Collision-Free Hashing from Lattice Problems

Oded Goldreich* Shafi Goldwasser† Shai Halevi
MIT - Laboratory for Computer Science
545 Technology Square, Cambridge, MA 02139

October 3, 1996

Abstract

Recently Ajtai described a construction of one-way functions whose security is equivalent to the difficulty of some well known approximation problems in lattices. We show that essentially the same construction can also be used to obtain collision-free hashing.

---

*On sabatical leave from Weizmann Institute of Science, Rehovot, Israel.
†Shafi's grant ACK.

1 Introduction

In a recent work [1] Ajtai described a problem that is *hard on the average* if some well-known lattice problems are *hard to approximate in the worst case*, and demonstrated how this problem can be used to construct one-way functions. We show that Ajtai's method can also be used to construct families of collision-free hash functions. Furthermore, a slight modification of this construction yields families of functions which are both universal and collision-free.

1.1 The Construction

The construction is very simple. For security parameter n we pick a random $n \times m$ matrix M with entries from $\mathcal{Z}_q$, where m and q are chosen so that $n \log q < m < \frac{q}{2n^4}$ and $q = O(n^c)$ for some constant $c > 0$. (E.g., $m = n^2, q = n^7$). See Section 3 for a discussion about the choice parameters. The hash function $h_M : \{0,1\}^m \rightarrow \mathcal{Z}_q^n$ is then defined for $s = s_1 s_2 \cdots s_m \in \{0,1\}^m$ as

$$h_M(s) = Ms \bmod q = \sum_i s_i M_i \bmod q$$

where M_i is the i'th column of M.

Notice that h_M's input is m-bit long, and its output is $n \log q$ bit long. Since we chose the parameters so that $m > n \log q$, there are collisions in h_M. As we will argue below, however, it is infeasible to find any of these collisions unless some well known lattice problems have good approximation in the worst case. It follows that, although it is easy to find solutions for the equations $Ms \equiv 0 \pmod q$, it seems hard to find binary solutions (i.e., a vector $s \in \{0,1\}^m$ in the solution space).

Remark. Using our notation, the candidate one-way function introduced by Ajtai is $f(M, s) \stackrel{\text{def}}{=} (M, h_M(s))$. We note that this function is *regular* (cf., [3]); that is, the number of preimage of any image is about the same. (Furthermore, for most M's the number of pre-images under h_M of almost all imapges is about the same.) To the best of our knowledge, it easier (and more efficient) to construct a pseudo-random generator based on a regular one-way function than based on an arbitrary one-way function (cf., [3] and [4]).

1.2 A Modification

A family of hash functions is called *universal* if a function uniformly selected in the family maps every two images uniformly on its range in a pairwise indepedent manner [2]. To obtain a family of functions which is both universal and collision-free, we slightly modify the above construction. First we set q to be a *prime* of the desired size. Then, in addition to picking a random matrix $M \in \mathcal{Z}_q^{n \times m}$, we also pick a random vector $r \in \mathcal{Z}_q^n$. The function $h_{M,r} : \{0,1\}^m \rightarrow \mathcal{Z}_q^n$ is then defined

$$\text{for } s = s_1 \cdots s_m \in \{0,1\}^m, \; h_M(s) = Ms + r \; (mod \; q) = r + \sum_i s_i M_i \; (mod \; q)$$

The modified construction resembles the standard construction of universal hash functions [2], with calculations done over $\mathcal{Z}_q$ instead of over $\mathcal{Z}_2$.

2 Formal Setting

In this section we give a brief description of some well known lattice problems, describe Ajtai's reduction, and our version of it.

2.1 Lattices

Definition 1: Given a set of n linearly independent vectors in $\mathcal{R}^n$, $V = \langle v_1, \cdots, v_n \rangle$, we define the lattice spanned by V as the set of all possible linear combinations of the v_i's with integral coefficients, namely

$$L(V) \stackrel{\text{def}}{=} \left\{ \sum_i a_i v_i \ : \ a_i \in \mathcal{Z} \text{ for all } i \right\}$$

We call V the basis of the lattice $L(V)$. We say that a set of vectors $L \subset \mathcal{R}^n$ is a lattice if there is a basis V such that $L = L(V)$.

