Lattices and Homomorphic Encryption, Spring 2013 Instructors: Shai Halevi, Tal Malkin

LWE-based Homomorphic Encryption

April 12-16, 2013 Scribe: Kina Winoto, Clément Canonne

We are going to describe the LWE-based homomorphic encryption scheme based on the works from [Gen09, BV11, BGV12, Bra12].

Parameters : Let n' be the security parameter, and we have $m = \text{poly}(n')$, $q > \text{super-poly}(n')$, and error bound $\sigma = \text{poly}(n')$. In general we think of q as "large" and all the other parameters as "small". Recall the following variant of the Regev LWE-based cryptosystem:

Key-Generation. Choose at random $A' \in_R \mathbb{Z}_q^{n' \times m}$ (random A'), $\vec{s'} \leftarrow \mathcal{D}_{\mathbb{Z}^{n'},\sigma}$ (small $\vec{s'}$), and $\vec{e'} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$ (small $\vec{e'}$). Set $\vec{a'} = \vec{s'}A' + \vec{e'} \bmod q$. We denote $n = n' + 1$,

$$A = \begin{pmatrix} A' \\ \vec{a'} \end{pmatrix} \in \mathbb{Z}_q^{n \times m},$$

and $\vec{s} = (\ \vec{s'}\ | -1) \in \mathbb{Z}_q^n$. The public key is $pk = A$ and the secret key if $sk = \vec{s}$. Note that both $\vec{s}$ and $\vec{s}A \bmod q = \vec{e'}$ are short vectors.

Encrypt$_A(b \in \{0,1\})$**.** Denote $\vec{b} = \lfloor \frac{q}{2} \rfloor \cdot (0 \ldots 0\, b\,)^T \in \mathbb{Z}_q^m$. Choose $\vec{r} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}$, and output the ciphertext $\vec{c} = A\vec{r} + \vec{b} \in \mathbb{Z}_q^n$.

Decrypt$_{\vec{s}}(\vec{c})$**.** Compute the inner-product $d = \langle \vec{s}, \vec{c} \rangle \bmod q$. Output 1 if $|d| > \frac{q}{4}$ and 0 if $|d| < \frac{q}{4}$.

Correctness. We note that $\langle \vec{s}, \vec{c} \rangle = \vec{s}(A\vec{r} + \vec{b}) = (\vec{s}A)\,\vec{r} + \langle \vec{s}, \vec{b} \rangle = \langle \vec{e'}, \vec{r} \rangle + \langle \vec{s}, \vec{b} \rangle \pmod q$. Since $\vec{e'}$ and $\vec{r}$ were chosen from an error distribution then they are both small and hence $|\langle \vec{e'}, \vec{r} \rangle| \ll q$. At the same time $\langle \vec{s}, \vec{b} \rangle = -b\lfloor \frac{q}{2} \rfloor$, hence $\langle \vec{e'}, \vec{r} \rangle + \langle \vec{s}, \vec{b} \rangle$ is closer to 0 when $b = 0$ and closer to $q/2$ when $b = 1$.

Security. If A was truly random then $A\vec{r}$ was close to random, even given A (because $\vec{r} \mapsto A\vec{r}$ is a good randomness extractor with seed A, and $\vec{r}$ has high min-entropy). Hence if A was random then the ciphertext would have no information on b, so guessing b implies distinguishing A from a random matrix, which is hard under the decision LWE assumption.

