**6.876J: Advanced Topics in Cryptography: Lattices** Fall 2015

# Lecture 14 - Fully Homomorphic Encryption

*Prof. Vinod Vaikuntanathan* *October 28, 2015*

## Overview

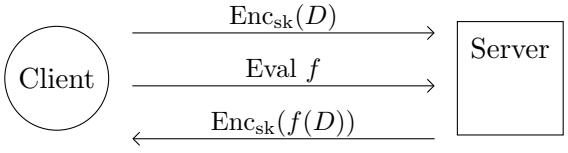
In the previous class we covered

- LWE. We looked at a search to decision reduction ($\text{SLWE}_{n,m,q,\chi} \leq \text{DLWE}_{n,m',q,\chi}$). However, the runing time of the reduction is $\text{poly}(q)$ and also $m = \text{poly}(1/\epsilon, m', q)$. Ideally, we would want running time $\text{poly}(\log q)$ and $m = O(m')$. This has been shown, but only when $\chi$ is a Gaussian.
- private and public-key encrpytion schemes from LWE.

In this lecture we will construct a fully homomorphic encryption scheme. Currently the only know construction of such a scheme is from LWE.

## Fully Homomorphic Encryption

**Motivation:** Through regular encryption we can store data remotely without revealing the data $D$ to the server. Is is possible to have the server compute some function $f$ on the data without having to decrypt it and send the encrypted result back to the client? Also, for what choice of functions is this possible?




We will show this is possible for both addition (XOR) and multiplication (AND).

**Definition 1.** *Let $\mathcal{C}$ be a class of circuits $C : \{0,1\}^n \to \{0,1\}$. An encryption scheme* (KeyGen, Enc, Dec, Eval) *is called $\mathcal{C}$-homomorphic if for all $f \in \mathcal{C}$ and ciphertexts $c_1, \ldots, c_n$,*

$$\text{Eval}(f, c_1, \ldots, c_n) = c^* \text{ such that if } \text{Dec}(c_i) = m_i \text{ for all } c_i \text{ then } \text{Dec}(c^*) = f(m_1, \ldots, m_n)$$

*For security, we use the regular notion of semantic security (or indistinguishability).*

**Definition 2.** *An encryption scheme is fully homomorphic if it is $\mathcal{C}$-homomoprhic where $\mathcal{C}$ is the class of all polynomial sized circuits.*

## Addition

We will start by looking how to compute addition over ciphertexts. In fact, we will use the private key scheme from last class. As a reminder, the secret key $s \in \mathbb{Z}_q^n$ and $m \in \{0,1\}$

$$\text{Enc}(s,m) = \left(a, \langle a, s\rangle + e + \left\lceil \frac{q}{2} \right\rfloor m\right)$$

where $e$ is some small random error term. To compute the addition, we will simply add the ciphertexts component-wise. So given plaintext-ciphertext pairs $(m_1, c_1)$ and $(m_2, c_2)$,

$$c_1 + c_2 = \left(a_1 + a_2, \langle a_1 + a_2, s\rangle + (e_1 + e_2) + \left\lceil \frac{q}{2} \right\rfloor (m_1 + m_2)\right)$$

So applying the decryption algorithm we would retrieve the parity of $m_1 + m_2$ given the total error is small ($|e_1 + e_2| \leq q/4$). It is easy to expand this to addition of $l = \text{poly}(n)$ ciphertexts. However, this will result in an accumulation of error. The total error of adding these $l$ ciphertexts is at most $l \cdot \max |e_i|$. So we need to start with small enough error ($|e_i| \leq \frac{q}{4l}$) to be able to decrypt after adding $l$ ciphertexts. For security, this means we need to assume the hardness of LWE for weaker parameters.

**Theorem 3** (WC-AC for LWE)**.**

$$\text{gapSVP}_{n, \frac{q}{\sigma}\text{poly}(n)} \leq \text{LWE}_{n,m,q,\chi=D_\sigma}$$

*where $D_\sigma$ is a discrete gaussian and $m$ only affects the running time of the reduction.*

Note that the approximation factor depends on the ratio between $q$ and $\sigma$.

Informally, we can use this theorem as a way to choose parameters. The best known algorithms for $\text{gapSVP}_{n,2^{n^\varepsilon}}$ takes time $2^{\widetilde{O}(n^{1+\varepsilon})}$. So we can set $\frac{q}{\sigma} = 2^{n^{.99}}$. For addition, $l = \text{poly}(n)$ at most $n^{\log n}$. So we let $q = n^{\log n}$ and $\sigma = \text{poly}(n)$ to do arbritrary additions.

## Multiplication

For multiplication we will use the Approximate Eigenvector Encryption Scheme [Gentry, Sahai, Waters '13; Brakerski '14]. The first uses an approximation factor of $n^d$ where $d$ is the depth of the circuit and the second work improves the approximation factor to $n^3$.

