16

Fully homomorphic encryption : Construction

In the last lecture we defined fully homomorphic encryption, and showed the "bootstrapping theorem" that transforms a partially homomorphic encryption scheme into a fully homomorphic encryption, as long as the original scheme can homomorphically evaluate its own decryption circuit. In this lecture we will show an encryption scheme (due to Gentry, Sahai and Waters, henceforth GSW) meeting the latter property. That is, this lecture is devoted to proving[1] the following theorem:

Theorem: Assuming the LWE conjecture, there exists a partially homomorphic public key encryption $(G, E, D, EVAL)$ that fits the conditions of the bootstrapping theorem. That is, for every two ciphertexts c and c', the function $d \mapsto D_d(c)\ NAND\ D_d(c')$ is can be homomorphically evaluated by $EVAL$.

[1] This theorem as stated was proven by Brakerski and Vaikuntanathan (ITCS 2014) building a line of work initiated by Gentry's original STOC 2009 work. We will actually prove a weaker version of this theorem, due to Brakerski and Vaikuntanathan (FOCS 2011), which assumes a quantitative strengthening of LWE. However, we will not follow the proof of Brakerski and Vaikuntanathan but rather a scheme of Gentry, Sahai and Waters (CRYPTO 2013). Also note that, as noted in the previous lecture, all of these results require the extra assumption of *circular security* on top of LWE to achieve a non-leveled fully homomorphic encryption scheme.

16.1 Prelude: from vectors to matrices

In the linear homomorphic scheme we saw in the last lecture, the ciphertexts were vectors $c \in \mathbb{Z}_q^n$ such that $\langle c, s \rangle$ was equal (up to scaling by $\lfloor \frac{q}{2} \rfloor$) to the plaintext bit. We saw that adding two ciphertexts modulo q corresponded to XOR'ing (i.e., adding modulo 2) the corresponding two plaintexts. That is, if we defined $c \oplus c'$ as $c + c' \pmod q$ then performing the $\oplus$ operation on the ciphertexts corresponded to adding modulo 2 the plaintexts. However, to get to a fully, or even partially, homomorphic scheme, we need to find a way to perform the NAND operation on the two plaintexts. The challenge is that it seems that to do that we need to find a way to evaluate *multiplications*: find a way to define some operation $\otimes$ on ciphertexts that corresponds to multiplying the plaintexts. Alas, a priori, there doesn't seem to be a natural way to *multiply* two vectors.

The GSW approach to handle this is to move from vectors to *matrices*. As usual, it is instructive to first consider the cryptographer's dream world where Gaussian elimination doesn't exist. In this case, the GSW ciphertext encrypting $b \in \{0,1\}$ would be an $n \times n$ matrix C over $\mathbb{Z}_q$ such that $Cs = bs$ where $s \in \mathbb{Z}_q^n$ is the secret key. (Note here that s is a *vector* in $\mathbb{R}_q^n$ while b is a *number* in $\{0,1\}$.) That is, the encryption of a bit b is a matrix C such that the secret key is an *eigenvector* (modulo q) of C with corresponding eigenvalue b. (Let us defer discussion of how the encrypting party generates such a ciphertext, since this is in any case only a "dream" toy example.) Clearly, given C and s we can recover b. The scheme trivially allows homomorphic evaluation of both addition and multiplication, since if $Cs = bs$ and $C's = b's$ then we can define $C \oplus C' = C + C'$ (where on the righthand side, addition is simply done in $\mathbb{Z}_q$) and $C \otimes C' = CC'$ (where again this refers to matrix multiplication in $\mathbb{Z}_q$). Now,

$$(C + C')s = (b + b')s$$

and

$$CC's = bb's.$$

Since if $b, b' \in \{0,1\}$, then $b \ NAND \ b' = 1 - bb'$ and in particular this equation holds modulo q, we would be able to take an encryption C of b and an encryption C' of b' and transform it to the encryption $(I - CC')$ of $b \ NAND \ b'$ (where I is the identity matrix).

