Electronic Colloquium on Computational Complexity, Report No. 7 (2005)

ECCC

On Random High Density Subset Sums

Vadim Lyubashevsky
Department of Computer Science
University of California at San Diego
La Jolla, CA 92093
vlyubash@cs.ucsd.edu

December 14, 2004

Abstract

In the Subset Sum problem, we are given n integers $a_1, \ldots, a_n$ and a target number t, and are asked to find the subset of the a_i's such that the sum is t. A version of the subset sum problem is the Random Modular Subset Sum problem. In this version, the a_i's are generated randomly in the range $[0, M)$, and we are asked to produce a subset of them such that the sum is $t(mod\, M)$. The hardness of RMSS depends on the relationship between the parameters M and n. When $M = 2^{O(n^2)}$, RMSS can be solved in polynomial time by a reduction to the shortest vector problem. When $M = 2^{O(\log n)}$, the problem can be solved in polynomial time by dynamic programming, and recently an algorithm was proposed that solves the problem in polynomial time for $M = 2^{O(\log^2 n)}$. In this work, we present an algorithm that solves the Random Modular Subset Sum problem for parameter $M = 2^{n^\epsilon}$ for $\epsilon < 1$ in time (and space) $2^{O(\frac{n^\epsilon}{\log n})}$. As far as we know, this is the first algorithm that performs in time better than $2^{\Omega(n^\epsilon)}$ for arbitrary $\epsilon < 1$.

1 Introduction

The Subset Sum (SS) problem is one of the original NP-hard problems. In the standard SS problem, we are given n numbers and a target t, and we are asked to find a subset of the n numbers whose sum is t. A variant of SS is the Random Modular Subset Sum (RMSS). In this variant, we are given a modulus M, a target t, and n numbers generated uniformly at random between 0 and M, and are asked to find a subset of the n numbers whose sum is $t(\text{mod } M)$. The complexity of RMSS depends on the quantity $\frac{n}{\log M}$, which

ISSN 1433-8092

is called the *density* of the instance. If $M = 2^{cn}$ for some $c < 1$, then for almost every target in the range $[0, M)$, some subset of the n given numbers will add up to it. These are the high density instances. If $M = 2^{cn}$ for some $c > 1$ then only a very low fraction of the numbers in the range $[0, M)$ can be written as a sum of some subset of the n numbers. These are the low density instances. All RMSS instances can be solved in time $O(M)$ using dynamic programming, so when $M = O(poly(n))$, the problem can be solved in polynomial time using dynamic programming or by other more efficient methods [3]. When $M = \Omega(2^{n^2})$, the problem can be solved in polynomial time by a reduction to the approximate shortest vector problem [5],[2]. In [4], Impagliazzo and Naor show that the hardest instances of RMSS are the ones where the density is 1 (i.e. $M = 2^n$). In the same paper, the authors show if that RMSS is a one-way function, then it is also a pseudo-random generator (for low densities of RMSS) and a one-way hash function (for high densities of RMSS). The assumption that RMSS is a one-way function for values of $M = O(2^n)$ is very plausible since the best known algorithms take time $O(2^{\frac{n}{2}})$ when $M = 2^n$.
In this paper, we concentrate on high density instances of RMSS.

1.1 Our Contributions and Related Work

Formally, the Random Modular Subset Sum problem is the following:

Random Modular Subset Sum:
INSTANCE: $M, t, a_1, \ldots a_n$ where the a_i's are independent uniformly distributed integers in the range $[0, M)$ and t is any number.
SOLUTION: $x_1, \ldots, x_n$, where $x_i \in \{0, 1\}$ and $\sum_{i=1}^{n} x_i a_i = t (\text{mod } M)$.

In this paper we propose a randomized algorithm for solving Random Modular Subset Sum instances with the parameter $M = 2^{n^\epsilon}$ for $\epsilon < 1$. It's possible that ϵ may be a function of n, but then we require that $\lim_{n\to\infty} \epsilon(n) < 1$. The algorithm runs in time (and space) $2^{O(\frac{n^\epsilon}{\log n})}$ and has success probability at least $1 - 2^{-\Omega(n^\epsilon)}$.
Throughout the paper, we will assume that $n^\epsilon = \omega(\log n)$, because otherwise dynamic programming solves the problem in polynomial time and we are not striving to improve on that.