1 Homomorphic Encryption From Regev's Cryptosystem

Let $\vec{c}_i$, $i = 1, 2$ be two ciphertexts where $\vec{c}_i$ decrypts to $b_i \in \{0, 1\}$. Namely, we have

$$\langle s, c_i \rangle = \text{noise}_i + b_i \lfloor q/2 \rfloor \pmod q$$

for a small $|\text{noise}_i| \ll q$. It is easy to see that the scheme is additively homomorphic, if we set $\vec{c} = \vec{c_1} + \vec{c_2} \bmod q$ we have

$$\begin{aligned} \langle \vec{s}, \vec{c} \rangle &= \left(\text{noise}_1 + b_1 \left\lfloor \frac{q}{2} \right\rfloor\right) + \left(\text{noise}_2 + b_2 \left\lfloor \frac{q}{2} \right\rfloor\right) \\ &= \text{noise}_1 + \text{noise}_2 + \text{rounding-error} \ + \ (b_1 \oplus b_2) \left\lfloor \frac{q}{2} \right\rfloor \end{aligned}$$

This as long as the accumulated noise remain below $q/4$, we get have a valid encryption of $b_1 \oplus b_2$.

1.1 Multiplicative Homomorphism

Tensor products. Recall the *tensor (outer) product* between two vectors: if $\vec{a} = \langle a_1, \dots, a_s \rangle \in \mathbb{Z}^s$ and $\vec{b} = \langle b_1, \dots, b_t \rangle \in \mathbb{Z}^t$, then $\vec{a} \otimes \vec{b} = (a_i b_j)_{(i,j)\in[s]\times[t]} \in \mathbb{Z}^{st}$. Furthermore, we have the *mixed product property*:

$$\left\langle \vec{a}, \vec{b} \right\rangle \cdot \left\langle \vec{c}, \vec{d} \right\rangle = \left\langle \vec{a} \otimes \vec{c}\ ,\ \vec{b} \otimes \vec{d} \right\rangle \tag{1}$$

Multiplication, step 1 For $\vec{c}_i$ valid encryption of b_i ($i \in \{1, 2\}$), define $\vec{c}^* \stackrel{\text{def}}{=} \vec{c}_1 \otimes \vec{c}_2$ and $\vec{s}^* \stackrel{\text{def}}{=} \vec{s} \otimes \vec{s}$. Then,

$$\begin{aligned} \left\langle \vec{s}^*, \frac{2}{q}\vec{c}^* \right\rangle &= \frac{2}{q} \langle \vec{s}, \vec{c}_1 \rangle \cdot \langle \vec{s}, \vec{c}_2 \rangle = \frac{2}{q} \left(b_1 \cdot \frac{q}{2} + e_1 + k_1 q \right) \left(b_2 \cdot \frac{q}{2} + e_2 + k_2 q \right) \\ &= b_1 b_2 \cdot \frac{q}{2} + \underbrace{(2k_1 + b_1)e_1 + (2k_2 + b_2)e_2 + \frac{2e_1e_2}{q}}_{e''} + \underbrace{(2k_1k_2 + k_1b_2 + k_2b_1)}_{k''} \cdot q. \end{aligned}$$

(Note however that $\frac{2}{q}\vec{c}^*$ is no longer an integer vector, but one with rational entries.)

Since the k_i's are small, e'' is only a small factor larger than $e_1 + e_2$ (certainly $|e''| < n^3 (|e_1| + |e_2|)$); to get a valid ciphertext, we *round* $\vec{c}^*$. Let $\vec{\delta}$ be the rounding error:

$$\left\langle \vec{s}^*, \left\lceil \frac{2}{q}\vec{c}^* \right\rfloor \right\rangle = \left\langle \vec{s}^*, \frac{2}{q}\vec{c}^* \right\rangle + \langle \vec{s}^*, \delta \rangle = b_1 b_2 \cdot \frac{q}{2} + e'' + k'' \cdot q + \langle \vec{s}^*, \delta \rangle$$

