The n^c-Unique Shortest Vector Problem is Hard

Vadim Lyubashevsky*

Abstract

The unique Shortest Vector Problem (uSVP) gained prominence because it was the problem upon which the first provably-secure lattice-based cryptosystems were built. But it was an open problem as to whether uSVP was as hard as the standard, more general, version of the shortest vector problem.

We show that there is a reduction from the approximate decision version of the shortest vector problem (GapSVP) to the unique shortest vector problem. In particular, we show that for any $\gamma > 6\sqrt{n}$, there is a reduction from GapSVP$_\gamma$ to $\frac{\gamma}{6\sqrt{n}}$-uSVP. This implies that the Ajtai-Dwork and the Regev cryptosystems are based on the hardness of the worst-case GapSVP$_{O(n^{2.5})}$ and GapSVP$_{O(n^2)}$, respectively. Our reduction is quite elementary, but it does use a clever, yet surprisingly simple (in retrospect!), idea of Peikert that was recently used by him to construct a cryptosystem based on the worst-case hardness of GapSVP$_{O(n^3)}$.

1 Introduction

Building cryptographic primitives based on the hardness of worst-case lattice problems began with the seminal work of Ajtai [Ajt96]. One of the first primitives that was built was a cryptosystem [AD97] with security based on the worst-case hardness of the unique Shortest Vector Problem (uSVP). The security of the system was based on the hardness of solving the $O(n^8)$-uSVP problem (in the γ-uSVP problem, we are asked to find the shortest vector in a lattice in which the shortest vector is at least γ times smaller than the next non-parallel lattice vector). This was followed by an improvement to their cryptosystem [GGH97], and the currently best version of it is based on the hardness of the $O(n^2)$-uSVP. Later, Regev built a different cryptosystem based on worst-case $O(n^{1.5})$-uSVP problem [Reg04]. But while other cryptographic primitives could be built on the hardness of the more general, and more well understood, decision version of the shortest vector problem (GapSVP), cryptosystems seemed to require the hardness of the potentially easier uSVP.

A breakthrough, in the sense of deviating from uSVP, came when Regev built a cryptosystem which was actually based on GapSVP (as well as some other standard lattice problems), but the assumption was that approximating GapSVP was hard even by quantum algorithms [Reg05]. And another breakthrough came just recently, when Peikert finally constructed a cryptosystem that is based on the hardness of the shortest vector problem under classical reductions [Pei08].

Of course, a different way of proving the existence of cryptosystems with security based on GapSVP is to show a reduction from GapSVP to uSVP, which is the approach taken here.

*Tel-Aviv University. E-mail: vlyubash@cs.ucsd.edu

1.1 Results and Related Work

Our main result is a randomized worst-case to worst-case reduction from approximating GapSVP within a factor γ (GapSVP$_\gamma$) to solving the $\frac{\gamma}{6\sqrt{n}}$-uSVP problem. We mention that the parameters are not optimized, and with just a simple modification (which would make things a little messier) it is possible to show a reduction to $\gamma \cdot \sqrt{\frac{\log n}{cn}}$-uSVP for some constant c.

Since our reduction is inspired by the recent work of Peikert [Pei08], it is worthwhile to first review the related portion of his result. At the heart of Peikert's cryptosystem construction, in our opinion, is a clever idea for using a seemingly "useless" oracle. The oracle that he uses is one that finds the closest lattice vector if the target point is already very close to the lattice (this is the Bounded Distance Decoding problem). At first glance, it is not clear how one might use such an oracle (although Regev found a quantum use for it [Reg05]) because it seems that one needs to know a close lattice vector if he is to successfully generate a target that is close to the lattice. And in that case, the oracle doesn't seem useful because it'll just return a vector that you already know. But Peikert shows how to use such an oracle to solve the GapSVP$_\gamma$ problem. Given a GapSVP$_\gamma$ instance, the idea is to randomly create a target point $\mathbf{t}$ around a lattice point $\mathbf{v}$, and then query the oracle. If the oracle returns back the lattice point $\mathbf{v}$, then the shortest vector is probably large. If, on the other hand, the oracle returns some other lattice point, then the shortest vector is definitely small. It can be shown that if the shortest vector is small, then the oracle cannot always know the lattice point $\mathbf{v}$, and therefore must return something else (or not return anything) which will give away the fact that the length of the shortest vector is small.