Our main tool is the algorithmic idea of Wagner[8]. Wagner considers the following problem: given n lists each containing $M^{\frac{1}{\log n}}$ independent integers

uniformly distributed in the interval $[0, M)$ and a target t, find one element from each list such that the sum of the elements is $t(mod\, M)$. Wagner shows that there exists an algorithm that in time $nM^{O(\frac{1}{\log n})}$, returns a list of solutions. The number of solutions that the algorithm returns is a random variable with expected value 1. That is, we expect to find one solution. Notice that this does not imply that the algorithm finds a solution with high probability because, for example, it can find 2^n solutions with probability 2^{-n} and find 0 solutions every other time. By inspecting Wagner's algorithm, there is no reason to assume such pathological behavior, and thus the algorithm works well as a cryptanalytic tool (as was intended by the author). In our work, we make some modifications to the parameters of the algorithm and are able to obtain a proof that the algorithm indeed succeeds with high probability in finding a solution. We believe that this proof may be of independent interest since Wagner's technique appears to have wide applications. This proof combined with one other result allows us to obtain an algorithm for subset sum.
Recently, Flaxman and Przydatek [1] proposed an algorithm that solves instances of Random Modular Subset Sum with $M = 2^{O(\log^2 n)}$ in polynomial time. Our result is a generalization of theirs. If we set $n^\epsilon = \log^2 n$, then we are able to solve Random Subset Sum with $M = 2^{\log^2 n}$ in time $2^{O(\frac{\log^2 n}{\log n})} = 2^{O(\log n)} = O(n^c)$ for some constant c. One additional small advantage is that in [1], M had to be less than $2^{\frac{\log^2 n}{16}}$, while there is no such restriction in our algorithm. The algorithm will run in polynomial time for $M = 2^{c\log^2 n}$ for any constant c.

1.2 Organization

To get the high level idea of the algorithm it's probably best to skip directly to Corollary 3.2 and then go on to Section 4, where the algorithm is presented. In Section 3, we prove that a modified version of Wagner's algorithm succeeds with high probability which then allows us to prove Corollary 3.2. The proof uses a technical lemma which is proved in Section 2.

1.3 Preliminaries

Statistical distance is a measure of how far apart two probability distributions are. In this subsection, we review the definition and some of the basic properties of statistical distance. The proofs may be found in [6].

Definition 1.1. *Let X and Y be random variables over a countable set A.*

The statistical distance between X and Y, denoted $\Delta(X,Y)$, is

$$\Delta(X,Y) = \frac{1}{2}\sum_{a\in A} |Pr[X=a] - Pr[Y=a]|$$

Proposition 1.2. *Let $X_1, \ldots X_k$ and $Y_1, \ldots Y_k$ be two lists of independent random variables. Then*

$$\Delta((X_1,\ldots,X_k),(Y_1,\ldots,Y_k)) \leq \sum_{i=1}^{k} \Delta(X_i,Y_i)$$

Proposition 1.3. *Let X, Y be two random variables over a set A. For any predicate $f : A \to \{0,1\}$,*

$$|Pr[f(X)=1] - Pr[f(Y)=1]| \leq \Delta(X,Y)$$

Proposition 1.4. *(Chernoff Bound) Let $S_1, \ldots, S_m$ be a sequence of m independent Bernoulli random variables, such that $Pr[S_i = 1] = p$. Let $S = S_1 + \ldots + S_m$. Then for $0 \leq \gamma \leq 1$, $Pr[S < (p-\gamma)m] \leq e^{-2m\gamma^2}$.*

2 List Merging Lemma

Imagine that we have two lists of integers which are independently, and uniformly distributed in the range $[-R, R)$ for some integer R. What we are trying to do is to create a new list that also consists of independent, uniformly distributed numbers in the range $[-S, S)$ for some $S < R$, with the requirement that every integer in the new list is a sum of one number from the first list and one number from the second list. The following lemma provides a relationship between how many numbers there are in the original lists and how many numbers there are in the new list depending on the relationship between R and S. Intuitively, the larger the difference between R and S, the fewer numbers we will have in the new list.

