CS 4501-6501 Topics in Cryptography February 4, 2015

Lecture 7

Lecturer: Mohammad Mahmoody *Scribe: Matt Irvine*

1 Learning With Errors: Decision vs. Search

In the previous class we discussed the Learning With Errors (LWE) problem where we are given a tuple $(\vec{a}, \langle \vec{a}, \vec{s} \rangle + e)$ where $\vec{a}$ is a vector chosen uniformly at random from $\mathbb{Z}_p^*$, $\vec{s}$ is a vector chosen from $\mathbb{Z}_p^*$, and e is the error in $\mathbb{Z}_p$ drawn from a suitable (generally Gaussian) distribution. There are two versions of this problem to consider. First, the search version: given a polynomial number of samples of such tuples, the problem is to find the secret $\vec{s}$. Second, the decision version: distinguish between noisy samples and random data. Though it intuitively seems that the decision version would be easier to solve, and it is trivially clear that solving the search version allows one to solve the decision version, it can be shown that the search version reduces to the decision version.

1.1 Reduction from Search to Decision

Consider an adversary A who can solve the decision version. That is, given a tuple $(\vec{a}, \langle \vec{a}, \vec{s} \rangle + e)$ as outlined above, A can distinguish between such a tuple and random data. This can be used to solve for $\vec{s}$ one component at a time. For the first component s_1 in $\vec{s}$ guess a value $k \in \mathbb{Z}_p$. Now choose a random and uniformly distributed r. Transform each of the given samples $(\vec{a_i}, b_i)$ by calculating the tuple $(\vec{a_i} + (r, 0, 0, ..., 0), b_i + rk)$. Now run this transformed tuple through the decision solver. If the guess for $k = s_1$ was correct, then the decision solver will recognize the transformed tuple as a noisy sample. Otherwise, since p is prime the transformation has a uniform random distribution and the decision solver recognizes it as random data. This process can be repeated for each component s_i of $\vec{s}$. Try all values of $s_i = k$ for each s_i and the decision solver will reveal when the guess for k is correct, allowing all components of $\vec{s}$ to be determined.

2 Constructing Encryption from LWE

Once we assume that solving the decision version of the LWE problem is hard, we can construct an encryption scheme that relies on this hardness. The hardness of LWE is shown to be reducible to the hardness of the approximate shortest vector problem, which is believed to be computationally hard, by Regev in [Reg05].

2.1 Private Key Encryption

2.1.1 Introductory Attempt at Private Key Encryption

Let $\vec{s}$ be a private key between Alice and Bob. Then we might attempt to create an encryption scheme by encoding 0 as a noisy sample of $\vec{s}$ by creating the tuple $(\vec{a}, \langle \vec{a}, \vec{s} \rangle + e)$ and encoding 1 as $(\vec{a}, b)$ where b is chosen at random. It is trivially clear that the encodings of 0 and 1 are indistinguishable from random data by the LWE assumption. However, this attempt does not give us a correct encryption scheme. If Alice sends $(\vec{a}, b)$ to Bob, then Bob can solve the formula $(\vec{a}, b) = (\vec{a}, \langle \vec{a}, \vec{s} \rangle + e)$ for e since he knows $\vec{a}$ and $\vec{s}$. If the calculated error is relatively small, that

is, $|e| < B$ for some constant B, then Bob knows that the given b is a representation of 0. However, if $|e| \geq B$ then Bob has no information about the value that b encodes.

2.1.2 Successful Attempt at Private Key Encryption

Only a small change is required to the introductory attempt in order to create a private key encryption system that is both secure and correct. Instead of encrypting 1 as $(\vec{a}, b)$ where b is completely random, we encrypt 1 as $(\vec{a}, \langle \vec{a}, \vec{s} \rangle + e)$ where e is shifted by $\frac{q}{2}$. The correctness of this scheme is clear from the introductory attempt. Bob will compute e from the formula $(\vec{a}, b) = (\vec{a}, \langle \vec{a}, \vec{s} \rangle + e)$ as before. If $|e| < B$ then Bob knows that b encodes 0. If $\frac{q}{2} + B < |e| < \frac{q}{2} - B$ then Bob knows that b encodes 1.