In our reduction, given a GapSVP$_\gamma$ instance $(\mathbf{B}, d)$, we create $n/2$ other lattices such that if there is no short vector in $\mathbf{B}$, then at least one of the instances will be a valid instance of uSVP (although we will not know which one). The idea is that caling the uSVP oracle should not produce any "unexpected" short vectors of a certain form if the shortest vector of $\mathbf{B}$ is large, and it must produce an "expected" short vector on the instances of uSVP that are valid. On the other hand, if the shortest vector of $\mathbf{B}$ is small, then the uSVP oracle must either not produce any vectors of a certain form, or produce an unexpected vector of the type that we are looking for. We give all the details in Section 3.

The immediate implication of our result is that the Ajtai-Dwork [AD97] and the Regev cryptosystems [Reg04] are based on the hardness of GapSVP$_{O(n^{2.5})}$ and GapSVP$_{O(n^2)}$ respectively. At this point, this compares favorably to Peikert's recent cryptosystem which is based on the hardness of the GapSVP$_{O(n^3)}$ problem. But it's quite possible that Peikert's cryptosystem can be improved to also be based on GapSVP$_{O(n^2)}$ with a better trap-door generating algorithm (see [Pei08] for details). Additionally, Peikert shows how to convert his semantically-secure cryptosystem to a CCA-secure one (with a factor-n loss in the reduction), but it is unclear whether the same can be done for the Ajtai-Dwork and the Regev cryptosystems.

2 Preliminaries

An n-dimensional *lattice* is an additive subgroup of $\mathbb{R}^n$. The set of vectors that generate a lattice is called a basis, and we will denote it as an $n \times n$ matrix $\mathbf{B}$ whose columns $\mathbf{b}_i$ are the generator vectors. The lattice generated by the basis $\mathbf{B}$ will be written as $\mathcal{L}(\mathbf{B})$. The *fundamental parallelepiped* of an $n \times n$ basis $\mathbf{B}$, written as $\mathcal{P}(\mathbf{B})$, is defined as the collection of all points that can be written as $\mathbf{By}$ where $\mathbf{y} \in [0,1)^n$. Every point $\mathbf{s} \in \mathbb{R}^n$ has a unique associated point $\mathbf{t}$ inside $\mathcal{P}(\mathbf{B})$ such that $\mathbf{s} = \mathbf{t}$

in the quotient group $\mathbb{R}^n/\mathcal{L}(\mathbf{B})$. Given an $\mathbf{s}$, it is possible to compute $\mathbf{t} = \mathbf{s} \bmod \mathbf{B}$ in polynomial time.

The *shortest vector* of a lattice $\mathcal{L}(\mathbf{B})$ is the non-zero vector in $\mathcal{L}(\mathbf{B})$ with the smallest ℓ_2 norm. The length of the shortest vector of $\mathcal{L}(\mathbf{B})$ is denoted by $\lambda_1(\mathcal{L}(\mathbf{B}))$ (or $\lambda_1(\mathbf{B})$ for short). Possibly the most well-known lattice problem is the Shortest Vector Problem (SVP). It comes in both the decisional and search versions, but in this paper we are only interested in the decision version. The approximation version of decisional SVP can be defined as a "gap" problem GapSVP$_\gamma$. In the GapSVP$_\gamma$ problem, we are given a basis $\mathbf{B}$ and a real number d, and are required to return YES if $\lambda_1(\mathcal{L}(\mathbf{B})) \leq d$, and return NO if $\lambda_1(\mathcal{L}(\mathbf{B})) > \gamma d$. If $\lambda_1(\mathcal{L}(\mathbf{B}))$ falls between d and γd, we can return anything. The GapSVP$_\gamma$ problem is NP-hard for any constant γ [Kho04, HR07]. The fastest algorithm for solving GapSVP$_\gamma$ for $1 \leq \gamma \leq poly(n)$ takes time $2^{O(n)}$[AKS01]. Using the LLL algorithm [LLL82], it is possible to find a vector that has length at most $2^{n/2}\lambda_1(\mathbf{B})$ in polynomial time.

