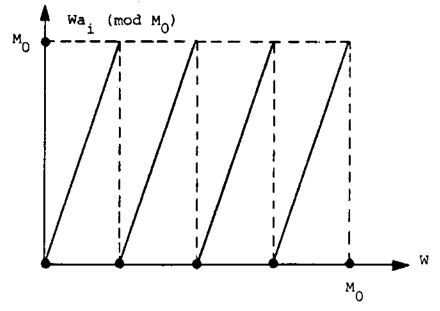

Fig. 1.

Let M_0 be the (unknown) dn bit modulus used in the construction of the encryption key. We now generalize the definition of a trapdoor pair by considering arbitrary real positive values of W. The graph of the function $Wa_i \pmod{M_0}$ for real multipliers $0 \le W < M_0$ has a sawtooth form, as shown in Fig. 1. The slope of the function (except at discontinuity points) is a_i, the number of minima is a_i, and the distance between successive minima is M_0/a_i (which is slightly more than 1).

Let us consider now the sawtooth curve associated with a_1. The multiplier W_0 has the property that $a_1' \equiv W_0 \cdot a_1 \pmod{M_0}$ is at most 2^{dn-n}. Since the slope of the curve is a_1, the horizontal distance between W_0 and the closest minimum of the a_1 curve to its left cannot exceed $2^{dn-n}/a_1 \approx 2^{-n}$. The unknown W_0 must thus be extremely close to some minimum of the a_1 sawtooth curve. Unfortunately, even if we impose the integrality constraint on W (which we do not), there are too many possible values for W_0, and we cannot check them one by one.

A similar analysis shows that W_0 must also be within a distance of $2^{dn-n+1}/a_2 \approx 2^{-n+1}$ from the closest a_2 curve minimum to its left. Consequently, the two minima of the a_1 and a_2 curves must be very close to each other (the a_2 minimum can be up to 2^{-n+1} to the left or up to 2^{-n} to the right of the a_1 minimum, depending on the exact location of W_0). This closeness condition greatly reduces the number of places in which W_0 may be, but in most cases it still does not characterize it uniquely.

Similarly, we can superimpose more sawtooth curves on the same diagram. The fact that W_0 is close to a minimum on each curve implies that all these minima are close to each other, and thus instead of finding W_0, we must find the accumulation points of minima of the various curves.

There is a simple rule of thumb that can help us estimate how many sawtooth curves have to be analyzed simultaneously before their set of accumulation points is reduced to manageable size. Extensive experimentation has shown that this estimate is realistic but not fail-safe. A formal analysis of the question can be found in Section IV.

Let l be the number of sawtooth curves we superimpose in our diagram. Consider the pth minimum of the a_1 curve, which is located at $W = pM_0/a_1$. The closest minimum of the a_i curve can be anywhere in the interval

$$[pM_0/a_1 - M_0/2a_i, pM_0/a_1 + M_0/2a_i],$$

whose length is $M_0/a_i \approx 1$. By making the reasonable (but unrigorous) assumption that the actual locations of the various a_i minima in these intervals are independent random variables with uniform probability distributions, we can estimate the probability that the minima of the $a_2,\cdots,a_l$ curves are all close enough to the pth minimum of the a_1 curve by

$$2^{-n+1}\cdot 2^{-n+2}\cdots 2^{-n+l-1}\approx 2^{-ln+n+l^2/2}.$$

Since we must consider a_1 possible values of p, the expected number of accumulation points is

$$a_1\cdot 2^{-ln+n+l^2/2}\approx 2^{dn-ln+n+l^2/2},$$

and this value is smaller than 1 whenever

$$(l-d-1)n>l^2/2.$$

When n is large enough, this condition is satisfied by

$$l>d+1,$$

and thus the number l is a constant that depends on d but not on n. The claim that the expected number of accumulation points is smaller than 1 should not be taken literally, since we know that one accumulation point always exists by construction. However, it is reasonable to assume that in practice the "built in" point will not be accompanied by too many "accidental" points when l is larger that $d+1$. In particular, when $n=100$ and $|M|=200$, $l=4$ seems to be a reasonable candidate for the number of sawtooth curves we have to analyze.