Now, as $\vec{\delta}$ is small ($\|\vec{\delta}\,\|_\infty < 1/2$) and $\|\vec{s}^*\,\|_\infty = \|\vec{s} \otimes \vec{s}\,\|_\infty = \|\vec{s}\,\|_\infty^2$, the extra term $\langle \vec{s}^*, \delta \rangle$ is small; reducing modulo q, we set

$$\vec{c}\,'' \stackrel{\text{def}}{=} \left\lceil \frac{2}{q}\vec{c}^* \right\rfloor \mod q \tag{2}$$

so that

$$\left\langle \vec{s}^*, \vec{c}\,'' \right\rangle = b_1 b_2 \cdot \frac{q}{2} + e^* + k^* \cdot q$$

for $e^* = e'' + \langle \vec{s}^*, \delta \rangle$. As before, $|\langle \vec{s}^*, \vec{c}\,'' \rangle| \ll q^2$ (since $\vec{s}^*$ is small and $\|\vec{c}\,''\,\|_\infty < q$) so $k^* \ll q$. Therefore, $\vec{c}\,''$ is a valid encryption of $b_1 b_2$ relative to $\vec{s}^*$ (but with squared dimension).

Remark 1. to compute $\vec{c}\,''$, we just used the two ciphertexts $\vec{c}_1$, $\vec{c}_2$: nothing leaked from b_1, b_2.

Multiplication, step 2 (reducing the dimension). The idea is to add to the public key a "gadget" that will allow us to translate the high-dimensional $\vec{c}\,''$ (wrt $\vec{s}^*$) back to a low-dimensional $\vec{c}$ (wrt $\vec{s}$). Roughly, this gadget will be an encryption of $\vec{s}^*$ under $\vec{s}$, but relative to a larger modulus $Q \stackrel{\text{def}}{=} q^2$: For every entry i of $\vec{s}^*$, we add to the public key a vector $\vec{w}_i \in \mathbb{Z}_Q^n$ s.t.

$$\langle \vec{s}, \vec{w}_i \rangle = k_i Q + \vec{s}^*_i q + e_i$$

with $e_i \ll q = \sqrt{Q}$. Putting these vectors together in a matrix, we get $W \in \mathbb{Z}_Q^{n \times n^2}$ with

$$\vec{s} W = Q\vec{k} + q\vec{s}^* + \vec{e} \tag{3}$$

where $\vec{k}, \vec{e} \in \mathbb{Z}^{n^2}$ and $\|k\,\|_\infty, \|e\,\|_\infty \ll q$. We next show how to convert *any* valid encryption of some bit b relative to $\vec{s}^*$ (and q) into a valid encryption of b relative to $\vec{s}$ (and q):

Input $\vec{c}^*$ s.t. $\langle \vec{s}^*, \vec{c}^* \rangle = b \cdot \frac{q}{2} + k^* q + e^*$ (with $|e^*|, |k^*| \ll q$).

Output $\vec{c} \stackrel{\text{def}}{=} \left\lceil \frac{1}{q}\vec{c}^{*}W^{\mathrm{T}} \right\rfloor \mod q$.

Correctness: let $\vec{\delta}$ and $q\vec{k}'$ denote respectively the rounding error and the "mod q term".

$$\begin{aligned}
\langle \vec{s}, \vec{c} \rangle &= \left\langle \vec{s},\ \frac{1}{q}\vec{c}^{*}W^{\mathrm{T}} - \vec{\delta} - q\vec{k}' \right\rangle = \frac{1}{q}\vec{s}W(\vec{c}^{*})^{\mathrm{T}} - \left\langle \vec{s}, \vec{\delta} \right\rangle - q\left\langle \vec{s}, \vec{k}' \right\rangle \\
&\stackrel{Eq.(3)}{=} \frac{1}{q}\left\langle q^2\vec{k} + q\vec{s}^{*} + \vec{e}, \vec{c}^{*} \right\rangle - \left\langle \vec{s}, \vec{\delta} \right\rangle - q\left\langle \vec{s}, \vec{k}' \right\rangle \\
&= \langle \vec{s}^{*}, \vec{c}^{*} \rangle + q\left\langle \vec{k}, \vec{c}^{*} \right\rangle - q\left\langle \vec{s}, \vec{k}' \right\rangle + \frac{1}{q}\langle \vec{e}, \vec{c}^{*} \rangle - \left\langle \vec{s}, \vec{\delta} \right\rangle \\
&= b \cdot \frac{q}{2} + q\underbrace{\left(k^{*} + \left\langle \vec{k}, \vec{c}^{*} \right\rangle - \left\langle \vec{s}, \vec{k}' \right\rangle\right)}_{\tilde{k}} + \underbrace{\left(e^{*} + \frac{1}{q}\langle \vec{e}, \vec{c}^{*} \rangle - \left\langle \vec{s}, \vec{\delta} \right\rangle\right)}_{\tilde{e}}
\end{aligned}$$