A γ-unique shortest vector in a lattice $\mathcal{L}(\mathbf{B})$ is a vector whose length is $\lambda_1(\mathbf{B})$, and there are no other vectors in $\mathcal{L}(\mathbf{B})$ of length between $\lambda_1(\mathbf{B})$ and $\gamma\lambda_1(\mathbf{B})$ that are not simply multiples of this vector. The γ-unique Shortest Vector Problem (γ-uSVP) is a search problem in which we are given a lattice basis $\mathbf{B}$ with the promise that there exists a γ-unique shortest vector, and we are asked to find it. This problem is known to be NP-hard (under randomized reductions) in its exact version [KS01], but very little is known about the hardness of this problem for larger factors γ.

One additional fact that we will require is the following lemma of Goldreich and Goldwasser [GG00] which states that the intersection of two spheres of large radii that are relatively close together is large.

Lemma 2.1 ([GG00]). *Let $\mathbf{x}$ be a vector in $\mathbb{R}^n$ such that $\|\mathbf{x}\| \leq d$. If $\mathbf{s}$ is a point chosen at random from a ball of radius $d\sqrt{n}$ centered at $\mathbf{0}$, then with some constant probability $\delta > 0$, $\|\mathbf{s} - \mathbf{x}\| \leq d\sqrt{n}$.*

3 The Reduction

We will now describe the reduction from GapSVP to uSVP which is given in Figure 1. The input to our algorithm is an LLL-reduced lattice basis $\mathbf{B}$ and a number d. We need to return YES if $\lambda_1(\mathbf{B}) \leq d$ and NO if $\lambda_1(\mathbf{B}) > \gamma d$. In other cases, any reply will suffice. The algorithm runs for n iterations, and if during none of these iterations it outputs YES, then we output NO. Every iteration starts by generating a point within a ball of radius $d\sqrt{n}$ around the origin and reducing it modulo $\mathbf{B}$ (we call the resulting point $\mathbf{t}$). The idea is then to set up an instance of uSVP such that if $\mathbf{t}$ is close to the lattice, then the uSVP instance will have a $\frac{\gamma}{6\sqrt{n}}$-unique shortest vector that our uSVP oracle can find. The instance that we set up is $\mathbf{C} = \begin{bmatrix} \mathbf{B} & \mathbf{t} \\ \mathbf{0} & \alpha \end{bmatrix}$ where the parameter α needs to depend on the length of the shortest vector of $\mathcal{L}(\mathbf{B})$ (as in Lemma 3.2). Of course we do not know $\lambda_1(\mathbf{B})$, which is why we will need to iterate through $n/2$ values of α to guarantee that one of them will produce an instance of uSVP. It is also crucial that we use the same $\mathbf{t}$ for all the $n/2$ instances where we vary the value of α.

In step 9 of the algorithm we call the uSVP oracle on our instance $\mathbf{C}$ and the oracle returns some $\mathbf{w}'$, which we write as

$$\mathbf{w}' = \begin{bmatrix} \mathbf{v} - \mathbf{t}\beta \\ -\alpha\beta \end{bmatrix}$$

GapSVP$_\gamma(\mathbf{B}, d)$
Input: LLL-reduced lattice basis $\mathbf{B}$ such that $\|\mathbf{b}_1\| \le 2^{n/2}\lambda_1(\mathbf{B})$, and a positive real d.
Output: YES if $\lambda_1(\mathbf{B}) \le d$ and NO if $\lambda_1(\mathbf{B}) > \gamma d$.

for $j = 1$ to n **do**
  $\mathbf{s} \leftarrow$ Generate a uniformly random point within the ball of radius $d\sqrt{n}$ centered at $\mathbf{0}$.
  $\mathbf{t} \leftarrow \mathbf{s} \bmod \mathbf{B}$
  $\alpha \leftarrow \frac{\sqrt{n}\|\mathbf{b}_1\|}{\gamma 2^{n/2}}$
  $betaWasOne \leftarrow false$
  for $i = 0$ to $n/2 - 1$ **do**
    $\alpha \leftarrow \alpha \cdot 2^i$
    $\mathbf{C} \leftarrow \begin{bmatrix} \mathbf{B} & \mathbf{t} \\ \mathbf{0} & \alpha \end{bmatrix}$
    Call the $\frac{\gamma}{6\sqrt{n}}$-uSVP oracle on $\mathbf{C}$, and let $\mathbf{w}'$ be its response.
    Write $\mathbf{w}' = \begin{bmatrix} \mathbf{v} - \mathbf{t}\beta \\ -\alpha\beta \end{bmatrix}$ where $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ and $\beta \in \mathbb{Z}$ (w.l.o.g. assume that $\beta \ge 0$)
    if $\beta = 1$ and $\|\mathbf{v} - \mathbf{t}\| \le d\sqrt{n}$ and $\mathbf{v} \ne \mathbf{t} - \mathbf{s}$ **then**
      output YES and end the algorithm
    end if
    if $\beta = 1$ and $\|\mathbf{v} - \mathbf{t}\| \le d\sqrt{n}$ and $\mathbf{v} = \mathbf{t} - \mathbf{s}$ **then**
      $betaWasOne \leftarrow true$
    end if
  end for
  if $betaWasOne = false$ **then**
    ouptut YES and end the algorithm
  end if
end for
output NO