Thus in a world without Gaussian elimination it is not hard to get a fully homomorphic encryption.[2]

16.2 Real world partially homomorphic encryption

We now discuss how we can obtain an encryption in the real world where, as much as we'd like to ignore it, there are people who walk among us (not to mention some computer programs) that can actually do eigenvalue computations. As usual, the idea is to "fool Gaussian elimination with noise" but we will see that we have to be much more careful about "noise management", otherwise even for the party holding the secret key the noise will overwhelm the signal.[3]

The main idea is that we can expect the following problem to be hard for a random secret $s \in \mathbb{Z}_q^n$: distinguish between samples of random matrices C and matrices where $Cs = bs + e$ for some $b \in \{0,1\}$ and "short" e satisfying $|e_i| \leq \sqrt{q}$ for all i. This yields a natural candidate for an encryption scheme where we encrypt b by a matrix C satisfying $Cs = bs + e$ where e is a "short" vector.[4]

[2] Strictly speaking, we only showed in this world how to get a *private key* fully homomorphic encryption. Our "real world" scheme will be a full fledged *public key* FHE. However, we note that private key homomorphic encryption is already very interesting and in fact sufficient for many of the "cloud computing" applications. Moreover, Rothblum gave a generic transformation from a *private key* homomorphic encryption to a *public key* homomorphic encryption.

[3] For this reason, Craig Gentry called his highly recommended survey on fully homomorphic encryption and other advanced constructions computing on the edge of chaos.

[4] We deliberately leave some flexibility in the definition of "short". While initially "short" might mean that $|e_i| < \sqrt{q}$ for every i, decryption will succeed as long as long as $|e_i|$ is, say, at most $q/100$.

We can now see what adding and multiplying two matrices does to the noise. If $Cs = bs + e$ and $C's = b's + e'$ then

$$(C + C')s = (b + b')s + (e + e')$$

and

$$CC's = C(b's + e') + e = bb's + (b'e + Ce').$$

We would have loved to say that we can define as above $C \oplus C' = C + C' \pmod q$ and $C \otimes C' = CC' \pmod q$. For this we would need that $(C + C')s$ equals $(b + b')s$ plus a "short" vector and $CC's$ equals $bb's$ plus a "short" vector. The former statement indeed holds: if e, e' are "short" then $e + e'$ is not too long either. That is, if $|e_i| < \delta q$ and $|e'_i| < \delta q$ for every i then $|e_i + e'_i| < 2\delta q$. So we can at least handle a significant number of additions before the noise gets out of hand. Similarly, if e is small then so is $b'e$. Unfortunately, we can't really say that $CC's - bb's$ is "short" since Ce could be a very large vector. Indeed, since C looks like a random matrix in $\mathbb{Z}_q$, no matter how small the entries of e, many of the entries of Ce are quite likely to be of magnitude at least, say, $q/2$ and so multiplying e by C takes us "beyond the edge of chaos".

16.3 Noise management via encoding

The problem we had above is that the entries of C are elements in $\mathbb{Z}_q$ that can be very large, while we would have loved them to be small numbers such as 0 or 1. At this point one could say

> *"If only there was some way to encode numbers between 0 and $q-1$ using only 0 and 1"*

If you think about it hard enough, it turns out that there is something known as the "binary basis" that allows us to encode a number $x \in \mathbb{Z}_q$ as a vector $\hat{x} \in \{0,1\}^{\log q}$.[5] What's even more surprising is that this seemingly trivial trick turns out to be immensely useful. Let us denote by $\hat{s}$ the encoding of s as a vector in $\{0,1\}^{n \log q}$ and by $\hat{C}$ the encoding of an $m \times n$ C as an $m \times n \log q$ matrix with $0,1$ entries by encoding each row separately. (We still think of the entries of these vectors and matrices as elements of $\mathbb{Z}_q$ and so all calculations are still done modulo q.) We let Q be the $n \times (n \log q)$ "decoding" matrix where for every $i \in [n]$ and $j \in [\log q]$, $Q_{i,ij} = 2^{j-1}$ and all other entries are zero. It's a good exercise to verify that for every vector v and matrix C, $Q\hat{v} = v$ and $\hat{C}Q^\top = C$.

[5] If we were being pedantic the length of the vector (and other constant below) should be the integer $\lceil \log q \rceil$ but I omit the ceiling symbols for simplicity of notation.