It is convenient to picture a lattice L in $\mathcal{R}^n$ as a "tiling" of the space $\mathcal{R}^n$ using small parallelepipeds, with the lattice points being the vertices of these parallelepipeds. The parallelepipeds themselves are spanned by some basis of L. We call the parallelepiped that are spanned by the "shortest basis of L" (the one whose vectors have the shortest Euclidean norm) the *basic cells* of the lattice L. See Figure 1 for an illustration of these terms in a simple lattice in $\mathcal{R}^2$.

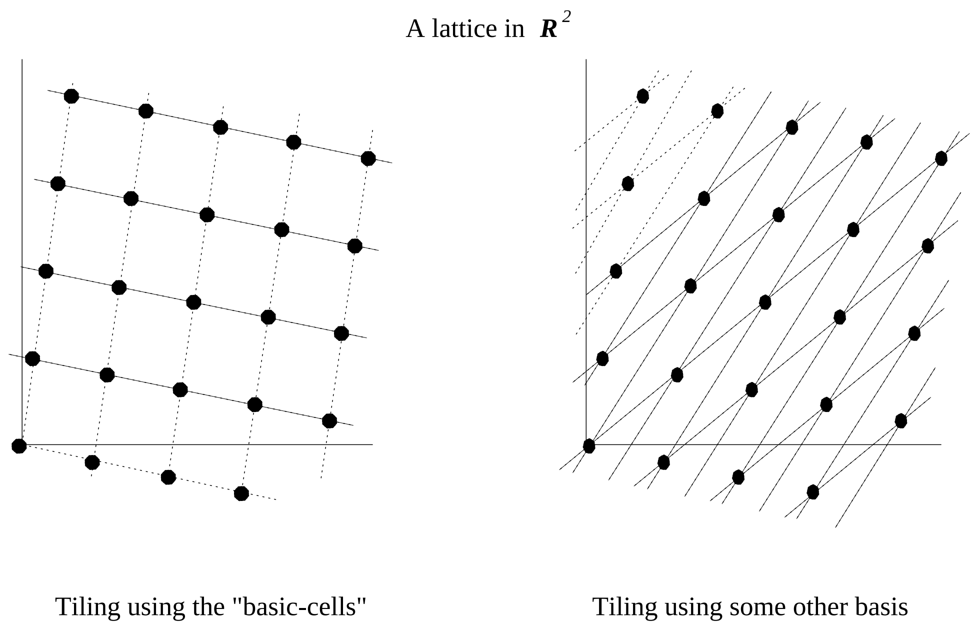

Figure 1: Tiling of a simple lattice in $\mathcal{R}^2$ with two different bases

Finding "short vectors" (i.e., vectors with small Euclidean norm) in lattices is considered a hard problem. There are no known efficient algorithms to find - given an arbitrary basis of a lattice - either the shortest non-zero vector in the lattice, or another basis for the same lattice whose longest vector is as short as possible. No efficient algorithms are known for approximation versions of these problems as well. The approximation versions considered are

(W1) Given an arbitrary basis B of a lattice L in $\mathcal{R}^n$, approximate (up to a polynomial factor in n) the length of the shortest vector in L.

(W2) Given an arbitrary basis B of a lattice L in $\mathcal{R}^n$, find another basis of L whose length is at most polynomially (in n) larger then that of the smallest basis of L (where the length of a basis is the length of its longest vector).

We choose 'W' for the notation to indicate that we will be interested in the worst-case complexity of these problems. The best known algorithms for these problems are the L^3 algorithm and Schnorr algorithm. The L^3 algorithm, due to Lenstra, Lenstra and Lovász [5] approximates these problems to within a ratio of $2^{n/2}$ in the worst case, and Schnorr's algorithm [6] improves this to $(1+\varepsilon)^n$ for any fixed $\varepsilon > 0$. Another problem which can be shown to be equivalent to the above approximation problems (cf., [1]) is the following:

(W3) Given an arbitrary basis B of a lattice L, find a set of n linearly independent lattice vectors, whose length is at most polynomially (in n) larger than the length of the smallest set of n linearly independent lattice vectors. (Again, the length of a set of vectors is the length of its longest vector.)