### A Non-Encryption Scheme

Consider the following:

$$\begin{aligned}
&\text{Public Key: } P \in \mathbb{Z}_q^{n\times n} \text{ where is } P \text{ is random matrix with nullity} > 0\\
&\text{Secret Key: } s \in \mathbb{Z}_q^n \text{ such that } sP \equiv 0 \pmod q
\end{aligned}$$

$$\begin{aligned}
&\text{Enc}(m) = PR + mI = C \in \mathbb{Z}_q^{n\times n}, \text{ where } R \xleftarrow{\$} \mathbb{Z}_q^{n\times n}\\
&\text{Dec}(C) = sC = s(PR + mI) = ms
\end{aligned}$$

Note that this scheme is insecure because we can recover $s$ given $P$, but we will ignore that for now. Encryption is essentially a OTP since $R$ is random $s$ is an eigenvector of $C$ with eigenvalue $m$. So we can both add and multiply cihpertexts. Let $sC = ms$ and $sC' = m's$.

$$s(C + C') = (m + m')s \qquad s(CC') = (mm')s$$

**Tweaking the Scheme**

We will modify the previous scheme to get closer to a real encryption scheme. We do not want $s$ to be in the kernal of $P$, but almost in the kernal ($sP \approx 0 \pmod q$). So we define the following scheme:

$$\begin{aligned}
&\text{Secret Key: } s = \begin{pmatrix} s' & -1 \end{pmatrix} \text{ where } s \text{ is an } n\text{-dimensional row vector and } s' \xleftarrow{\$} \mathbb{Z}_q^{n-1} \\
&\text{Public Key: } P = \begin{bmatrix} P' \\ s'P' + e \end{bmatrix} \text{ with } P' \xleftarrow{\$} \mathbb{Z}_q^{n-1 \times n} \text{ and error } e \\
&\qquad \text{Enc}(\mu) = PR + \mu I = C \in \mathbb{Z}_q^{n \times n}, \text{ where } R \xleftarrow{\$} \{0,1\}^{n \times n} \\
&\qquad \text{Dec}(C) = sC = s(PR + \mu I) = -eR + \mu s
\end{aligned}$$

Note $sP = \begin{pmatrix} s' & -1 \end{pmatrix} \begin{bmatrix} P' \\ s'P' + e \end{bmatrix} = -e$ which shows $s$ is close to being in the kernal. It is important that $R$ have small entries as otherwise the error term $-eR$ would become large and we would be unable to decrypt. This scheme is also not semantically secure. But let us try to add and multiply over this scheme. Let $sC = \tilde{e} + \mu s$ and $sC' = \tilde{e}' + \mu' s$ with $\tilde{e} = -eR$.

$$s(C + C') = (\tilde{e} + \tilde{e}') + (\mu + \mu')s \qquad s(CC') = (\tilde{e} + \mu s)C' = (\tilde{e}C' + \mu\tilde{e}') + \mu\mu' s$$

This is a problem for multiplication as $\tilde{e}C'$ could be a large error term since $P$ has large entries.

**Approximate Eigenvector Encryption Scheme**

We will make one more change to address the problem of the previous scheme. We increase the number of LWE sample from $n$ to $m = n(\lfloor \log q \rfloor + 1)$. The other change is replacing $I$ with a new matrix $G$. Like the scheme for addition, we need a way to account for noise of the error. So we will construct $G$ to perform error correction.

$$\begin{aligned}
&\text{Secret Key: } s = \begin{pmatrix} s' & -1 \end{pmatrix} \text{ where } s \text{ is an } n\text{-dimensional row vector and } s' \xleftarrow{\$} \mathbb{Z}_q^{n-1} \\
&\text{Public Key: } P = \begin{bmatrix} P' \\ s'P' + e \end{bmatrix} \text{ with } P' \xleftarrow{\$} \mathbb{Z}_q^{n-1 \times m} \text{ and error } e \\
&\qquad \text{Enc}(\mu) = PR + \mu G = C \in \mathbb{Z}_q^{n \times m}, \text{ where } R \xleftarrow{\$} \{0,1\}^{n \times m} \\
&\qquad \text{Dec}(C) = sC = s(PR + \mu G) = -eR + \mu sG
\end{aligned}$$

Let $G \in \mathbb{Z}^{n \times m}$ such that

$$G = \begin{bmatrix}
1 & 2 & \dots & 2^{\lfloor \log q \rfloor} & 0 & & & & \dots & 0 \\
0 & \dots & & 0 & 1 & \dots & 2^{\lfloor \log q \rfloor} & 0 & \dots & 0 \\
\vdots & & & & & & \ddots & & & \\
0 & \dots & & & & & 0 & 1 & \dots & 2^{\lfloor \log q \rfloor}
\end{bmatrix}$$

Also, $G = g \otimes I$, where $g = \begin{bmatrix} 1 & 2 & \dots & 2^{\lfloor \log q \rfloor} \end{bmatrix}$. So for any $a \in \mathbb{Z}_q$, consider finding a solution $v \in \mathbb{Z}^n$ satisfying $gv = a$ with small coefficients. One can see that the binary decomposition of $a$ is a solution. Likewise for a vector $a \in \mathbb{Z}_q^n$, a solution to $Gv = a$ with $v \in \mathbb{Z}^m$ having small coefficients is expanding each component of $a$ into its binary decomposition. $G$ is a linear operator given by matrix multiplication and $G^{-1}$, the inverse operator (not linear), expands $a$ into its component-wise binary decomposition.