Two problems remain: how to get rid of M_0 (whose value is actually unknown) and how to find the accumulation points of the minima of the l sawtooth curves.

The key observation is that the locations of the accumulation points in Fig. 1 depend on the *slopes* of the curves but not on their *sizes*. If we divide both coordinates in the ith curve by M_0, we get the sawtooth curve of the function $Va_i(\text{mod } 1)$, $0\le V<1$, which is independent of M_0, as shown in Fig. 2. In the new coordinate system, the slope of the curve remains a_i, and the number of minima remains a_i, but the distance between successive minima is reduced to $1/a_i$. The original W_0 parameter is replaced by a new $V_0=W_0/M_0$ parameter, and the allowable distance between this parameter and the closest a_i curve minimum is reduced by a factor of approximately 2^{dn} (from 2^{-n+i-1} to $2^{-dn-n+i-1}$).

The problem of locating the accumulation points of l minima in the new coordinate system can be described by linear inequalities with l integral unknowns. The conditions that the pth minimum of a_1, qth minimum of a_2, rth minimum of a_3, etc., are sufficiently close to each other

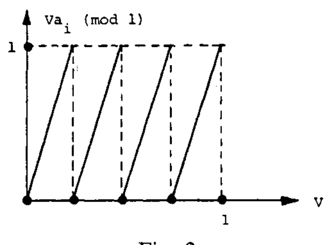

Fig. 2.

are

$$\begin{array}{ll} p,q,r,\cdots,\text{integers}, & 1\le p\le a_1-1,\\ -\epsilon_2\le p/a_1-q/a_2\le\epsilon_2', & 1\le q\le a_2-1,\\ -\epsilon_3\le p/a_1-r/a_3\le\epsilon_3', & 1\le r\le a_3-1,\\ \vdots & \vdots \end{array}$$

where the ϵ_i and ϵ_i' represent the allowable deviations to the right and to the left of p/a_1, respectively. By multiplying each double inequality by its denominators, we get the equivalent system

$$\begin{array}{ll} p,q,r,\cdots,\text{integers}, & 1\le p\le a_1-1,\\ -\delta_2\le pa_2-qa_1\le\delta_2', & 1\le q\le a_2-1,\\ -\delta_3\le pa_3-ra_1\le\delta_3', & 1\le r\le a_3-1,\\ \vdots & \vdots \end{array}$$

which shows that the values of the $a_2,a_3,\cdots$ are simultaneously reduced to small absolute values when multiplied by p and reduced mod a_1.

Fig. 3 is a typical, enlarged section of the superimposed diagram in the vicinity of W_0/M_0. The problem of simultaneously minimizing two numbers by multiplication modulo a third number can be solved with a simple continued fraction algorithm. In the general case, we have to use Lenstra's integer programming algorithm, which is much slower but still polynomial in the size of the coefficients for any fixed number of unknowns. This algorithm is basically a decision procedure that tells us if a certain system of linear inequalities has integral solutions. By using binary search on the successive bits of p, we can find all the accumulation points of the l sawtooth curves.

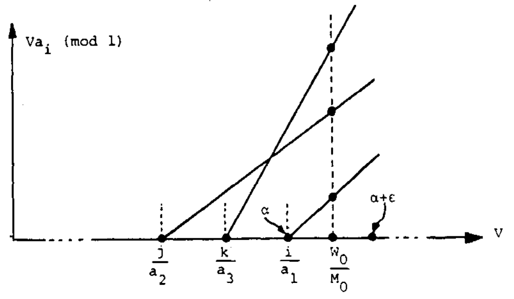

Fig. 3.