Lemma 2.1. *Let L_1 and L_2 be lists of numbers in the range $\left[-\frac{R}{2}, \frac{R}{2}\right)$ and let p and c be positive reals such that $e^{-\frac{c}{12}} < p < \frac{1}{8}$. Let L_3 be a list of numbers $a_1 + a_2$ such that $a_1 \in L_1, a_2 \in L_2$, and $a_1 + a_2 \in \left[-\frac{Rp}{2}, \frac{Rp}{2}\right)$. If L_1 and L_2 each contain at least $\frac{c}{p^2}$ independent, uniformly distributed numbers in $\left[-\frac{R}{2}, \frac{R}{2}\right)$, then with probability greater than $1 - e^{-\frac{c}{4}}$, L_3 contains at least $\frac{c}{4p^2}$ independent, uniformly distributed numbers in the range $\left[-\frac{Rp}{2}, \frac{Rp}{2}\right)$.*

Proof. To prove the lemma, we will give an algorithm to construct the list L_3 that will consist entirely of independent, uniformly distributed numbers in the range $\big[-\frac{Rp}{2}, \frac{Rp}{2}\big)$. And with high probability, there will be at least $\frac{c}{4p^2}$ elements in this list. First, we will prove two simple lemmas about the distribution of the numbers from lists L_1 and L_2 in the range $\big[-\frac{R}{2}, \frac{R}{2}\big)$.

Lemma 2.2. *With probability greater than $1 - e^{-\frac{c}{2}}$, any interval of length Rp that is fully contained inside range $\big[-\frac{R}{2}, \frac{R}{2}\big)$, has at least $\frac{c}{2p}$ numbers from L_1 (similarly L_2).*

Proof. L_1 (similarly L_2) has $\frac{c}{p^2}$ independent uniformly distributed numbers in the range $\big[-\frac{R}{2}, \frac{R}{2}\big)$. For each one, the probability that it falls into a specified interval that has size Rp is p. Let $S = S_1 + \ldots + S_{\frac{c}{p^2}}$ where $S_i = 0$ if the i^{th} number does not fall in the interval and $S_i = 1$ if the i^{th} number falls into the interval. By applying the Chernoff bound in Proposition 1.4 with $\gamma = \frac{p}{2}, m = \frac{c}{p^2}$, we get $Pr[S < \frac{c}{2p}] \leq e^{-\frac{c}{2}}$. □

Lemma 2.3. *With probability greater than $1 - e^{-\frac{c}{3}}$, any interval of length $\frac{Rp}{2}$ that is fully contained inside range $\big[-\frac{R}{2}, \frac{R}{2}\big)$, has at least $\frac{c}{12p}$ numbers from L_1 (similarly L_2).*

Proof. The proof is almost identical to the one for Lemma 2.2. □

Now we continue with the proof of Lemma 2.1. Consider the intervals

$$I_k = \left[-\frac{R}{2} + \frac{(1+3k)Rp}{2}, -\frac{R}{2} + \frac{(2+3k)Rp}{2}\right)$$

for integers k where $0 \leq k < \frac{1}{2p}$. Notice that the intervals are disjoint, have size $\frac{Rp}{2}$, and there is a distance of at least Rp between every two intervals. From each interval I_k, select an a_k from L_1 that is in the interval. By Lemma 2.3, such an a_k exists in each interval with probability greater than $1 - e^{-\frac{c}{3}}$ (There are actually $\frac{c}{12p}$ numbers in the region with high probability, but we only need one). So the probability that each of the $\frac{1}{2p}$ intervals contains an element from L_1 is greater than $1 - \frac{1}{2p}e^{-\frac{c}{3}}$. We need to notice two things about the a_k's. First, by our choice of the intervals, the distance between any a_i and a_j is at least Rp. And second, since $p < \frac{1}{8}$, $-\frac{R}{2} + \frac{Rp}{2} \leq a_k \leq \frac{R}{2} - \frac{Rp}{2}$. Now, for each k, define B_k to be the list of numbers in L_2 in the range $\big[-a_k - \frac{Rp}{2}, -a_k + \frac{Rp}{2}\big)$. Now we notice that since all the a_k are at a distance