In our final scheme the ciphertext encrypting b will be a $(n \log q) \times (n \times \log q)$ matrix C such that $Cv = bv + e$ for a "short" $e \in \mathbb{Z}_q^{n \log q}$ and $v = Q^\top s$ for $s \in \mathbb{Z}_q^n$.

Now given ciphertexts C, C' that encrypt b, b' respectively, we will define $C \oplus C' = C + C' \pmod q$ and $C \otimes C' = \widehat{CQ^\top} C'$. Since we have $Cs = bv + e$ and $C' = b'v + e'$ we get that

$$(C \oplus C')s = (C + C')s = (b + b')v + (e + e')$$

and

$(C \otimes C')s = \widehat{CQ^\top} C's = \widehat{CQ^\top}(b'Q^\top s + e') = b'CQ^\top s + \widehat{CQ^\top} e' = b'bv + (b'e + \hat{C}e')$.

Combining this we can see that if we define

$$C \overline{\wedge} C' = (I - C \otimes C')$$

then if C encrypts b and C' encrypts b' with noise parameters μ and μ' respectively then $C \overline{\wedge} C'$ encrypts $b \ \ NAND b'$ with noise parameter at most $\mu + n \log q \mu'$.

16.4 Putting it all together

We now describe the full scheme. We are going to use the $q(n)$-dLWE assumption for $q(n) = 2^{\sqrt{n}}$. It is not hard to show that we can relax our assumption to $q(n)$-LWE $q(n) = 2^{polylog(n)}$ and Brakerski and Vaikuntanathan showed how to relax the assumption to standard (i.e. $q(n) = poly(n)$) LWE though we will not present this here.

- **Key generation:** As in the scheme of last lecture the secret key is $s \in \mathbb{Z}_s^n$ with $s_1 = \lfloor \frac{q}{2} \rfloor$ and the public key is a generator G_s such that samples from $G_s(1^n)$ are indistinguishable from independent random samples from $\mathbb{Z}_q^n$ but if c is output by G_s then $|\langle c, s \rangle| < \sqrt{q}$, where the inner product (as all other computations) is done modulo q and for every $x \in \mathbb{Z}_q = \{0, \ldots, q-1\}$ we define $|x| = \min\{x, q - x\}$.

- **Encryption:** To encrypt $b \in \{0,1\}$, let $c_1, \ldots, c_{(} n \log q) \leftarrow_R G(1^n)$ output $C = \widehat{bQ^\top + D}$ where D is the matrix whose rows are $c_1, \ldots, c_{n \log q}$ and I is the $(n \log q) \times (n \log q)$ identity matrix.

- **Decryption:** To decrypt the ciphertext C, we output 0 if $|(CQ^\top s)_1| < 0.1q$ and 1 if $0.6q > |(CQ^\top s)_1| > 0.4q$. (It doesn't matter what we output on other cases.)

- **NAND evaluation:** Given ciphertexts C, C', we define $C \overline{\wedge} C'$ as $I - \widehat{CQ^\top} C'$.

16.5 Analysis of our scheme

To show that that this scheme is a valid partially homomorphic scheme we need to show the following properties:

1. **Correctness:** The decryption of an encryption of $b \in \{0,1\}$ equals b.
2. **CPA security:** An encryption of 0 is computationally indistinguishable from an encryption of 1 to someone that got the public key.
3. **Homomrphism:** If C encrypts b and C' encrypts b' then $C \overline{\wedge} C'$ encrypts $b \; NAND \; b'$ (with a higher amount of noise). The growth of the noise will be the reason that we will not get immediately a fully homomorphic encryption.
4. **Shallow decryption circuit:** To plug this scheme into the bootstrapping theorem we will need to show that its decryption algorithm (or more accurately, the function in the statement of the bootstrapping theorem) can be evaluated in depth $polylog(n)$ (independently of q), and that moreover, the noise grows slowly enough that our scheme is homomorphic with respect to such circuits.

Once we obtain 1-4 above, we have proven the existence of a fully homomorphic encryption scheme. We now address those points one by one.

16.5.1 Correctness

To ensure correctness, it suffices to show that for every bit b, if C is the encryption of b then it is an $(n \log q) \times (n \log q)$ matrix satisfying

$$CQ^\top s = bQ^\top s + e$$

where $\max |e_i| \ll q$.