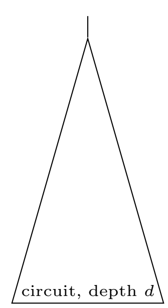
where $\tilde{e}$ is small (as a sum of three small terms); finally, since $\langle \vec{s}, \vec{c} \rangle = \tilde{k}q + b \cdot \frac{q}{2} + \tilde{e}$ with $\tilde{e}$ small and $\vec{s}$ small then $|\langle \vec{s}, \vec{c} \rangle| \ll Q = q^2$, so we also have $\tilde{k} \ll q$. Hence, $\vec{c}$ is a valid encryption of b wrt. $\vec{s}$ and q. $\square$

Parameters. We consider the "error" in the ciphertext to be the value $\langle \vec{s}, \vec{c} \rangle - q/2 \mod q$. The error grows with homomorphic operations, where for addition we have $|e| \simeq |e_1| + |e_2|$. For multiplication the noise grows a bit faster, and we have:

- (step 1) $|e^{*}| \simeq (|e_1| + |e_2|)n^3$
- (step 2) $|\tilde{e}| \simeq |e^{*}| + O(n^3)$

We see that no matter what operation, the error grows by at most a polynomial factor. How does that propagate in the circuit?

output: need error $< \frac{q}{4}$



inputs: error say n^3

At level i, we get an error which can be as big as n^{3i}, and for correctness we require that the error at the output node be smaller than $\frac{q}{4}$. We thus need $q > 4n^{3d}$, that is $\log q = \Omega(3d \log n)$ (recall that the scheme also requires $q \gg \mathrm{poly}(n)$, and that for security one must have $q \leq 2^{o(n)}$ (so that LLL cannot be used to break it)). Furthermore, we need D-LWE to be hard even modulo $Q = q^2$; all taken into account, $\boxed{n \geq \log^2 q}$ (say) is sufficient.

Key-Switching security: we have to explain why adding the W matrix to the public key does not compromise security. At first glance,

$$\vec{s}W = q\vec{s}^* + \vec{e} \mod Q$$

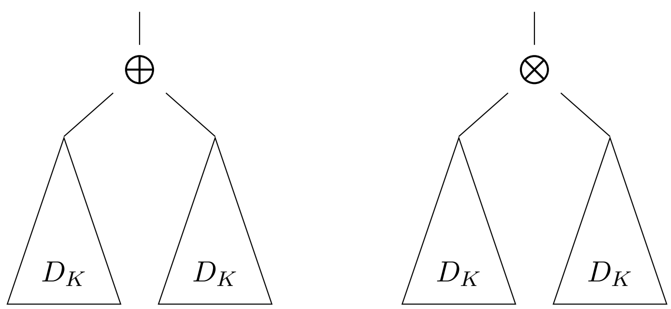
looks like a LWE problem, but the reduction to LWE that we used for the original cryptosystem does not work, because $\vec{s}^*$ is a function of $\vec{s}$. There are two common solutions to this issue:

- *Solution 1:* we can have a different secret key for each level i of the circuit, and encrypt $\vec{s}_i^*$ wrt. $\vec{s}_{i+1}$ (thus resolving the dependence, so that the reduction can be applied). The public key would contain all gadgets $W_{\vec{s}_i^* \to \vec{s}_{i+1}}$.
- *Solution 2:* define this as a new hardness assumption, the *"circular security assumption"*.