of at least Rp from each other, the lists B_k are disjoint. By Lemma 2.2, with probability greater than $1 - e^{-\frac{c}{2}}$ each B_k contains at least $\frac{c}{2p}$ numbers. So the probability that each of the B_k contain more than $\frac{Rp}{2}$ elements is greater than $1 - \frac{1}{2p}e^{-\frac{c}{2}}$. Since adding a_k to any element in B_k produces an element in the range $\big[-\frac{Rp}{2}, \frac{Rp}{2}\big)$, we can create $\frac{1}{2p} \cdot \frac{c}{2p} = \frac{c}{4p^2}$ such elements. They are uniformly distributed in the range $\big[-\frac{Rp}{2}, \frac{Rp}{2}\big)$ because all the numbers in B_k are uniformly distributed in the range $\big[-a_k - \frac{Rp}{2}, -a_k + \frac{Rp}{2}\big)$, and they are independent because all the B_k are disjoint. By the union bound, the probability of having $\frac{c}{4p^2}$ elements is greater than

$$\begin{aligned} 1 - \frac{1}{2p}e^{-\frac{c}{3}} - \frac{1}{2p}e^{-\frac{c}{2}} &> 1 - \frac{1}{p}e^{-\frac{c}{3}} \\ &> 1 - e^{-\frac{c}{4}} \end{aligned}$$

The last inequality follows from the assumption that $e^{-\frac{c}{12}} < p$. □

Since we will be using the algorithm in the proof of Lemma 2.1 as a subroutine in the next section, it would be useful to know the running time of constructing the list L_3 from L_1 and L_2. The algorithm MergeLists(L_1, L_2) takes two lists each containing $\frac{c}{p^2}$ numbers and returns a list L_3 with the properties described in the statement of the lemma.

MergeLists(L_1, L_2)
Sort L_1. Sort L_2.
For $k = 0$ to $\frac{1}{2p}$
    Pick an a_k from L_1 that's in the interval I_k (As defined in the proof).
    Define list B_k to be elements from L_2 in the range $\big[-a_k - \frac{Rp}{2}, -a_k + \frac{Rp}{2}\big)$
    For every element $b \in B_k$
        Add the element $a_k + b$ to the list L_3.

The step that takes the longest time in the algorithm is the sorting of the lists. Thus the algorithm performs $O(\frac{c}{p^2} \log \frac{c}{p^2})$ arithmetic operations.

3 Proof of the Modified Wagner's Algorithm

In this section, we use the results from Section 2 in order to obtain a proof that a modified version of Wagner's algorithm [8] succeeds with high probability. Our algorithm takes as input b lists each containing $kM^{\frac{2}{\log b}}$ numbers

uniformly and independently distributed in the range $[-\frac{M}{2}, \frac{M}{2})$, and returns b numbers (one from each list) whose sum is zero. The probability of success depends on the parameters b and k.

Theorem 3.1. *Given b independent lists each consisting of $kM^{\frac{2}{\log b}}$ independent uniformly distributed elements in the range $[-\frac{M}{2}, \frac{M}{2})$, there exists an algorithm that with probability at least $1 - be^{-\frac{k}{4b^2}}$ returns one element from each of the b lists such that the sum of the b elements is 0. The running time of this algorithm is $O(b \cdot kM^{\frac{2}{\log b}} \cdot \log(kM^{\frac{2}{\log b}}))$ arithmetic operations.*

Proof. The algorithm will be building a tree whose nodes are lists of numbers. The initial b lists will make up level 0 of the tree. Define intervals

$$I_i = \left[-\frac{M^{1-\frac{i}{\log b}}}{2}, \frac{M^{1-\frac{i}{\log b}}}{2} \right)$$

for integers $0 \leq i \leq \log b$. (All the logarithms are base 2). Notice that all the numbers in the lists at level 0 are in the range I_0. Level 1 of the tree will be formed by pairing up the lists on level 0 in any arbitrary way, and for each pair of lists, L_1 and L_2 (there are $\frac{b}{2}$ such pairs), create a new list L_3 whose elements are in the range I_1 and of the form $a_1 + a_2$ where $a_1 \in L_1$ and $a_2 \in L_2$. So level 1 will have half the number of lists as level 0, but the numbers in the lists will be in a smaller range. We construct level 2 in the same way; that is, we pair up the lists in level 1 and for each pair of lists L_1 and L_2, create a new list L_3 whose elements are in the range I_2 and of the form $a_1 + a_2$ where $a_1 \in L_1$ and $a_2 \in L_2$. Notice that if we continue in this way, then level $\log b$ will have one list of numbers in the interval $I_{\log b}$ where each number is the sum of b numbers, one from each of the original b lists at level 0. And since the only possible integer in $I_{\log b}$ is 0, if the list at level $\log b$ is not empty, then we are done. So what we need to prove is that the list at level $\log b$ is not empty with high probability.

