Adam Hesterberg

Lecture Notes

*6.876* *2015-11-16*

# 1 Guest Lecture by Silas Richelson

## 1.1 Review

Vinod's algorithm, GPV-Sample$(A, S, u, \sigma)$:

1. $A \in \mathbb{Z}_q^{n \times m}$
2. $S$ is a type 1 trapdoor for $A$ i.e. $S \in \mathbb{Z}^{m \times m}$ is a "short" matrix such that $AS \equiv 0 \pmod{q}$.
3. $\sigma$ (which should be larger than the largest norm of a vector in the Gram-Schmidt orthogonalization by something like $\omega(\sqrt{\log(n)})$) is a measure of smoothness.

The output is determined as in previous classes. (Choose a random $r$ from the discrete Gaussian $D_{\Lambda^\perp, u, \sigma}$, etc.)

The security property is that the following two distributions should be statistically close:

1. First choose $r$ from $D_{\mathbb{Z}^m, \sigma}$, then let $u = Ar$ (taken mod $q$, giving something in $\mathbb{Z}_q^n$)—that is, take a "random trapdoor".
2. First choose $r$ according to GPV-Sample$(A, S, u, \sigma)$, then choose $u$ randomly from $\mathbb{Z}_q^n$. (That is, run GPV-Sample).

This property should remind one of a trapdoor permutation: sampling a random point and computing its image should be indistinguishable from sampling a random point and computing its preimage.

## 1.2 Identity-Based Encryption

Goal: allow public keys to be anything, e.g. a name, so you don't have to do a key-generation process. But how would you generate a corresponding secret key?

Model: There's a master and many users $ID_1, \ldots, ID_n$. The master begins by generating a master public key MPK and a corresponding secret key MSK. If I want a secret key, I request it from the master, who generates it using their secret key somehow. The other users, but not the master, might collude to try to break the system. Encrypt using MPK and the identity of the recipient.

There are four parts:

1. IBE-Setup($1^n$) is the master's setup: outputs (MPK, SPK).

2. IBE-Extract(MPK,MSK,ID) is each person's setup: it outputs $SK_{ID}$, where $(PK_{ID}, SK_{ID})$ is a key pair. $PK_{ID}$ should be trivially obtainable from ID.

3. IBE-Encrypt($MPK, PK_{ID}, msg$) is encryption: outputs ciphertext $ct_{ID}$ (which depends on the message and identity).

4. IBE-Decrypt($ct_{ID}, SK_{ID}$) outputs msg.

There are a few necessary properties:

1. Correctness: For every message $msg$ and identity $ID$, if you do everything properly, it should work with high probability: that is, $IBE - Decrypt(IBE{-}Encrypt(MPK, PK_{ID}, msg), IBE{-}Extract(MPK, MSK, ID)) = msg$.

2. Security: No polynomial-time adversary should win the following game with probability more than $\frac{1}{2} + negl(n)$:

   The challenger gets $(MPK, MSK)$ from $IBE - Setup$. Then, polynomially many times, the adversary sends $ID_i$ and gets $SK_i$ from $IBE - Extract(MPK, MSK, ID_i)$. Then, the adversary sends an identity ID that's not one whose secret key they got in the previous phase and two messages $m_0$ and $m_1$; the challenger picks a random $b \in \{0, 1\}$ and sends the encrypted ciphertext $ct = IBE - Encrypt(MPK, ID, m_b)$. Then, the adversary can repeat the previous step (send ID, get secret key). Finally, the adversary guesses $b \in \{0, 1\}$, and wins iff they guess correctly.

   (This security property corresponds to CPA security. It can be modified to work for CCA security too.)

Construction:

1. IBE-Setup($1^n$) generates a master public and secret key by making a lattice with a trapdoor: that is, it takes a statistically random (according to the trapdoor statistical indistinguishability guarantee) $A \in \mathbb{Z}_q^{n \times m}$ along with a trapdoor $S$. MPK is A and MSK is S.

2. IBE-Extract($A, S, ID$): assuming a random oracle[1] $H$, let $u = H(ID) \in \mathbb{Z}_q^n$; choose $r$ according to GPV-Sample($A, S, u, \sigma$) (for any $\sigma$ sufficiently large for statistical indistinguishability), and output $(PK_{ID}, SK_{ID}) = (u, r)$. That is, $r \in \mathbb{Z}_q^m$ is a "short" (i.e. Gaussianly distributed) vector such that $Ar = u$.

[1] We should be able to remove this later.