To make the running time of the algorithm provable as a polynomial, the algorithm should be aborted if it finds more than a certain number k of accumulation points (say, $k=100$). An extreme example of a bad key is when all the a_i are equal, since in this case all the sawtooth minima are accumulation points. By changing k and l, it is possible to control the fraction of keys for which the algorithm fails to compute a trapdoor pair (see Section IV for more details). Note that failure to solve all the instances of a problem is not a severe handicap in the context of cryptography, since a cryptosystem becomes useless when most of its keys can be efficiently cryptanalyzed.

Before publishing his encryption key, the receiver could permute the order of the elements in the sequence so that a_i would no longer correspond to the ith smallest element in the original superincreasing sequence. This variant of the basic Merkle–Hellman cryptosystem can still be cryptanalyzed in polynomial time by our technique. Since the cryptanalyst has to identify only the l smallest elements in the superincreasing sequence, he can guess them in $O(n^l)$ ways. Incorrect guesses are likely to make the integer programming problem impossible to satisfy, and thus the correct guess can be easily identified. Since l is a constant that does not depend on n the complexity of our technique is increased by just a polynomial factor. Alternatively, the cryptanalyst can relax the tight ϵ bounds on the distance between the various sawtooth minima, so that the integer programming problem can be satisfied not only when the l smallest superincreasing values are correctly guessed, but for any choice of l small enough values. By properly choosing the relaxed values of the ϵ bounds, it is possible to replace the $O(n^l)$ factor by a constant, which in practical applications saves time.

Analyzing the first l sawtooth curves lets us concentrate on a few small regions in which the actual value of W_0/M_0 must be located. Within these regions, the sawtooth curves are piecewise linear with just a few discontinuity points, and thus their values can be expressed and compared without excessive case analysis.

The second part of the algorithm discards from these regions all those subregions in which the sequence of sawtooth values is not super increasing, or its sum is larger than 1. Every rational point in the remaining subregions corresponds to a trapdoor pair. Since W_0/M_0 could not have been discarded by this process, some nonempty subregion must remain.

Let p be one of the values computed in the first part of the algorithm. Consider the interval $[p/a_1, (p+1)/a_1)$ between successive a_1 minima. The expected number of discontinuity points of other curves in it is $O(n)$. Let $V_1, \cdots, V_s$ be the list of V coordinates of these discontinuity points, sorted into increasing order. Between any V_t and V_{t+1}, all the a_i curves look like simple linear segments. The ith linear segment can be expressed by the formula

$$Va_i - \tau_i^t, \qquad V_t \le V < V_{t+1},$$

in which τ_i^t is the number of minima of the a_i curve in $(0, V_t]$ (i.e., τ_i^t/a_i is the point in which the line crosses the V axis).

Consequently, the range, size, and superincreasing conditions can be written as

$$V_t \le V < V_{t+1},$$

$$\sum_{i=1}^{n} \left(Va_i - \tau_i^t\right) < 1,$$

$$\left(Va_i - \tau_i^t\right) > \sum_{j=1}^{i-1} \left(Va_j - \tau_j^t\right), \qquad \text{for } i = 2, \cdots, n.$$

The solution of this system of linear inequalities in V is a (possibly empty) subinterval of $[V_t, V_{t+1})$, and membership of W/M in such a subinterval for some p and t is a necessary and sufficient condition for M and W to be a trapdoor pair.

If the order of the elements of the encryption key is permuted before the elements are published, we have to use a permuted superincreasing condition as well. We cannot guess the correct permutation of the n elements in polynomial time. However, because any superincreasing sequence is also an increasing sequence we can reduce the number of possible permutations that we must consider. We augment the definition of the $V_1, \cdots, V_s$ sequence by including not only the discontinuity points of all the curves, but also the V coordinates of all the intersections between pairs of curves (this may increase the expected value of s from $O(n)$ to $O(n^2)$). Within each new $[V_t, V_{t+1})$ interval, there is a well-defined vertical ordering of the various curves, and thus only one possible permutation of their names sorts them into an increasing sequence. Consequently, only $O(n^2)$ out of the possible $n!$ permutations have to be considered at each accumulation point.