Claim: *With probability at least $1 - be^{-\frac{k}{4b^2}}$, for $0 \leq i \leq \log b$, level i consists of $\frac{b}{2^i}$ lists each containing $\frac{k}{4^i} \cdot M^{\frac{2}{\log b}}$ independent, uniformly distributed numbers in the range I_i.*

Notice that proving this claim immediately proves the theorem. The claim will be proved by induction. It's true for level 0. Assume that it is true for some level i where $0 \leq i < \log b$. We want to show that it is true for level $i + 1$. We pair up the $\frac{b}{2^i}$ lists at level i and from each pair, create one list

in the range I_{i+1}. So we have $\frac{b}{2^{i+1}}$ lists with numbers in the range I_{i+1}. To prove that each list contains $\frac{k}{4^i} \cdot M^{\frac{2}{\log b}}$ independent, uniformly distributed elements in the range I_{i+1}, we will invoke Lemma 2.1. We use the lemma with the following parameters:

$$R = M^{1-\frac{i}{\log b}},\ c = \frac{k}{4^i},\ p = M^{-\frac{1}{\log b}}$$

Thus if we take two lists, L_1 and L_2, at level i, by the inductive hypothesis they each have $\frac{c}{p^2}$ independent uniformly distributed elements in the range $[-\frac{R}{2}, \frac{R}{2})$. By Lemma 2.1, we can create a new list, L_3, with $\frac{c}{4p^2} = \frac{k}{4^{i+1}} \cdot M^{\frac{2}{\log b}}$ independent, uniformly distributed elements in the range

$$\left[\frac{-Rp}{2}, \frac{Rp}{2}\right) = \left[-\frac{M^{1-\frac{i+1}{\log b}}}{2}, \frac{M^{1-\frac{i+1}{\log b}}}{2}\right) = I_{i+1}$$

where each element is the sum of a number from L_1 and a number from L_2. The probability of success is at least $1 - e^{-\frac{c}{4}}$. Since $c = \frac{k}{4^i}$ and $i < \log b$, $c > \frac{k}{b^2}$ and so the probability of success for each list merge is at least $1 - e^{-\frac{k}{4b^2}}$. The total number of times that we combine lists during the whole algorithm is $b-1$, so the probability that everything is successful is at least $1 - be^{-\frac{k}{4b^2}}$. Since we are combining lists of at most $kM^{\frac{2}{\log b}}$ elements, the running time, as stated in the previous section, is $O(kM^{\frac{2}{\log b}} \cdot \log(kM^{\frac{2}{\log b}}))$, and since we are doing $b-1$ list combines, the total running time is $O(b \cdot kM^{\frac{2}{\log b}} \cdot \log(kM^{\frac{2}{\log b}}))$. □

Corollary 3.2. *Given b independent lists each consisting of $kM^{\frac{2}{\log b}}$ independent uniformly distributed elements in the range $[0, M)$, and a target t, there exists an algorithm that with probability at least $1 - be^{-\frac{k}{4b^2}}$ returns one element from each of the b lists such that the sum of the b elements is $t (mod\, M)$. The running time of this algorithm is $O(b \cdot kM^{\frac{2}{\log b}} \cdot \log(kM^{\frac{2}{\log b}}))$.*

Proof. First, we subtract the target t from every element in the first list. Then we transform every number (in every list) in the interval $[0, M)$ into an equivalent number modulo M in the interval $[-\frac{M}{2}, \frac{M}{2})$. We do this by simply subtracting M from every number greater or equal to $\frac{M}{2}$. Now we apply Theorem 3.1. Since exactly one number from every list got used, it means that $-t$ got used once. And since the sum is 0, we found an element from each list such that their sum is $t (mod\, M)$. □