Figure 1: Solving GapSVP$_\gamma$ when given an oracle for $\frac{\gamma}{6\sqrt{n}}$-uSVP.

where $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ and $\beta \in \mathbb{Z}$. The reason we can do this is that we can solve the equation $\mathbf{C}\mathbf{x}' = \mathbf{w}'$ for $\mathbf{x}'$, and multiplying $\mathbf{B}$ by the first n coordinates of $\mathbf{x}'$ gives us a vector in $\mathcal{L}(\mathbf{B})$, while the last coordinate of $\mathbf{x}'$ gives us the coefficient β for the last column of $\mathbf{C}$. Without loss of generality, we can assume that $\beta \ge 0$, by simply multiplying the vector $\mathbf{x}'$ by -1 if necessary. It can be shown that if $\mathbf{t}$ is close to the lattice, then at least one of the $n/2$ lattices $\mathcal{L}(\mathbf{C})$ will be a valid uSVP instance in which the oracle must return a vector in the lattice that uses the last column only once. If this happens, we check to make sure that the vector returned by the oracle is the vector that we "expected". If it is, then we just go on with the algorithm, otherwise, we output YES signifying that $\lambda_1(\mathbf{B}) \le d$. We also need to make sure that in the case that $\mathbf{t}$ is actually close to $\mathcal{L}(\mathbf{B})$, the uSVP oracle does return an expected vector at least once. This is what the boolean variable $betaWasOne$ is for. If at the end of the $n/2$ iterations it is still set to false, we know that none of the lattices $\mathcal{L}(\mathbf{C})$ were valid uSVP instances, and we conclude that $\lambda_1(\mathbf{B}) \le d$.

If $\lambda_1(\mathbf{B}) \le d$, then it can be shown that for each choice of $\mathbf{t}$, the uSVP oracle will either never work for all the $n/2$ iterations of α, or it will return an unexpected short vector with some constant probability. Therefore repeating the algorithm n times with different values of $\mathbf{t}$ will identify the YES-instances of GapSVP with probability negligibly close to one. If the algorithm never answers YES, then we can be almost certain that we have a NO-instance of GapSVP.

We now give a formal proof of our main result.

Theorem 3.1. *For any $\gamma > 6\sqrt{n}$, there is a randomized polynomial-time Cook-reduction from GapSVP$_\gamma$ to $\frac{\gamma}{6\sqrt{n}}$-uSVP.*

Proof. Suppose that $(\mathbf{B}, d)$ is a NO-instance of GapSVP$_\gamma$. This means that $\lambda_1(\mathbf{B}) > \gamma d$, and therefore the point $\mathbf{t}$ that is generated by the algorithm is a distance of at most $d\sqrt{n} < \frac{\lambda_1(\mathbf{B})\sqrt{n}}{\gamma}$ away from the lattice. In this case we need to show that the algorithm never outputs YES. There are two places where the algorithm will reply YES. The first one of these is in line 11, if $\beta = 1$, $\|\mathbf{v} - \mathbf{t}\| \leq d\sqrt{n}$ and $\mathbf{v} \neq \mathbf{t} - \mathbf{s}$. But this cannot happen because $\mathbf{t} - \mathbf{s}$ is a vector in $\mathcal{L}(\mathbf{B})$ and

$$\|\mathbf{v} - (\mathbf{t} - \mathbf{s})\| \leq \|\mathbf{v} - \mathbf{t}\| + \|\mathbf{s}\| \leq 2d\sqrt{n} < \frac{2\lambda_1(\mathbf{B})\sqrt{n}}{\gamma} < \frac{2\lambda_1(\mathbf{B})\sqrt{n}}{6\sqrt{n}} = \frac{\lambda_1(\mathbf{B})}{3},$$

which is a contradiction.

