CS 4501-6501 Topics in Cryptography	2/2/2015
Lecture 6	
Lecturer: Mohammad Mahmoody	*Scribe: Kate Highnam, Matthew Irvine*

1 Public Key using LWE (and left-over hash lemma)

LWE Varies over a wide spectrum from "easier to break" to "hardest to break". The easier to break end is where the adversary only needs to "distinguish" (0 or 1). The hardest to break end is where the adversary must "find" some secret (p = ?).

Really this all does not matter because the hardest can be reduced to the easiest and vice versa until they are the same. To simplify this implementation, assume the basic idea that in LWE there is a secret $\vec{s}: (s_1, ..., s_n)$ and the adversary can request a "sample" and get $(\vec{a}, < \vec{a}, \vec{s} > +e)$, where the latter component is scalar and $\vec{a}$ is the message the adversary receives.

Assuming the noise distribution x has a "small" variance (uniquely identifiable) then s is uniquely identifiable from m items in $\{< \vec{a_i}, b > | 1 \leq i \leq m\}$.

2 Reduction of Tasks for the Adversary

The following subsections contain a short description of tasks with varying difficulty for an Adversary along with a quick reduction. Please ignore the subsection numbers; the version numbers following them indicate where the task stands on the scale of difficulty with 1 being the hardest and 5 being the easiest.

2.1 Version 1 - Find s

This task for the adversary is to find the secret key, s, with a probability of approximately $1-neg(n)$ where $neg(n)$ is some negligible number. Assuming no efficient adversary can do the previously stated is the weakest assumption.

2.2 Version 4 - Distinguish between Random World and Random s

For this version, the adversary must distinguish between the setting above where s is random and a "random" world, even by $\frac{1}{2} + \frac{1}{poly(n)}$. Here the main assumption made is that distinguishing is hard.

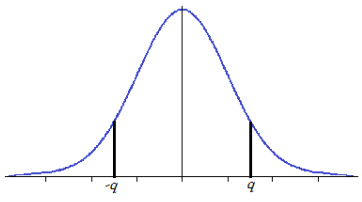

Figure 1: q: the base for modulos computation.

Reduction of Version 1 to Version 4: If Version 1 is solvable then use Version 4 to find "some" s, then go back and "test" s. If the result is consistent with samples then it is from the real world; otherwise it is inconsistent with the samples and must be from the random world. Finding s is sufficient for distinguishing between worlds.

2.3 Version 5 - Distinguish s within a Set

For a subset $S \subseteq {Z_q}^n$ of some "large" size : $\frac{|S|}{q^n} \geq \frac{1}{poly(n)}$, the task in this version is to distinguish whether $s \in S$ is being used or some random answers are returned by probability $\frac{1}{2} + \frac{1}{poly(n)}$.

Reduction of Version 4 to Version 5: If Version 4 is solved, then version 5 is solved (since 5 is even easier to solve). Think of the contra-positive form: if algorithm A is not able to distinguish between a s being used versus a random answer for almost all $s \in {Z_q}^n$ then A is not distinguishing $s \Leftarrow {Z_q}^n$ versus the random answer.

2.4 Version 3 - Distinguish with Higher Probability

The adversary is tasked with distinguishing between random s and random answers by approximately 100% probability.

Reduction of Version 3 to Version 4 Solving Version 3 trivially solves Version 4.

Reduction of Version 5 to Version 3 We can distinguish between $s \in S$ being used versus a random answer for "large" $\frac{|S|}{q^n} \geq \frac{1}{poly(n)}$ in Version 5. The goal of both Version 3 and Version 5 is to distinguish "random s" versus "random answers" with "high" probability. Assuming s is used and letting $(\vec{a}, < \vec{a}, \vec{s} > +e)$, where the latter component is b, be a noisy sample for s. Consider $(\vec{a}, b+ < \vec{a}, \vec{t} >)$ for a vector $\vec{t}$ and pretend this is a noisy sample for $\vec{s} + \vec{t}$. Then:

$(\vec{a}, < \vec{a}, \vec{s} + \vec{t} > +e) = (\vec{a}, < \vec{a}, \vec{s} > + < \vec{a}, \vec{t} > +e)$

This works since $\vec{s}$ is only shifted by $\vec{t}$ where we choose $\vec{t}$. Now let A be the algorithm to solve Version 5 and O be the "oracle" giving you random or noisy answers. Also let p^* be the estimate for $Prob[A \Rightarrow 1$ given random answers $]$

Then repeatedly sample random t to transform answers of O into "$s + t$"-based answers (pretending s exists). Let us call this oracle $\tilde{O}$ where $(s+t) \in S$ with $prob \geq \frac{1}{poly(n)}$. Using a "estimate" for the probability $p' = Prob[A \Rightarrow 1$ given answers from $\tilde{O}]$, if p' is far from p^*, then we are not in random case.

Assume solving Version 3 where distinguishable between completely random oracle O versus S is used with noisy answers with probability of 1. This can be used when solving Version 1, find s with probability of 1. This case relies on q being prime and working in Z_q.

Focus on s_1, a finite component of S, and $q < poly(n)$ where q is prime (assuming s exists). The goal now is to find enough to "test" if $s_1 = i$ or not for a fixed i. We currently have an Algorithm A that distinguishes between random answers and when some s exists.

The trick is to find where there is a transformation from $(\vec{a}, \vec{b})$, a random noisy sample from S, to $(\vec{a}', \vec{b}')$ such that if $s_1 = i$, $(\vec{a}', \vec{b}')$ is generated for some secret s'. However if $s_1 \neq i$, then $(\vec{a}, \vec{b})$ must be random vectors. (Hint: Shift the last part b by $(i - s_1)(random \in q)$ where q is prime)

3 Conclusion

Thus the easiest versions are equivalent to the hardest versions of LWE. By using the harness of the "easiest" version, the hardness of distinguishing between random values and $s \in S$ (or just random s) with noisy answers.