4 Subset Sum Algorithm

In this section, we present the algorithm that will solve Random Modular Subset Sum for parameter $M = 2^{n^\epsilon}$. The correctness of the algorithm will rely on Corollary 3.2 from the previous section as well as a proposition which is proved in [4].
Consider the following function X which defines a distribution on the range $[0, M)$:
X {
pick n random $x_i \in \{0, 1\}$
output $\sum_{i=1}^{n} x_i a_i (mod\, M)$
}
We want to know when the distribution of the above function is statistically close to the uniform distribution (U) in the range $[0, M)$. This of course depends on the a_i's, M, and n. If, for example, all the a_i are 0, then X will always be 0, and thus $\Delta(X, U) \approx 1$. What is proved in [4] is that if $M = 2^{cn}$ where $c < 1$, then for almost all choices of a_i, $\Delta(X, U)$ is small. Formally,

Proposition 4.1. *(Impagliazzo, Naor [4])For a given modulus $M = 2^{cn}$ where $c < 1$, for all but a $2^{-\frac{(1-c)n}{4}}$ fraction of the possible choices for $(a_1, \ldots, a_n)$, $\Delta(X, U) < 2^{-\frac{(1-c)n}{4}}$; where X is the distribution described above and U is the uniform distribution on the range $[0, M)$.*

We will say that, for a given M, the list $(a_1, \ldots, a_n)$ is *"well-distributed"* if using these a_i's in the function X, we get $\Delta(X, U) < 2^{-\frac{(1-c)n}{4}}$. The above proposition says that if we pick the a_i randomly, then with probability greater than $1 - 2^{-\frac{(1-c)n}{4}}$, the list of a_i's we end up with will be *"well-distributed"*.
The next lemma says that if we have a *"well-distributed"* list of a_i, then we can create another (potentially bigger) list by taking random subsets of the a_i, and the statistical distance of this new list from a list of uniformly distributed numbers will be small.

Lemma 4.2. *For a given n "well-distributed" numbers $a_1, \ldots, a_n$ modulo $M = 2^{cn}$ for some $c < 1$, the statistical distance between the lists $(X_1, \ldots, X_m)$ and $(U_1, \ldots, U_m)$, where the X_i are random variables distributed according to function X above and U_i are random variables distributed uniformly in the range $[0, M)$, is at most $m2^{-\frac{(1-c)n}{4}}$.*

Proof. All the X_i are independent since we just sum up a random subset of the a_i's, and the U_i are independent since we are choosing them independently from the uniform distribution. So we just apply Proposition 1.2. □

The above proposition and lemma basically say that if we have "many" random numbers and a "small" modulus M, then taking random subsets of these numbers is, with high probability, essentially equivalent to picking random numbers modulo M. Now we are ready to describe the algorithm.

The idea for our algorithm will be to first take the n given numbers modulo $M = 2^{n^\epsilon}$ and construct $\frac{1}{2}n^{1-\epsilon}$ lists each containing $n^2 M^{\frac{2}{\log \frac{1}{2}n^{1-\epsilon}}}$ numbers that are almost independent and uniformly distributed in $[0, M)$. Of course, we can't do that directly because we don't have enough numbers. So what we will do is break up the n numbers into $\frac{1}{2}n^{1-\epsilon}$ groups each containing $2n^\epsilon$ numbers, and then for each group, we generate a list of $n^2 M^{\frac{2}{\log \frac{1}{2}n^{1-\epsilon}}}$ numbers by taking sums of random subsets of the elements in the group. By Proposition 4.1 and Lemma 4.2, we can say that with high probability the lists are not too far from being uniformly distributed. Now that we have lists that are big enough, we can apply Corollary 3.2 which states that we can find one number from each list such that the sum of the numbers is the target. And since every number in the list is some subset of the numbers in the group from which the list was generated, we have found a subset of the original n numbers which sums to the target.