The second case is that the boolean variable *betaWasOne* never gets set to true. We will now show that this can never happen either. Because $dist(\mathbf{t}, \mathcal{L}(\mathbf{B})) \leq \frac{\lambda_1(\mathbf{B})\sqrt{n}}{\gamma}$, Lemma 3.2 tells us that if α falls in between $\frac{\lambda_1(\mathbf{B})\sqrt{n}}{2\gamma}$ and $\frac{\lambda_1(\mathbf{B})\sqrt{n}}{\gamma}$, then the vector $\begin{bmatrix} \mathbf{v} - \mathbf{t} \\ -\alpha \end{bmatrix}$ is the $\frac{\gamma}{6\sqrt{n}}$-unique shortest vector of $\mathcal{L}(\mathbf{C})$. Notice that α starts at

$$\alpha = \frac{\sqrt{n}\|\mathbf{b}_1\|}{\gamma \cdot 2^{n/2}} \leq \frac{\sqrt{n}2^{n/2}\lambda_1(\mathbf{B})}{\gamma 2^{n/2}} = \frac{\sqrt{n}\lambda_1(\mathbf{B})}{\gamma}$$

and ends at

$$\alpha = \frac{\sqrt{n}\|\mathbf{b}_1\|}{\gamma \cdot 2^{n/2}} \cdot 2^{n/2-1} = \frac{\sqrt{n}\|\mathbf{b}_1\|}{2\gamma} \geq \frac{\sqrt{n}\lambda_1(\mathbf{B})}{2\gamma}$$

And because we always multiply α by 2, some value of the initial α multiplied by 2^i must fall in the appropriate range whose endpoints are a factor of 2 apart. Thus when α falls in this range, the $\frac{\gamma}{6\sqrt{n}}$-uSVP oracle will return the vector $\begin{bmatrix} \mathbf{v} - \mathbf{t}\beta \\ -\alpha\beta \end{bmatrix}$ where $\beta = 1$ and $\|\mathbf{v} - \mathbf{t}\| \leq d\sqrt{n}$. And as we showed above, this implies that $\mathbf{v} = \mathbf{t} - \mathbf{s}$, and therefore the boolean variable *betaWasOne* will be set to true.

Now we suppose that $(\mathbf{B}, d)$ is a YES-instance of GapSVP$_\gamma$, which means that $\lambda_1(\mathbf{B}) \leq d$. In this case, it's possible that none of the constructions of the matrix $\mathbf{C}$ result in valid instances of the uSVP problem and so the oracle may behave in any way that it wants. Nevertheless, we will show that even an adversarial uSVP oracle cannot prevent our algorithm from outputting YES with extremely high probability. Notice that if the oracle is to have any chance of preventing the algorithm from outputting YES, he must set the boolean *betaWasOne* to true for every choice of $\mathbf{t}$ by outputting a $\mathbf{w}'$ in which the $\mathbf{v} - \mathbf{t}\beta$ component has $\beta = 1$ and $\mathbf{v} = \mathbf{t} - \mathbf{s}$. Note that finding such a $\mathbf{v}$ implies knowing which $\mathbf{s}$ was selected by the algorithm in step 2, and we will now show that he cannot always guess the $\mathbf{s}$. Let $\mathbf{x}$ be a vector in the lattice such that $\|\mathbf{x}\| \leq d$. Notice that every time we pick an $\mathbf{s}$ and reveal $\mathbf{t} = \mathbf{s} \bmod \mathbf{B}$ to the oracle, by Lemma 2.1, there is some constant probability δ that $\|\mathbf{s} - \mathbf{x}\| \leq d\sqrt{n}$, and therefore in this case, given $\mathbf{t}$, the oracle cannot know with probability greater than $\frac{1}{2}$ whether we randomly generated $\mathbf{s}$ or $\mathbf{s}-\mathbf{x}$ (since both $\mathbf{s} \bmod \mathbf{B}$ and $\mathbf{s}-\mathbf{x} \bmod \mathbf{B}$ equal to $\mathbf{t}$). Therefore for a δ fraction of the $\mathbf{t}$'s that we give him, the oracle cannot guess the exact $\mathbf{s}$ with probability greater than 1/2. And so guessing the $\mathbf{s}$ in all n rounds has success probability of at most $(1-\delta/2)^n$, which is negligible for constant δ. □