Except, if you've been asked about the same $u$ (i.e. the same ID) before, output the same thing instead.[2]

3. Before definiting IBE-Encrypt, note that we can't just use Regev encryption. Recall that in Regev Encryption, you have an $A \in \mathbb{Z}_q^{n \times m}$, you generate a secret key $s \in \mathbb{Z}_q^n$, choose $e \in D_{\mathbb{Z}^m,r}$ and let $v = A^t s + e \in \mathbb{Z}_q^m$, which the LWE assumption says looks completely random, be the public key. To encrypt a message $b$ with $v$, choose $r \in D_{\mathbb{Z}^m,\sigma}$, let $u = Ar \in \mathbb{Z}_q^n$, let $w = v^t r + \frac{q}{2}b$, and output the ciphertext $(u, w)$. To decrypt a message $(u, w)$ with $s$, calculate $(w - s^t u)(\frac{2}{q})$ and round it.

   In Regev encryption, public keys are sparse in $\mathbb{Z}_q^m$. In the random oracle model of IBE, public keys are dense in $\mathbb{Z}_q^n$. That's why we can't just use Regev encryption.

   So, we'll use "Dual[3] Regev Encryption", which basically switches the key-generation and encryption algorithms.

   (a) Again, we have a public $A \in \mathbb{Z}_q^{n \times m}$.

   (b) To generate a key, choose $r \in D_{\mathbb{Z}^m,\sigma}$, let $u = Ar \in \mathbb{Z}_q^n$, and use $(pk, sk) = (u, r)$.

   (c) To encrypt a message $b$ with $u$, choose $s \in \mathbb{Z}_q^n$, $e \in D_{\mathbb{Z}^m,\sigma}$, and $e' \in D_{\mathbb{Z},\sigma}$, and let $v = A^t s + e \in \mathbb{Z}_q^m$, $w = u^t s + e' + (\frac{q}{2})b$ and output the ciphertext $(v, w)$.

   (d) To decrypt a message $(v, w)$ with secret key $r$, round $(w - v^t r)\frac{2}{q}$.

   (In regular Regev, the public key and ciphertext are $(v, u, w)$; in dual Regev, they're $(u, v, w)$. In regular Regev, the public key and ciphertext are $(s, r)$; in dual, they're $(r, s)$.)

   That's correct because $w - v^t r = (\frac{q}{2})b + (e' - e^t r)$, and the second term is short.

   Heuristically, the proof of security is about the same as the proof of security for regular Regev Encryption: use the leftover hash lemma to show that $u$ is statistically random, then use LWE on the random $w$.

   So, finally, define IBE-Encrypt$(A, u, b)$ as follows: choose $s \in \mathbb{Z}_q^n$, $e \in D_{\mathbb{Z}^m,\sigma}$, and $e' \in D_{\mathbb{Z},\sigma}$, and let $v = A^t s + e \in \mathbb{Z}_q^m$, $w = u^t s + e' + (\frac{q}{2})b$ and output the ciphertext $(v, w)$.

   Define IBE-Decrypt((v,w),r) as $\lfloor (w - v^t r)\frac{2}{q} \rceil$.

[2] This might not be necessary. Maybe, without this, if you get many $r$ for the same $u$, you can reconstruct $S$? It's convenient in the security proof: it makes the Adversary's attack indistinguishable from one in which they don't send any IDs, but we just choose the random oracle output ourselves. Then the keys $(u_i, r_i)$ we send the Adversary are a random $u_i \in \mathbb{Z}_q^n$ and a random $r_i$ from GPV-Sample$(A, S, u_i, \sigma)$, which is, by the GPV indistinguishability guarantee, indistinguishable from if we'd just generated the secret key $r_i$ randomly from $D_{\mathbb{Z}^m,\sigma}$ and multiplied by $A$ (mod $q$ for the public key $u_i = Ar_i$, which the Adversary can generate themself.

[3] This is not a technical term.

How can we remove the random oracle from that construction? We'll remove the random oracle along the lines of the classic tree construction of pseudorandom functions from pseudorandom generators.

We'll construct a tree as follows. The root is $A \in \mathbb{Z}_q^{n \times m}$ with a trapdoor $S$. Each identity will be some node of the tree, at which there'll be a matrix $A' = [A||\bar{A}] \in \mathbb{Z}_q^{n \times (m+\bar{m})}$ with a trapdoor $S'$, and we need to be careful that all those $S'$ don't reveal anything of $S$. So, we'll define an algorithm "Extend" to compute such a trapdoor. The $i$th column $S'_i$ of $S'$ will be computed as follows: draw $\bar{t}_i$ from $D_{\mathbb{Z}^{\bar{m}},\sigma}$ and $s'_i = \begin{pmatrix} t_i \\ \bar{t}_i \end{pmatrix}$. Set $u = -\bar{A}\bar{t}_i \in \mathbb{Z}_q^n$, so $A's'_i = At_i = -\bar{A}\bar{t}_i$. Draw $t_i$ from GPV-Sample$(A, S, u, \sigma)$.

Next lecture: using the above to get complete IBE. Also, ABE.

Last lecture by Vinod might be predicate-based encryption.