A few remarks about **(W3)** are in order:

1. Note that not every linearly independent set of n lattice points is a basis for that lattice. For example, if $V = \{v_1, v_2\}$ span some lattice in $\mathcal{R}^2$, then the set $\{2v_1, v_2\}$ is a linearly independent set of 2 vectors which does not span $L(V)$, since we cannot represent v_1 as an integral linear combination of $2v_1$ and v_2.

2. In the sequel we reduce the security of our construction to the difficulty of solving Problem (W3). It will be convenient to use the following notation: For a given polynomial $Q(\cdot)$, denote by $(\text{W3})_Q$ the problem of approximating the smallest independent set in an n-dimensional lattice up to a factor of $Q(n)$.

2.2 Ajtai's Reduction

In his paper Ajtai described the following problem:

Problem (A1):
Parameters: $n, m, q \in \mathcal{N}$, such that $n \log q < m \leq \frac{q}{2n^4}$ and $q = O(n^c)$ for some constant $c > 0$.

Input: A matrix $M \in \mathcal{Z}_q^{n \times m}$.

Output: A vector $x \in \mathcal{Z}_q^m, x \neq 0$ so that $Mx \equiv 0 \pmod{q}$, and $\|x\| < n$ (where $\|x\|$ denotes the Euclidean norm of x).

Here, we used 'A' (in the notation) to indicate that we will be interested in the average-case complexity of this problem. Ajtai proved the following theorem, reducing the worst-case complexity of (W3) to the average-case complexity of (A1):

Ajtai's Theorem [1]: Suppose that it is possible to solve a *uniformly selected* instance of Problem (A1) in expected $T(n, m, q)$-time, where the expectation is taken over the choice of the instance as well as the coin-tosses of the solving algorithm. Then it is possible to solve Problem (W3) in expected $\text{poly}(|I|) \cdot T(n, \text{poly}(n), \text{poly}(n))$ time *on every* n-dimensional instance I, where the expectation is taken over the coin-tosses of the solving algorithm.

Remark: Ajtai has noted that the theorem remain valid also when Problem (A1) is relaxed so that the desired output is allowed to have Euclidean norm of up to $\text{poly}(n)$ (i.e., one requires $\|x\| \leq \text{poly}(n)$ rather than $\|x\| < n$) [1].

2.3 Our Version

We observe that one can use essentially the same proof to prove that the following problem is also hard on the average

Problem (A2):

Parameters: n, m, q as in (A1).

Input: A matrix $M \in \mathcal{Z}_q^{n \times m}$.

Output: A vector $x \in \{-1, 0, 1\}^m, x \neq 0$ so that $Mx \equiv 0 \pmod{q}$.

Theorem 1: Suppose that it is possible to solve a *uniformly selected instance* of Problem (A2) in expected $T(n, m, q)$-time, where the expectation is taken over the choice of the instance as well as the coin-tosses of the solving algorithm. Then it is possible to solve Problem (W3) in expected $\text{poly}(|I|) \cdot T(n, \text{poly}(n), \text{poly}(n))$ time *on every* n-dimensional instance I, where the expectation is taken over the coin-tosses of the solving algorithm.

Proof: By the above Remark, Ajtai's Theorem holds also when modifying Problem (A1) so that the output is (only) required to have Euclidean norm of up to m. Once so modified, Problem (A1) becomes more relaxed than Problem (A2) and so the current theorem follows. $\square$

For the sake of self-containment we sketch the main ideas of the proof of Ajtai's Theorem (equivalently, of Theorem 1) in Section 4. The reader is referred to [1] for further details.

3 Constructing Collision-Free Hash Functions

Recall our construction of a family of collision-free hash functions:

Picking a hash-function

To pick a hash-function with security-parameters n, m, q (where $n \log q < m \leq \frac{q}{2n^4}$ and $q = O(n^c)$), we pick a random matrix $M \in \mathcal{Z}_q^{n \times m}$.

Evaluating the hash function

Given a matrix $M \in \mathcal{Z}_q^{n \times m}$ and a string $s \in \{0, 1\}^m$, compute

$$h_M(s) = Ms \bmod q = \sum_i s_i M_i \bmod q$$

The collision-free property is easy to establish assuming that Problem (A2) is hard on the average. That is,

Theorem 2: Suppose that given a uniformly chosen matrix, $M \in \mathcal{Z}_q^{n \times m}$, as above it is possible to find in (expected) $T(n, m, q)$-time $x \neq y \in \{0, 1\}^m$ so that $Mx \equiv My \pmod{q}$. Then it is possible to solve a *uniformly selected instance* of Problem (A2) in (expected) $T(n, m, q)$-time.