Lemma 3.2. *Let* $\mathbf{C}$ *be an* $(n+1)\times(n+1)$ *matrix*

$$\mathbf{C} = \begin{bmatrix} \mathbf{B} & \mathbf{t} \\ \mathbf{0} & \alpha \end{bmatrix}$$

where $\mathbf{B}$ *is an* $n \times n$ *matrix,* $\mathbf{t}$ *is an* $n \times 1$ *vector,* $\mathbf{0}$ *is a* $1 \times n$ *vector of zeros and* α *is a positive real number. For any* $\gamma > 6\sqrt{n}$*, if*

$$\frac{\lambda_1(\mathbf{B})\sqrt{n}}{2\gamma} \leq \alpha \leq \frac{\lambda_1(\mathbf{B})\sqrt{n}}{\gamma} \text{ and } dist(\mathbf{t}, \mathcal{L}(\mathbf{B})) \leq \frac{\lambda_1(\mathbf{B})\sqrt{n}}{\gamma},$$

then $\mathcal{L}(\mathbf{C})$ *has a* $\frac{\gamma}{6\sqrt{n}}$*-unique shortest vector. In particular, if* $\mathbf{v}$ *is the vector in* $\mathcal{L}(\mathbf{B})$ *for which* $dist(\mathbf{t},\mathbf{v}) = dist(\mathbf{t},\mathcal{L}(\mathbf{B}))$*, then the vector* $\begin{bmatrix} \mathbf{v}-\mathbf{t} \\ -\alpha \end{bmatrix}$ *is the* $\frac{\gamma}{6\sqrt{n}}$*-unique shortest vector of* $\mathcal{L}(\mathbf{C})$*.*

Proof. We first show that $\lambda_1(\mathbf{C}) \leq \frac{1.5\cdot\lambda_1(\mathbf{B})\sqrt{n}}{\gamma}$. Let $\mathbf{v}$ be the vector in $\mathcal{L}(\mathbf{B})$ such that $\|\mathbf{t}-\mathbf{v}\| \leq \frac{\lambda_1(\mathbf{B})\sqrt{n}}{\gamma}$, then the vector $\mathbf{v}' = (\mathbf{v}^T, 0)^T - (\mathbf{t}^T, \alpha)^T$ is a vector in $\mathcal{L}(\mathbf{C})$ and

$$\|(\mathbf{v}^T,0)^T - (\mathbf{t}^T,\alpha)^T\| \leq \sqrt{\left(\frac{\lambda_1(\mathbf{B})\sqrt{n}}{\gamma}\right)^2 + \alpha^2} \leq \sqrt{\left(\frac{\lambda_1(\mathbf{B})\sqrt{n}}{\gamma}\right)^2 + \left(\frac{\lambda_1(\mathbf{B})\sqrt{n}}{\gamma}\right)^2} < \frac{1.5\cdot\lambda_1(\mathbf{B})\sqrt{n}}{\gamma}.$$

We now want to show that every other vector in $\mathcal{L}(\mathbf{C})$ that is not a multiple of $\mathbf{v}'$ has length at least

$$\lambda_1(\mathbf{B})/4 \geq \frac{\gamma}{6\sqrt{n}} \cdot \lambda_1(\mathbf{C}).$$