2 Bootstrapping

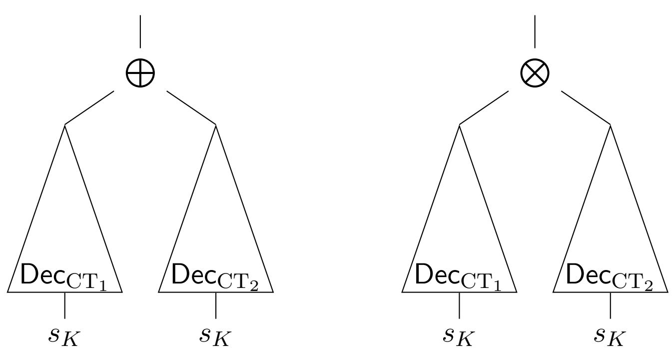
The homomorphic encryption scheme above has one drawback – in order to use it (set the parameters, and so on) the depth d of the circuit the ciphertexts will be fed into must be known and fixed in advance. How to have *one* cryptosystem which allows us evaluate *any* circuit – without committing on d beforehand?

Suppose we had a *homomorphic* cryptosystem with decryption circuit D_K, which can evaluate (without errors) the circuits



Then, we could "bootstrap" to any circuit by

- adding an encryption of the secret key to the public one (using the circular security assumption to argue it does not compromise security);
- then, given two ciphertexts CT_1, CT_2 that we want to add or multiply, considering the following two circuits C_{add}, C_{mult}:



where $\mathsf{Dec}_{\mathrm{CT}}$ is D_K with the ciphertext CT hard-wired; it takes as input an allowed secret key and tries to use it to decrypt[1]. When evaluating C_{add} on the encrypted bits of the secret key (which we get in the public key), what we get is an encryption of $C_{\text{add}}(\mathrm{CT}_1, \mathrm{CT}_2; s_K)$ (as we can by assumption homomorphically evaluate the sum of two D_K's): if CT_1, CT_2 are valid encryptions of b_1, b_2, then $C_{\text{add}}(\mathrm{CT}_1, \mathrm{CT}_2; s_K) = b_1 \oplus b_2$, so we obtain an encryption of $b_1 \oplus b_2$ (and similarly for $C_{\text{mult}}(\mathrm{CT}_1, \mathrm{CT}_2; s_K)$).

Wrapping it up All that remains to prove is that we have such a cryptosystem, which is able to homomorphically handle its own decryption. Consider the decryption algorithm given by

$$\mathsf{Dec}_{\vec{s}}(\vec{c}) \stackrel{\text{def}}{=} \left\lceil \frac{2}{q} \left(\langle \vec{s}, \vec{c} \rangle \mod q \right) \right\rfloor$$

where $\vec{s}, \vec{c} \in \mathbb{Z}_q^n$ (each entry needs $\log q$ bits). The input size is $n \log q$; since arithmetic is in $\mathsf{NC1}$, then decryption is in $\mathsf{NC1}$; and therefore our decryption circuit has depth $O(\log(n \log q)) = O(\log n)$ (as we require (a) $q = 2^{o(n)}$ for security). Because we need to support these $O(\log n)$ levels, q must also satisfy (b) $q > n^{O(\log n)}$. There is no inconsistency between (a) and (b), so this decryption algorithm is a good candidate for D_K.

References

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan, *Fully homomorphic encryption without bootstrapping*, ITCS, 2012, pp. 97–106.

[Bra12] Zvika Brakerski, *Fully homomorphic encryption without modulus switching from classical gapsvp*, Advances in Cryptology - CRYPTO'12, Lecture Notes in Computer Science, vol. 7417, Springer, 2012, pp. 868–886.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan, *Efficient fully homomorphic encryption from (standard) LWE*, FOCS, 2011, pp. 97–106.

[Gen09] Craig Gentry, *Fully homomorphic encryption using ideal lattices*, STOC, 2009, pp. 169–178.

[1] D_K takes as input both a ciphertext and a secret key; here, we fix some of its inputs.