Proof: If we can find two binary strings $s_1 \neq s_2 \in \{0, 1\}^m$ so that $Ms_1 \equiv Ms_2 \pmod{q}$ then we have $M(s_1 - s_2) \equiv 0 \pmod{q}$. Since $s_1, s_2 \in \{0, 1\}^m$, we have $x \stackrel{\text{def}}{=} (s_1 - s_2) \in \{-1, 0, 1\}^m$, which constitutes a solution to Problem (A2) for the instance M. $\square$

3.1 The Parameters

The proof of Theorem 1 imposes restrictions on the relationship between the parameters n, m and q. First of all, we should think of n as the security parameter of the system, since we derive the difficulty of solving Problem (A2) by assuming the difficulty of approximating some problems over n-dimensional lattices.

The relation $m > n \log q$ is necessary for two reasons. The first is simply because we want the output of the hash function to be shorter than its input. The second is that when $m < n \log q$, a random instance of problem (A2) typically does not have a solution at all, and the reduction procedure in the proof of Theorem 1 falls apart.

The relations $q = O(n^c)$ and $m < q/2n^4$ also come from the proof of Theorem 1. Their implications for the security of the system are as follows:

- The larger q is, the stronger the assumption which needs to be made regarding the complexity of problem (W3). Namely, the security proof shows that (A2) with parameters n, m, q is hard to solve on the average, if the problem $(\mathrm{W3})_{(qn^6)}$ is hard in the worst case, where $(\mathrm{W3})_{(qn^6)}$ is the problem of approximating the shortest independent set of a lattice up to a factor of qn^6.

  Thus, for example, if we worry (for a given n) that an approximation ratio of n^{15} is feasible, then we better choose $q < n^9$. Also, since we know that approximation within exponential factor is possible, we must always choose q to be sub-exponential in n.

- By the above, the ratio $R \stackrel{\mathrm{def}}{=} \frac{q/n^4}{m}$ must be strictly bigger than 1 (above, for simplicy, we stated $R > 2$). The larger R is, the better the reduction becomes: In the reduction from (W3) to (A2) we need to solve several random (A2) problems to obtain a solution to one (W3) problem. The number of instances of (A2) problem which need to be solved depends on R. Specifically, this number behaves roughly like $n^2/\log R$ This means that when $q/n^4 = 2m$ we need to solve about n^2 (A2) instances for every (W3) instance, which yields a ratio of $O(n^2)$ between the time it takes to break the hashing scheme and the time it takes to solve a worst-case (W3) problem. On the other hand, when R approaches 1 the number of iterations (in the reduction) grows rapidly (and tends to infinity).

Notice also that the inequalities $n \log q < m < \frac{q}{n^4}$ implies a lower bound on q, namely $\frac{q}{\log q} > n^5$, which means that $q = \Omega(n^5 \log n)$.

4 Self-contained Proof Sketch of Theorem 1

At the heart of the proof is the following procedure for solving (W3): It takes as inputs a basis $B = \langle b_1, \cdots, b_n \rangle$ for a lattice and a set of n linearly independent lattice vectors $V = \langle v_1, \cdots, v_n \rangle$, with $|v_1| \le |v_2| \le \cdots \le |v_n|$. The procedure produces another lattice vector w, such that $|w| \le |v_n|/2$ and w is linearly independent of $v_1, \cdots, v_{n-1}$. We can then replace the vector v_n with w and repeat this process until we get a "very short independent set". When invoking this procedure, we denote by S the length of the vector v_n (which is the longest vector in V).

In the sequel we describe this procedure and show that as long as S is more than n^c times the size of the basic lattice-cell (for some constant $c > 0$), the procedure succeeds with high probability. Therefore we can repeat the process until the procedure fails, and then conclude that (with high probability) the length of the longest vector in V is not more that n^c times the size of the basic lattice-cell. For the rest of this section we will assume that S is larger than n^c times the size of the basic lattice-cell.