Theorem 4.3. *Given n numbers, $a_1, \ldots, a_n$, that are independent and uniformly distributed in the range $[0, M)$ where $M = 2^{n^\epsilon}$ for some function $\epsilon(n) < 1$ that is bounded away from 1, and a target t, there exists an algohrithm that in time $2^{O\left(\frac{n^\epsilon}{\log n}\right)}$ and with probability at least $1 - 2^{-\Omega(n^\epsilon)}$ will find $x_1, \ldots, x_n \in \{0, 1\}$ such that $\sum_{i=1}^{n} a_i x_i = t (mod\, M)$.*

Proof. We will show that the below *SubsetSum* algorithm satisfies the claim in the theorem.

$SubsetSum(a_1, \ldots, a_n, t, M)$ /** Here $M = 2^{n^\epsilon}$
(1) Break up the n numbers into $\frac{1}{2}n^{1-\epsilon}$ groups each containing $2n^\epsilon$ numbers.
(2) For group $i = 1$ to $\frac{1}{2}n^{1-\epsilon}$ do
(3) List $L_i = GenerateListFromGroup(\{a_j | a_j \in group\ i\}, M)$
(4) Apply the algorithm from Corollary 3.2 to $L_1, \ldots, L_{.5n^{1-\epsilon}}, t, M$.

GenerateListFromGroup$(\{a_1, \ldots, a_m\}, M)$
(5) Initialize list L to be an empty list.
(6) For $i = 1$ to $n^2 2^{\frac{2n^\epsilon}{\log .5n^{1-\epsilon}}}$ do /** Notice that $n^2 2^{\frac{2n^\epsilon}{\log .5n^{1-\epsilon}}} = n^2 M^{\frac{2}{\log \frac{1}{2}n^{1-\epsilon}}}$
(7) Generate m random $x_j \in \{0, 1\}$
(8) Add the number $\sum_{j=1}^{m} a_j x_j (mod\, M)$ to list L.
(9) Return L.

If we assume for a second that the lists $L_1, \ldots L_{.5n^{1-\epsilon}}$ in line (4) consist of independent, uniformly distributed numbers, then we are applying the algorithm from Corollary 3.2 with parameters $b = \frac{1}{2}n^{1-\epsilon}$ and $k = n^2$. So with probability

$$1 - be^{-\frac{k}{4b^2}} = 1 - \frac{1}{2}n^{1-\epsilon}e^{-\frac{n^2}{4(.5n^{1-\epsilon})^2}} = 1 - e^{-\Omega(n^{2\epsilon})}$$

the algorithm will return one element from each list such that the sum of the elements is $t(mod\, M)$. And this gives us the solution to the subset sum instance. The running time of the algorithm is

$$\begin{aligned} O(b \cdot kM^{\frac{2}{\log b}} \cdot \log(kM^{\frac{2}{\log b}})) &= O((b \cdot kM^{\frac{2}{\log b}})^2) \\ &= O\Big(\Big(\frac{1}{2}n^{1-\epsilon}n^2 M^{\frac{2}{\log .5n^{1-\epsilon}}}\Big)^2\Big) \\ &= O\Big(\Big(\frac{1}{2}n^{1-\epsilon}n^2 2^{\frac{2n^\epsilon}{\log .5n^{1-\epsilon}}}\Big)^2\Big) \\ &= 2^{O\left(\frac{n^\epsilon}{(1-\epsilon)\log n}\right)} \\ &= 2^{O\left(\frac{n^\epsilon}{\log n}\right)} \end{aligned}$$

The problem is that each list is not generated uniformly and independently from the interval $[0, M)$. But we will show that with high probability, they are close enough to being uniform. We will first observe that with high probability, the a_i's in every group are *"well-distributed"*. Since each group consists of $2n^\epsilon$ numbers, and the modulus is 2^{n^ϵ}, Proposition 4.1 implies that with probability at least

$$1 - 2^{-\frac{(1-.5)2n^\epsilon}{4}} = 1 - 2^{-\frac{n^\epsilon}{4}}$$

the numbers in any one group are *"well-distributed"*, and since there are $\frac{1}{2}n^{1-\epsilon}$ groups, the probability that all the groups are *"well-distributed"* is at

least

$$1 - \frac{1}{2}n^{1-\epsilon}2^{-\frac{n^\epsilon}{4}} = 1 - 2^{-\Omega(n^\epsilon)}$$