V. Number of Accumulation Points

As we described in Section III, the algorithm is aborted if the l sawtooth curves have at least k accumulation points. In this section we analyze the effect of the l and k parameters on the fraction of the keys for which the algorithm fails, and we show that in a simplified probabilistic model this fraction can be made arbitrarily small.

To simplify the analysis, we assume that a_1 is a fixed prime number and that $a_2, \cdots, a_l$ are independent random variables with uniform probability distribution in $[1, a_1 - 1]$. The primality assumption guarantees that all the modular inverses considered in this section are well defined, but it is not essential and can be replaced by a careful case analysis. We further simplify our notation by assuming that all the δ_i and δ_i' bounds in the integer programming problem are equal, and we denote this common bound by δ.

For each $2 \le i \le l$, we define S_i to be the set of indices of a_1 minima which are close enough to some minimum of a_i.

Definition:

$$S_i = \{1 \le p \le a_1 - 1 | \exists q_i, 1 \le q_i \le a_i - 1, \text{ such that } -\delta \le pa_i - qa_1 \le \delta\}.$$

Since all the S_i are sets of minima of a common a_1 curve, their intersection $S_2 \cap \cdots \cap S_l$ is exactly the set of accumulation points in which an a_1 minimum is simultaneously close to minima of all the other curves.

An alternative characterization of these sets, which is easier to analyze and manipulate follows.

Lemma 1: $S_i = \{j_i a_i^{-1} (\text{mod } a_1) | -\delta \le j_i \le \delta,\ j_i \ne 0\}$.

Proof: When $p \equiv j_i a_i^{-1} (\text{mod } a_1)$, $pa_i \equiv j_i a_i^{-1} a_i \equiv j_i (\text{mod } a_1)$, and thus there is a q_i such that $pa_i = j_i + q_i a_1$. Since $-\delta \le j_i \le \delta$, $pa_i - q_i a_1$ is within the required bounds. The value $j_i = 0$ is not allowed by the definition of S_i. □

The relationship $p \equiv j_i a_i^{-1} (\bmod a_1)$ establishes for each p a one-to-one correspondence between the sequence $a_2, \cdots, a_l$ and the sequence $j_2, \cdots, j_l$. A given p is an accumulation point of $a_2, \cdots, a_l$ if and only if all the corresponding j_i are nonzero integers $[-\delta, \delta]$. Alternatively, when p and a sequence of small j_i are given, there is a unique sequence of a_i for which p is an accumulation point with these j_i indices.

Lemma 2: Let p' and p'' be two accumulation points of $a_2, \cdots, a_l$, and let $j_2', \cdots, j_l'$ and $j_2'', \cdots, j_l''$ be their associated j indices. If $\delta < \sqrt{a_1/2}$, then both sequences are integral multiples of some common $j_2, \cdots, j_l$ sequence for which the greatest common divisor (gcd) $(j_2, \cdots, j_l) = 1$.

Proof: From $p' \equiv j_i' a_i^{-1} (\bmod a_1)$ and $p'' \equiv j_i'' a_i^{-1} (\bmod a_1)$ we can derive the equality

$$a_i \equiv j_i' p'^{-1} \equiv j_i'' p''^{-1} (\bmod a_1),$$

which can be simplified to

$$j_i' j_i''^{-1} \equiv p' p''^{-1} (\bmod a_1).$$

The right-hand side does not depend on i, and thus for any s and t,

$$j_s' j_s''^{-1} \equiv j_t' j_t''^{-1} (\bmod a_1),$$

or

$$j_s' j_t'' \equiv j_t' j_s'' (\bmod a_1).$$

By the assumption of δ, each $j'j''$ product can range only between $-a_1/2$ and $a_1/2$, and thus the equation holds even without the $(\bmod a_1)$ clause

$$j_s' j_t'' = j_t' j_s''.$$

This equality can hold for all s and t only if the j' and j'' sequences are rational multiples of each other. Since they contain only integers, they must be multiples of some common sequence $j_2, \cdots, j_l$ of integers whose gcd is 1. □