The procedure consists of five steps: We first construct an "almost cubic" parallelepiped of lattice vectors, which we call a *pseudo-cube*. Next, we divide this pseudo-cube into many small parallelepipeds (not necessarily of lattice vectors), which we call *sub-pseudo-cubes*. We then pick some random lattice points in the pseudo-cube (cf., Step 3) and consider the location of each point with respect to the partition of the pseudo-cube into sub-pseudo-cubes (cf., Step 4). Each such location is represented as a vector in $\mathcal{Z}_q^n$ and the collection of these vectors forms an instance of Problem (A2). A solution to this instance yields a lattice point which is pretty close to a "corner" of the pseudo-cube. Thus, our final step consists of using the solution to this (A2) instance to compute the "short vector" w. Below we describe each of these steps in more details.

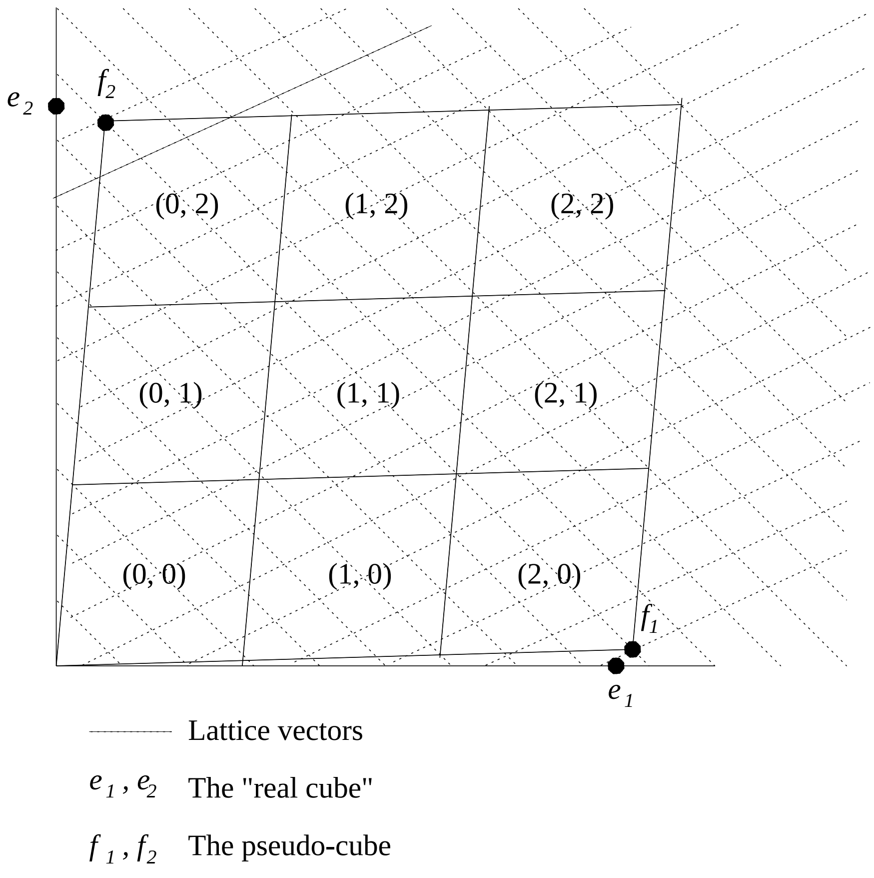

Figure 2: The basic construction in the proof of Theorem 1 (for $q = 3$)

1. Constructing a "pseudo-cube". The procedure first constructs a parallelepiped of lattice vectors which is "almost a cube". This can be done by taking a sufficiently large cube (say, a cube with side length of $n^3 S$), expressing each of the cubes' basis vectors as a linear combination of the v_i's, and then rounding the coefficients in this combination to the nearest integers. Denote the vectors thus obtained by $f_1, \cdots, f_n$ and the parallelepiped which is spanned by them by C. The f_i's are all lattice vectors, and their distance from the basis vectors of the "real cube" is very small compared to the size of the cube[1]. Hence the parallelepiped C is very "cube-like". We call this parallelepiped a *pseudo-cube*.

[1]The f_i's can be as far as $Sn/2$ away from the basis vectors of the real cube, but this is still much smaller than the size of the cube itself.