Now, addition is identical to the previous scheme. So let us look at multiplication given ciphertexts $C, C' \in \mathbb{Z}_q^{n \times m}$. Instead of directly multiplying we will compute either

$$C \cdot G^{-1}(C') \text{ or } C' \cdot G^{-1}(C)$$

with $G^{-1}(C') \in \mathbb{Z}^{m \times m}$ expanding each column of $C'$ into its binary decomposition.

$$\begin{aligned} s(C \cdot G^{-1}(C')) &= (e + \mu sG)G^{-1}(C') \\ &= (eG^{-1}(C') + \mu sC') \\ &= (eG^{-1}(C') + \mu e') + \mu\mu' sG \end{aligned}$$

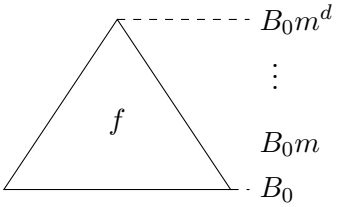
This gives us the desired result. Now we will bound the error term ($e_{\text{mult}}$). Since each entry in $G^{-1}(C')$ is in $\{0, 1\}$,

$$\|e_{\text{mult}}\| \leq m\|e\|_\infty + \mu\|e'\|_\infty$$

Note that while the product is correctly given back, the order of multiplication has an impact on the error. It is possible to get an error reduction. For example if $\mu = 0$, then we remove the $e'$ error.

The security of this scheme follows almost identical to Regev's public key scheme as our public key is many LWE samples. So we now have a scheme capable of computing addition and multiplication.

A crucial question remains of how many operation can one perform? Given a circuit $f$ of depth $d$. If we start with a magnitude of error $B_0 = \|e\|_\infty = \text{poly}(n)$, then as we evaluate the circuit at each level we pick up at most a factor of $m$ (assuming multiplication). So the final error is $B_0 m^d = m^{O(d)}$. Because we want to be able to decrypt at the end, we must have $m^{O(d)} < q/4$ or $(n \log q)^d < q$. So as a result we have a leveled fully homomorphic encryption scheme as $d$ must be specified in advance to choose a sufficiently small starting error. For security $q < 2^{n^\varepsilon}$ with $\varepsilon < 1$. This gives us a bound $d < \frac{n^\varepsilon}{\log n}$.

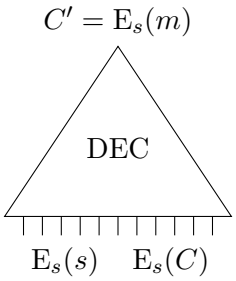



## Bootstrapping

However, we would like to compute circuits of arbitrary depth. Every method we currently know how to do this relies on the bootstrapping theorem [Gentry '09]. As we do computations, we

accumulate error. So we would like a procedure to do error reduction. Given a ciphertext $C$ with error $\eta$ that encrypts $m$, we want a ciphertext $C'$ with error $\eta' << \eta$ that encrypts $m$. Note that decrpytion itself is a good error reduction as it completely removes the noise, but we cannot hand over the secret key.

So instead we will hand over an encrpytion of the secret key ($\mathrm{E}_s(s)$). This circular encryption is no longer LWE and one needs to assume security under this new setting where the adversary has the secret key encrypted under itself.




Homomorphic encrpytion lets us compute functions over encrpyted inputs and get the encrypted output. So we can compute decryption on encrypted inputs. So this is an identity operation except the error has changed. The error of $C'$ does not depend on the error of $C$ as we used fresh error in the inputs. The final error is $n^{O(d_{\mathrm{DEC}})}\mathrm{poly}(n)$. So if we set $q > n^{O(d_{\mathrm{DEC}})}\mathrm{poly}(n)$ we can reduce the error of a ciphertext.

Therefore, we can compute an operation and then perform error reduction to get a ciphertexts with small error. Repeating this process we can evaluate arbitrary circuits while keeping the error small as $n^{O(d_{\mathrm{DEC}})}\mathrm{poly}(n)$ is fixed. So with bootstrapping we get a fully homomorphic encryption scheme.

**Open Questions:** Is is possible to do fully homomorphic encryption without making circular security assumptions? Either by showing the security of the circular object or coming up with a new construction?

Note that for Regev encryption, it is secure to release the encryption of the secret key. Consider each index of the key encrypted under itself,

$$\left(a, \langle a, s\rangle + e + s_i \left\lceil \frac{q}{2} \right\rfloor\right) = \left(a, \left\langle a + \left\lceil \frac{q}{2} \right\rfloor u_i, s \right\rangle + e\right) = \left(a' - \left\lceil \frac{q}{2} \right\rfloor u_i, \langle a', s\rangle + e\right)$$

where $u_i$ is the $i^{\text{th}}$ standard basis vector. The last equality holds from variable substitution and the security follows from noting this last expression is indistinguishable from random.