Corollary: When $\delta < \sqrt{a_1/2}$ and $S_2 \cap \cdots \cap S_l$ is not empty, there is a basic accumulation point with $j_2, \cdots, j_l$ indices whose gcd is 1, and all the other accumulation points are obtained by multiplying this j_i sequence by $-1, 2, -2, 3, -3$, etc., until some sequence element exceeds δ. When $\delta \geq \sqrt{a_1/2}$, the structure of $S_2 \cap \cdots \cap S_l$ becomes much harder to analyze, and we do not have any simple characterization for it.

Definition: $N(l, k, \delta)$ is the number of $a_2, \cdots, a_l$ sequences in $[1, a_1 - 1]$ for which the intersection $S_2 \cap \cdots \cap S_l$ contains at least k points when the allowable distance is δ.

We are interested in the conditional probability that the l curves have at least k accumulation points when it is known that they have at least one. Since the first event implies the second event, this conditional probability is just

$$N(l, k, \delta)/N(l, 1, \delta).$$

Lemma 3: For any $\delta < \sqrt{a_1/2}$ and $l \geq 3$, there is a constant τ between $3/\pi^2$ and $1/2$ that depends only on l such that

$$N(l, 1, \delta) \approx \tau(a_1 - 1)(2\delta)^{l-1}.$$

Proof: We can overcount the number of $a_2, \cdots, a_l$ sequences that have at least one accumulation point by counting the number of $p, a_2, \cdots, a_l$ sequences in which p is an accumulation point of the a_i. This number is equal to the number of $p, j_2, \cdots, j_l$ sequences in which p is arbitrary and the j_i are nonzero integers in $[-\delta, \delta]$, which is $(a_1 - 1)(2\delta)^{l-1}$. To correct the overcounting, we consider only j_i sequences whose gcd is 1. By Lemma 2, for each a_i, sequence with accumulation points there are exactly two j_i sequences with gcd of 1 (each sequence is the negation of the other). For $l = 3$, the fraction of integer sequences of length $l - 1$ whose gcd is 1 converges to $6/\pi^2$ (see [3]) and for higher values of l this fraction approaches 1. Since each a_i sequence with accumulation points is counted exactly twice, we have to divide this constant by 2 to get the correct constant τ. □

Lemma 4: If $\delta < \sqrt{a_1/2}$, then $N(l, k, \delta) \leq N(l, 1, \delta/\lceil k/2 \rceil)$.

Proof: Let $j_2, \cdots, j_l$ be the sequence of indices with gcd of 1 whose existence is proved in Lemma 2. Since $a_2, \cdots, a_l$ has at least k accumulation points, this j_i sequence can be multiplied by $\lceil k/2 \rceil$, and all its elements will still be in $[-\delta, \delta]$. Consequently, all the original j_i indices are in the range $[-\delta/\lceil k/2 \rceil, \delta/\lceil k/2 \rceil]$, and thus the a_i sequence has at least one accumulation point even when the δ bound is replaced by the tighter $\delta/\lceil k/2 \rceil$ bound. □

We can now prove the main theorem.

Theorem 1: When $\delta < \sqrt{a_1/2}$ and $l \geq 3$, the *conditional probability* $N(l, k, \delta)/N(l, 1, \delta)$ is at most $(1/\lceil k/2 \rceil)^{l-1}$.