2. Dividing the pseudo-cube into "sub-pseudo-cubes". We then divide C into q^n equal "sub-pseudo-cubes", each of which can be represented by a vector in $\mathcal{Z}_q^n$ as follows:

$$\text{for } T = \begin{pmatrix} t_1 \\ \vdots \\ t_n \end{pmatrix} \in \mathcal{Z}_q^n, \quad \text{define} \quad C_T \stackrel{\text{def}}{=} \left\{ \sum_i \alpha_i f_i \ : \ \frac{t_i}{q} \le \alpha_i < \frac{t_i+1}{q} \right\}$$

For each sub-pseudo-cube C_T, we call the vector $o_{_T} = \sum_i \frac{t_i}{q} f_i$ *the origin of* C_T. ($o_{_T}$ is the vector in C_T which is closest to the origin). We note that any vector in $v \in C_T$ can be written as $v = o_{_T} + \delta$ where δ is the location of v inside the sub-pseudo-cube C_T. See Figure 2 for an illustration of that construction (with $n = 2, q = 3$).

The parameter q was chosen so that each C_T is "much smaller" than S. That is, the side-length of each sub-pseudo-cube C_T is $Sn^3/q \le S/2nm$. With this choice, each C_T is still much larger than the basic lattice cell (since S is much bigger than the size of the basic cell). This, together with the fact that the C_T's are close to being cubes, implies that each C_T contains approximately the same number of lattice points.

3. Choosing random lattice points in C. We then choose m random lattice points $u_1, \cdots u_m \in C$. To do that, we use the basis $B = \{b_1, \cdots, b_n\}$ of the lattice. To choose each point, we take a linear combination of the basis vectors b_i with large enough integer coefficients (say, in the range $[0, 2^{n^c} \cdot \max(S, |B|)]$ for some constant c). This gives us some lattice point p.

We then "reduce p mod C". By this we mean that we look at a tiling of the space $\mathcal{R}^n$ with the pseudo-cube C, and we compute the location vector of p in its surrounding pseudo-cube. Formally, this is done by representing p as a linear combination of the f_i's, and taking the fractional part of the coefficients in this combination. The resulting vector is a lattice point, since it is obtained by subtracting integer combination of the f_i's from p. Also, this vector must lie inside C, since it is a linear combination of the f_i's with coefficients in $[0, 1)$. It can be shown that if we choose the coefficients from a large enough range, then the distribution induced over the lattice point in C is statistically close to the uniform distribution.

4. Constructing an instance of Problem (A2). After we have chosen m lattice points $u_1, \cdots, u_m$, we compute for each u_i the vector $T_i \in \mathcal{Z}_q^n$ which represent the sub-pseudo-cube in which u_i falls. That is, for each i we have $u_i \in C_{T_i}$.

Since, as we said above, each sub-pseudo-cube contains approximately the same number of points, and since the u_i's are distributed almost uniformly in C, then the distribution induced on the C_{T_i}'s is close to the uniform distribution, and so the distribution over the T_i's is close to the uniform distribution over $\mathcal{Z}_q^n$.

We now consider the matrix whose columns are the vectors T_i, $M = (T_1|T_2|\cdots|T_m)$. By the above argument, it is an "almost uniform" random matrix in $\mathcal{Z}_q^{n\times m}$, and so, it is an "almost uniform" random instance of Problem (A2).

5. Computing a "short lattice vector". We now have a random instance M of Problem (A2), and so we can use the algorithm whose existence we assume in Theorem 1 to solve it in expected $T(n, m, q)$ time. The solution is a vector $x = \{x_1, \cdots, x_m\} \in \{-1, 0, 1\}^m$ so that $Mx = \sum_i x_i T_i = 0 \pmod q$.