For starters, let us see that the dimensions make sense: the encryption of b is computed by $C = b\widehat{Q^\top + D}$ where D is an $(n \log q) \times n$ matrix satisfying $|Ds|_i \leq \sqrt{q}$ for every i and I is the $(n \log q) \times (n \log q)$. Since $Q^\top$ is also an $(n \log q) \times n$ matrix, adding $bQ^\top + D$ makes

sense and applying the $\hat{\cdot}$ operation will transform every row to length $n \log q$ and hence C is indeed a square $(n \log q) \times (n \log q)$ matrix.

Let us now see what this matrix does to the vector $v = Q^\top s$. Using the fact that $\hat{M}Q^\top = M$ for every matrix m, we get that

$$Cv = (bQ^\top + D)s = bv + Ds$$

But by construction $|(Ds)_i| \leq \sqrt{q}$ for every i, hence completing the proof of correctness.

16.5.2 CPA Security

To show CPA security we need to show that an encryption of 0 is indistinguishable from an encryption of 1. However, by the security of the pseudorandom generator, an encryption of b computed according to our algorithm will be indistinguishable from an encryption of b obtained when the matrix D is a random $(q \log n) \times n$ matrix. Now in this case the encryption is obtained by applying the $\hat{\cdot}$ operation to $bQ^\top + D$ but if D is uniformly random then for every choice of b, $bQ^\top + D$ is uniformly random (since a fixed number plus a random number modulo q yields a random number) and hence the matrix $bQ^\top + D$ (and so also the matrix $\widehat{bQ^\top + D}$) contains no information about b. This completes the proof of CPA security (can you see why?). If we want to plug in this scheme in the bootstrapping theorem, then we will also assume that it is *circular secure*. It seems a reasonable assumption though unfortuantely at the moment we do not know how to derive it from LWE. (If we don't want to make this assumption we can still obtained a *leveled* fully homomorphic encryption as discussed in the previous lecture.)

16.5.3 Homomorphism

Let $v = Qs$, $b \in \{0,1\}$ and C be a ciphertext such that $Cv = bv + e$. We define the *noise* of C, denoted as $noise(C)$ to be the maximum of $|e_i|$ over all $i \in [n \log q]$. We make the following claim:

Lemma (noisy homomorphism): Let C, C' be ciphertexts encrypting b, b' respectively with $noise(C), noise(C') \leq q/4$. Then $C'' = C \overline{\wedge} C'$ encrypts $b \; NAND \; b'$ with

$$noise(C'') \leq (2n \log q) \max\{noise(C), noise(C')\}$$

Proof: This follows from the calculations we have done before. We get that

$\widehat{CQ^\top}C'v = \widehat{CQ^\top}(b'v + e') = b'\widehat{CQ^\top}Q^\top s + \widehat{CQ^\top}e' = b'(Cv) + \widehat{CQ^\top}e' = bb'v + b'e + \widehat{CQ^\top}e'$

But since $\widehat{CQ^\top}$ is a $0/1$ matrix with every row of length $n \log q$, for every i $(\widehat{CQ^\top}e')_i \leq (n \log q) \max_j |e_j|$. QED

16.5.4 Shallow decryption circuit

Recall that to plug in our homomorphic encryption scheme into the bootstrapping theorem, we needed to show that for every ciphertexts C, C' (generated by the encryption algorithm) the function

$$f(d) = D_d(C) \; NAND \; D_d(C')$$

can be homomorphically evaluated where d is the secret key and $D_d(C)$ denotes the decryption algorithm applied to C.

In our case the secret key is the descrption $\hat{s}$ of our vector s as a bit string of length $n \log q$. Given a ciphertext C, the decryption algorithm takes the dot product modulo q of s with the first row of $CQ^\top$ and outputs 0 (respectively 1) if the resulting number is small (respectively large). By repeatedly applying the noisy homomorphism lemma, we can show that can homorphically evaluate every circuit of NAND gates whose *depth* ℓ satisfies $(2n \log q)^\ell \ll q$. If $q = 2^{\sqrt{n}}$ then (assuming n is sufficiently large) then as long as $\ell < n^{0.49}$ this will be satisfied. In particular to show that $f(\cdot)$ can be homomorphically evaluated it will suffice to show that for every fixed vector $c \in \mathbb{Z}_q^{n \log q}$ there is a $polylog(n) \ll n^{0.49}$ depth circuit F that on input a string $\hat{s} \in \{0,1\}^{n \log q}$ will output 0 if $|\langle c, \hat{s} \rangle| < q/10$ and output 1 if $|\langle c, \hat{s} \rangle| > q/5$. (We don't care what F does otherwise. The above suffices since given a ciphertext C we can use f with the vector c being the top row of $CQ^\top Q$, and hence $\langle c, \hat{s} \rangle$ would correspond to the first entry of $CQ^\top s$. Note that if F has depth ℓ then the function $f()$ above has depth at most $\ell + 1$.)