Proof:

$$\begin{aligned} N(l, k, \delta)&/N(l, 1, \delta) \\ &\leq N(l, 1, \delta/\lceil k/2 \rceil)/N(l, 1, \delta) \\ &= \tau(a_1 - 1)(2\delta/\lceil k/2 \rceil)^{l-1}/\tau(a_1 - 1)(2\delta)^{l-1} \\ &= (1/\lceil k/2 \rceil)^{l-1}. \end{aligned}$$

□

Example: When $l = 4$, $k = 100$, and $\delta < \sqrt{a_1/2}$, the probability that four randomly chosen sawtooth curves have at least 100 accumulation points when it is known that they have at least one is at most $(1/50)^3 = 1/125\,000$. Thus if we use Lenstra's algorithm to find the accumulation points and abort after 100 points are found, the probability of failure is negligible.

In our cryptanalytic application, δ is approximately 2^{dn-n} and a_1 is approximately 2^{dn}. The condition $\delta < \sqrt{a_1/2}$ is thus equivalent to the condition $d < 2$. We were unable to prove the upper bound of Theorem 1 for cryptosystems in which the ratio d between the modulus size and the number of elements is larger than 2, but Jeff Lagarias (private communication) recently announced a different upper bound which is applicable to the whole range $1 < d < \infty$.

V. Discussion

In this paper we have shown that almost all the single-iteration Merkle–Hellman cryptosystems can be broken in polynomial time and that the probability of failure can be made arbitrarily small. The most time-consuming part of the algorithm is the application of Lenstra's integer programming algorithm, whose worst-case complexity is polynomial in n but exponential in l. The exact complexity of this algorithm is still unknown, and the current upper bound of the form $\operatorname{poly}(n) \cdot \exp((d+2)^3)$ is based on highly pessimistic assumptions about the algorithm's progress at each stage. The average-case complexity of the algorithm is probably much better than the worst-case complexity, and further study is required before the real possibilities and limitations of the cryptanalytic attack proposed in this paper can be quantified.

An important property of the proposed attack is that it is directed at the public key rather than at individual ciphertexts. The cryptanalyst can thus work on standby or low-volume keys even before they are used for the first time and can spend months of computer time on each key if this later enables him to decrypt each ciphertext in microseconds.

The most important problem left open in this paper is the cryptographic security of multi-iteration Merkle–Hellman cryptosystems. At each iteration the randomly chosen modulus must be larger than the sum of the elements, and thus the inverse modular multiplications simultaneously reduce the size of all the elements by at least $\log n$ bits. In principle, this condition finds the (almost certainly) unique interval in which W/M must be located, but not W and M themselves. In the case of single-iteration knapsacks, any such pair was useful, since it generated an easily solvable superincreasing sequence. In the case of multi-iteration knapsacks, on the other hand, only the correct W and M enable the cryptanalyst to do the inverse multiplication properly and to attack the inner iterations one by one.

Acknowledgment

The research reported in this paper is the result of a team effort that started shortly after Merkle and Hellman published their seminal paper. Dozens of researchers from all over the world published papers on this subject, and I have greatly benefited from their ingenious ideas and beautiful insights. They are too numerous to list here, but I gratefully acknowledge the contribution of all of them.

References

[1] J. W. Cassels, *An Introduction to Diophantine Approximations.* Cambridge: Cambridge Univ., 1957.
[2] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22 no. 6, pp. 644–654, Nov. 1976.
[3] D. E. Knuth, *The Art of Computer Programming*, Vol. 2. Reading, MA: Addison-Wesley, 1969.
[4] H. W. Lenstra, "Integer programming with a fixed number of variables," Univ. of Amsterdam, Dept. Mathematics Tech. Rep. vol. 81–03, Apr. 1981. (To appear in Math. Oper. Res.)
[5] R. Merkle and M. Hellman, "Hiding information and signatures in trapdoor knapsacks," *IEEE Trans. Inform. Theory*, vo. IT-24 no. 5, pp. 525–530, Sept. 1978.
[6] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. Ass. Comput. Mach.* vol. 21 no. 2, Feb. 1978.