Once we found x, we compute the vector $w' = \sum_{i=1}^m x_i u_i$. Let us examine the vector w': Recall that we can represent each u_i as the sum of $o_i \stackrel{\text{def}}{=} o_{_{T_i}}$ (the origin vector of C_{T_i}) and δ_i (the location of u_i inside C_{T_i}). Thus,

$$w' = \sum_{i=1}^{m} x_i u_i = \sum_{i=1}^{m} x_i o_i + \sum_{i=1}^{m} x_i \delta_i$$

A key observation is that since $\sum_i x_i T_i = \bar{0} \pmod q$, "reducing the vector $(\sum_i x_i o_i)$ mod C" we get $(\sum_i x_i o_i) \pmod C = \bar{0}$. To see why this is the case, recall that each $o_i = o_{T_i}$ has the form $\sum_j \frac{t_i(j)}{q} f_j$, where $t_i(j) \in \{0, ..., q-1\}$ is the j^{th} component of T_i. The hypothesis $\sum_i x_i t_i(j) = 0 \pmod q$ for $j = 1, .., n$, yields that

$$\sum_i x_i o_{T_i} \;=\; \sum_i x_i \sum_j \frac{t_i(j)}{q} f_j \;=\; \sum_j (\frac{\sum_i x_i t_i(j)}{q}) f_j \;=\; \sum_j c_j f_j$$

where all c_j's are integers. Since "reducing the vector $\sum_i x_i o_{T_i}$ mod C" means subtracting from it an integer linear combination of f_j's, the resulting vector is $\bar{0}$. Thus, "reducing w' mod C" we get

$$w' \pmod C = \sum_{i=1}^{m} x_i \delta_i$$

Since each δ_i is just the location of some point inside the sub-pseudo-cube C_{T_i} then the size of each δ_i is at most $n \cdot S/2mn = S/2m$. Moreover as $x_i \in \{-1, 0, 1\}$ for all i we get

$$\left\| \sum_i x_i \delta_i \right\| \leq \sum_i |x_i| \cdot \|\delta_i\| \leq m \cdot \frac{S}{2m} = \frac{S}{2}$$

This means that the vector $w' \pmod C$ is close up to $\frac{S}{2}$ to one of the "corners" of C. Thus all we need to do is to find the difference vector between $w' \pmod C$ and that corner. Doing that is very similar to reducing w' mod C: We express w' as a linear combination of the f_i's, but instead of taking the fractional part of the coefficients we take the difference between these coefficients and the closest integers. This gives us the "promised vector" w, a lattice point whose length is at most $S/2$.

The only thing left to verify is that with high probability, w can replace the largest vector in V (i.e., it is linearly independent of the other vectors in V). To see that, notice that the vector x does not depend on the exact choice of the u_i's, but only on the choice of their sub-pseudo-cubes C_{T_i}'s. Thus we can think of the process of choosing the u_i's as first choosing the C_{T_i}'s, computing the x_i's and only then choosing the δ_i's.

Assume (w.l.o.g.) that we have $x_1 \neq 0$. Let us now fix all the δ_i's except δ_1 and then pick δ_1 so as to get a random lattice point in C_{T_1}. Thus, the probability that w falls in some fixed subspace of $\mathcal{R}^n$ (such as the one spanned by the $n-1$ smallest vectors in V), equals the probability that a random point in C_{T_1} falls in such subspace. Since C_{T_1} is a pseudo-cube which is much larger than the basic cell of L, this probability is very small.

Acknowledgments

We thank Dan Boneh and Jin Yi Cai for drawing our attention to an error in a previous version of this note.

References

[1] M. Ajtai. Generating Hard Instances of Lattice Problems. In *28th ACM Symposium on Theory of Computing*, pages 99–108, Philadelphia, 1996.

[2] L. Carter and M. Wegman. Universal Classes of Hash Functions. *J. Computer and System Sciences*, Vol. 18, pages 143–154 (1979).

[3] O. Goldreich, H. Krawczyk and M. Luby. On the existence of pseudorandom generators. *SIAM J. on Computing*, Vol. 22-6, pages 1163–1175 (1993).

[4] J. Hastad, R. Impagliazzo, L.A. Levin and M. Luby. Construction of Pseudorandom Generator from any One-Way Function. To appear in *SIAM J. on Computing*. Preliminary versions by Impagliazzo et. al. in *21st STOC* (1989) and Hastad in *22nd STOC* (1990).

[5] A.K. Lenstra, H.W. Lenstra, L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen* 261, pages 515–534 (1982).

[6] C.P. Schnorr. A more efficient algorithm for a lattice basis reduction. *Journal of Algorithms*, Vol. 9, pages 47–62 (1988).