If $c = (c_1, \ldots, c_{n \log q})$ is a vector then to compute its inner product with a $0/1$ vector s we simply need to sum up the numbers c_i where $s_i = 1$. Summing up n' numbers can be done via the obvious recursion in depth that is $\log n'$ times the depth for a single addition of two numbers. However, the naive way to add two numbers in $\mathbb{Z}_q$ will have depth $poly(\log q)$ which is too much for us. Fortunately, because we only care about accuracy up to $q/10$, we can drop all but the first $100 \log(n')$ most significant digits of our numbers, since including them can change the sum of the n numbers

by at most $n'(q/(n')^{100}) \ll q$. Hence we can easily do this work in $poly(\log n' = poly(\log n)$ depth.

Let us now show this more formally:

Lemma: For every $c \in \mathbb{Z}_q^m$ there exists some function $f : \{0,1\}^m \to \{0,1\}$ such that (1) $f(\hat{s}) = 0$ if $|\langle \hat{s}, c \rangle| < 0.1q$ (2) $f(\hat{s}) = 1$ if $0.4q < |\langle \hat{s}, c \rangle| < 0.6q$ and (3) there is a circuit computing f of depth at most $100(\log m)^3$.

Proof: For every number $x \in \mathbb{Z}_q$, write $\tilde{x}$ to be the number that is obtained by writing x in the binary basis and setting all digits except the $10 \log m$ most significant ones to zero.

Note that $\tilde{x} \leq x \leq \tilde{x} + x + q/m^{10}$. We define $f(\hat{s})$ to equal 1 if $|\sum \hat{s}_i \tilde{c}_i (\mod \tilde{q})| \geq 0.3\tilde{q}$ and to equal 0 otherwise (where as usual the absolute value of x modulo $\tilde{q}$ is the minimum of x and $\tilde{q} - x$.) Note that all numbers involved have zeroes in all but the $10 \log m$ most significant digits and so these less significant digits can be ignored. Hence we can add any pair of such numbers modulo $\tilde{q}$ in depth $O(\log m)^2$ using the standard elementary school algorithm to add two ℓ-digit numbers in $O(\ell^2)$ steps. Now we can add the m numbers by adding pairs, and then adding up the results, and this way in a binary tree of depth $\log m$ to get a total depth of $O(\log m)^3$. So, all that is left to prove is that this function f satisfies the conditions (1) and (2). Note that $|\sum \hat{s}_i \tilde{c}_i - \sum \hat{s}_i c_i| < mq/m^10 = q/m^9$ so now we want to show that the effect of taking modulo $\tilde{q}$ is not much different from taking modulo q. Indeed, note that this sum (before a modular reduction) is an integer between 0 and qm. If x is such an integer and we divide x by q to write $x = kq + r$ for $r < q$, then since $x < qm$, $k < m$, and so we can write $x = k\tilde{q} + k(q - \tilde{q}) + r$ so the difference between $k \mod q$ and $k \mod \tilde{q}$ will be (in our standard modular metric) at most $mq/m^{10} = q/m^9$. Overall we get that if $\sum \hat{s}_i c_i \mod q$ is in the interval $[0.4q, 0.6q]$ then $\sum \hat{s}_i \tilde{c}_i (\mod \tilde{q})$ will be in the interval $[0.4q - 100q/m^9, 0.6q + 100q/m^9]$ which is contained in $[0.3\tilde{q}, 0.7\tilde{q}]$. QED

This completes the proof that our scheme can fit into the bootstrapping theorem, hence completing the description of the fully homomorphic encryption scheme. **QED**

16.6 Example application: Private information retrieval

To be completed