Assume for the sake of contradiction that $\mathbf{w}'$ is a vector of length less than $\lambda_1(\mathbf{B})/4$ that is not a multiple of $\mathbf{v}'$. Notice that if in obtaining $\mathbf{w}'$, we used the last column of $\mathbf{C}$ 0 times, then the shortest vector we could possibly get has length $\lambda_1(\mathbf{B})$. If we used the last column of $\mathbf{C}$ at least $\frac{\gamma}{2\sqrt{n}}$ times, then we get a vector of length at least $\alpha \cdot \frac{\gamma}{2\sqrt{n}} \geq \frac{\lambda_1(\mathbf{B})\sqrt{n}}{2\gamma} \cdot \frac{\gamma}{2\sqrt{n}} \geq \lambda_1(\mathbf{B})/4$. Therefore we must have used the last column of $\mathbf{C}$ some number β times where $0 < |\beta| < \frac{\gamma}{2\sqrt{n}}$. We can rewrite the vector $\mathbf{w}' = ((\mathbf{w}-\beta\mathbf{t})^T, -\beta\alpha)^T$ where $\mathbf{w}$ is some vector in $\mathcal{L}(\mathbf{B})$. Since $\|\mathbf{w}'\| \leq \lambda_1(\mathbf{B})/4$, then certainly $\|\mathbf{w}-\beta\mathbf{t}\| \leq \lambda_1(\mathbf{B})/4$ as well. Now recall that there is a vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ such that $\|\mathbf{v}-\mathbf{t}\| \leq \frac{\lambda_1(\mathbf{B})\sqrt{n}}{\gamma}$. Therefore

$$\begin{aligned}
\|\mathbf{w}-\beta\mathbf{v}\| &= \|(\mathbf{w}-\beta\mathbf{t}) - (\beta\mathbf{v}-\beta\mathbf{t})\| \\
&\leq \|\mathbf{w}-\beta\mathbf{t}\| + |\beta|\cdot\|\mathbf{v}-\mathbf{t}\| \\
&\leq \frac{\lambda_1(\mathbf{B})}{4} + |\beta|\cdot\frac{\lambda_1(\mathbf{B})\sqrt{n}}{\gamma} \\
&\leq \frac{\lambda_1(\mathbf{B})}{4} + \frac{\gamma}{2\sqrt{n}}\cdot\frac{\lambda_1(\mathbf{B})\sqrt{n}}{\gamma} \\
&\leq \frac{\lambda_1(\mathbf{B})}{4} + \frac{\lambda_1(\mathbf{B})}{2} \\
&< \lambda_1(\mathbf{B}).
\end{aligned}$$

Also because we assumed that $\mathbf{w}'$ is not a multiple of $\mathbf{v}'$, it means that $\mathbf{w} \neq \beta\mathbf{v}$, and therefore the vector $\mathbf{w} - \beta\mathbf{v}$ is a non-zero vector in $\mathcal{L}(\mathbf{B})$ whose length is less than $\lambda_1(\mathbf{B})$, which is a contradiction. $\square$

4 Acknowledgements

I would like to thank Chris Peikert for useful comments.

References

[AD97] M. Ajtai and C. Dwork, *A public-key cryptosystem with worst-case/average-case equivalence*, STOC, 1997, An improved version is described in ECCC 2007.

[Ajt96] M. Ajtai, *Generating hard instances of lattice problems*, STOC, 1996, pp. 99–108.

[AKS01] M. Ajtai, R. Kumar, and D. Sivakumar, *A sieve algorithm for the shortest lattice vector problem*, STOC, 2001, pp. 601–610.

[GG00] O. Goldreich and S. Goldwasser, *On the limits of nonapproximability of lattice problems*, J. Comput. Syst. Sci. **60** (2000), no. 3.

[GGH97] O. Goldreich, S. Goldwasser, and S. Halevi, *Eliminating decryption errors in the Ajtai-Dwork cryptosystem*, CRYPTO, 1997, pp. 105–111.

[HR07] I. Haviv and O. Regev, *Tensor-based hardness of the shortest vector problem to within almost polynomial factors*, STOC, 2007, pp. 469–477.

[Kho04] S. Khot, *Hardness of approximating the shortest vector problem in lattices*, FOCS, 2004, pp. 126–135.

[KS01] R. Kumar and D. Sivakumar, *On the unique shortest lattice vector problem*, Theor. Comput. Sci. **255** (2001), no. 1-2, 641–648.

[LLL82] A. K. Lenstra, H. W. Lenstra Jr., and L. Lovasz, *Factoring polynomials with rational coefficients*, Mathematische Annalen (1982), no. 261, 513–534.

[Pei08] C. Peikert, *Public-key cryptosystems from the worst-case shortest vector problem*, Tech. report, ECCC, 2008.

[Reg04] O. Regev, *New lattice-based cryptographic constructions*, J. ACM **51** (2004), no. 6, 899–942.

[Reg05] ———, *On lattices, learning with errors, random linear codes, and cryptography*, STOC, 2005.