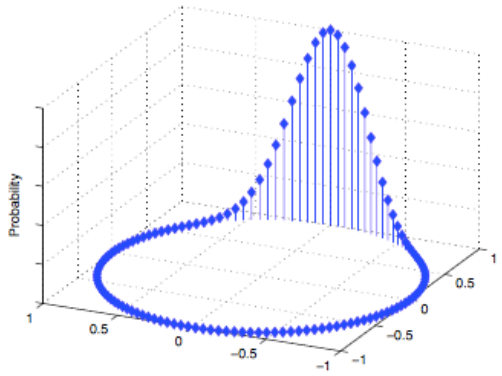

Figure 1: A Gaussian distribution of e, where the values of $\mathbb{Z}_p$ are distributed around the circle for $p = 127$ [Reg05]

2.1.3 Security of Private Key Encryption

It is more difficult to prove the security of our successful encryption scheme. Again the question is whether $Enc(0)$ and $Enc(1)$ can be distinguished by an adversary who does not know the secret key $\vec{s}$. Let A represent the adversary's algorithm and P_b be the probability that $A(Enc(b)) = 1$ for the given b. If P_b is different for $b = 0$ and $b = 1$ then A can successfully distinguish between $Enc(0) and Enc(1)$ if $abs P_0 - P_1 > \varepsilon$ where ε is on the order of $\frac{1}{poly(n)}$. However, if the adversary has such an efficient algorithm A then the adversary can solve the Learning with Errors problem in polynomial time. Suppose that the adversary is given the tuple $E_0 = (\vec{a}, \langle \vec{a}, \vec{s} \rangle + e)$ where $e \leftarrow x_0$. That is, e is not shifted by $\frac{q}{2}$. In this case, $Prob(A = 1) = P_0$. Now consider the case where the adversary is given the tuple $R_0 = (\vec{a}, b)$ where b is chosen randomly and is not shifted by $\frac{q}{2}$. In this case, $Prob(A = 1) = q_0$. We have already shown that E_0 cannot be distinguished from random data, so $P_0 = q_0$. Now we repeat this for encodings of 1. Suppose that the adversary is given $E_1 = (\vec{a}, \langle \vec{a}, \vec{s} \rangle + e)$ where $e \leftarrow x_0 + \frac{q}{2}$. Here $Prob(A = 1) = P_1$. Also consider the case where the adversary is given the tuple $R_1 = (\vec{a}, b)$ where b is chosen randomly then shifted by $\frac{q}{2}$. By the LWE assumption, E_1 cannot be distinguished from R_1 because if some algorithm A' could distinguish between the two then we could solve the LWE decision problem by adding $\frac{q}{2}$ to the given b and running A' on that input. Therefore, $P_1 = q_1$. Now, recall that in R_0 b is chosen randomly and is not shifted by $\frac{q}{2}$ whereas in R_1 b is chosen randomly and is then shifted by $\frac{q}{2}$. However, shifting a uniformly distributed random number by a constant results in a random number with the same distribution, so $q_0 = q_1$. Therefore, $P_0 = q_0 = q_1 = P_1$ so it must be impossible for such

a polynomial time algorithm A to exist if LWE is hard. Therefore, we can conclude that if LWE is hard then this encryption scheme is secure; that is, an adversary cannot distinguish between an encryption of 0 and an encryption of 1 efficiently.

2.2 Homomorphism of Private Key Encryption

The given private key encryption scheme is additively homomorphic. Given some ciphertext $C = c_1, ..., c_n$ where each $c_i = Enc(m_i)$ for some bit m_i, recall that each c_i takes the form $(\vec{a}, \langle \vec{a}, \vec{s} \rangle + e)$ where either $e \leftarrow x$ or $e \leftarrow x + \frac{q}{2}$ depending on if c_i encodes 0 or 1 respectively. Consider the case where we are attempting to add two values c_1 and c_2. Then $c' = c_1 + c_2 = (\vec{a_1} + \vec{a_2}, b_1 + b_2)$. This results in $c' = ((\vec{a_1} + \vec{a_2}), \langle \vec{a_1} + \vec{a_2}, \vec{s} \rangle + e_1 + e_2$. Let $e' = e_1 + e_2$. If $m_0 = m_1 = 0$ then $e' \leftarrow x + x$, which is the representation of 0 as expected. If $m_0 = m_1 = 1$ then $e' \leftarrow x + x + \frac{q}{2} + \frac{q}{2}$ and since all operations are modulo q, this is equivalent to $x + x$ which again is the representation of 0 as expected. If m_0 and m_1 are different then $e' \leftarrow x + x + \frac{q}{2}$ which is the representation of 1 as expected. From this, we can see that the encryption scheme is additively homomorphic.

2.2.1 Noise Bounds for Additive Homomorphism

LWE is useful in this encryption scheme because we can shift the Gaussian-distributed noise, allowing identification of encodings of 0 versus encodings of 1 given the private key $\vec{s}$. This identification can take place because the noise is bounded by some constant B. However, when we add components as shown above, the bound on the resulting noise gets looser. For each of the additions shown above, if $e' = e_1 + e_2$ and e_1 and e_2 each have a noise bound of B, then e' has noise bounded by $2B$. If $|B|$ is sufficiently smaller than q we can achieve a large number of additions before losing correctness. In general, with a bound B on noise and all operations taking place modulo q, we can add (XOR) $\frac{q}{2B}$ cyphertexts.

References

[Reg05] Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2005.