Now, assuming that the numbers in each group are *"well-distributed"*, we can apply Lemma 4.2 which tells us that the statistical distance between each list that we created and lists of independent, uniformly distributed numbers in the range $[0, M)$, is at most $m2^{-\frac{(1-c)2n^\epsilon}{4}}$ where $c = \frac{1}{2}$ and $m = n^2 2^{\frac{2n^\epsilon}{\log .5n^{1-\epsilon}}}$. Since there are a total of $\frac{1}{2}n^{1-\epsilon}$ lists, the statistical distance between the $\frac{1}{2}n^{1-\epsilon}n^2 2^{\frac{2n^\epsilon}{\log .5n^{1-\epsilon}}}$ numbers generated by our algorithm and numbers generated from the uniform distribution is at most

$$\frac{1}{2}n^{1-\epsilon}n^2 2^{\frac{2n^\epsilon}{\log .5n^{1-\epsilon}}}2^{-\frac{2n^\epsilon}{8}} = 2^{-\Omega(n^\epsilon)}$$

for all $\epsilon < 1$. So the statistical distance between the input that we give to Corollary 3.2 and uniformly generated input is $2^{-\Omega(n^\epsilon)}$. Thus, by Proposition 1.3, the probability that the algorithm from Corollary 3.2 succeeds on this input is at most $2^{-\Omega(n^\epsilon)}$ less than the probability that it succeeds on the uniformly generated input. Thus the probability of success is at least

$$1 - e^{-\Omega(n^{2\epsilon})} - 2^{-\Omega(n^\epsilon)} = 1 - 2^{-\Omega(n^\epsilon)}$$

So to summarize, the probability that every group is *"well-distributed"* is $1 - 2^{-\Omega(n^\epsilon)}$ and the probability that the algorithm succeeds given that all the groups are *"well-distributed"* is also $1 - 2^{-\Omega(n^\epsilon)}$. Thus the probability that our algorithm succeeds is at least

$$1 - 2 \cdot 2^{\Omega(n^\epsilon)} = 1 - 2^{-\Omega(n^\epsilon)}$$

□

5 Conclusion and Open Problems

In this work, we showed that Random Modular Subset Sum with n numbers and modulus 2^{n^ϵ}, for $\epsilon < 1$, can be solved with high probability in time $2^{O(\frac{n^\epsilon}{\log n})}$.

Besides the obvious open problem of improving the running time of the algorithm, the other open problem (which we believe to be easier than improving the running time) is what happens when $\epsilon = 1$. For example, if the modulus is $M = 2^{.5n}$, there still, with very high probability, exists a solution to the subset sum problem, but we cannot apply the algorithm in this paper to

this problem since we cannot break the n numbers into enough groups that are adequately large.

6 Acknowledgements

I am very grateful to Russell Impagliazzo and Daniele Micciancio for their help and advice with this paper.

References

[1] Abraham Flaxman, Bartosz Przydatek. "Solving Medium-Density Subset Sum Problems in Expected Polynomial Time" STACS 2005.

[2] Alan Frieze. "On the Lagarias-Odlyzko Algorithm for the Subset Sum Problem" SIAM J. Comput., vol. 15 1986 pp. 536-539

[3] Zvi Galil, Oded Margalit. "An Almost Linear Time Algorithm for the Dense Subset Sum Problem", SIAM J. Comput., vol 20 1991, pp. 1157-1189

[4] Russell Impagliazzo, Moni Naor. "Efficient Cryptographic Schemes Provably as Secure as Subset Sum" Journal of Cryptology Volume 9, Number 4, 1996, pp. 199-216

[5] Jeffrey C. Lagarias, Andrew M. Odlyzko. "Solving Low Density Subset Sum Problems", J. of the ACM, Volume 32, 1985 pp. 229-246

[6] Daniele Micciancio, Shafi Goldwasser. *Complexity of Lattice Problems: A Cryptographic Perspective.* Kluwer Academic Publishers, Boston, Massachussetts, 2002.

[7] Richard Schroeppel, Adi Shamir. "A $T = O(2^{(n/2)})$, $S = O(2^{(n/4)})$ Algorithm for Certain NP-Complete Problems." SIAM J. Comput. vol. 10 1981 pp. 456-464

[8] David Wagner. "A Generalized Birthday Problem" CRYPTO 2002, LNCS, Springer-Verlag, pp. 288-303

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'