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 16.1 — FHE from LWE. Assuming the LWE conjecture, there exists a partially homomorphic public key encryption $(G, E, D, EVAL)$ that fits the conditions of the bootstrapping theorem (Theorem 15.5). That is, for every two ciphertexts c and c', the function $d \mapsto D_d(c) \ NAND \ D_d(c')$ 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, every ciphertext was a vector $c \in \mathbb{Z}_q^n$ such that $\langle c, s \rangle$ equals (up to scaling by $\lfloor \frac{q}{2} \rfloor$) 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 define $c \oplus c'$ as $c + c' \pmod q$ then performing the $\oplus$ operation on the ciphertexts corresponds 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. 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. (We defer discussion of how the encrypting party generates such a ciphertext, since this is in any case only a "dream" toy example.)

P You should make sure you understand the *types* of all the identifiers we refer to. In particular, above C is an $n \times n$ *matrix* with entries in $\mathbb{Z}_q$, s is a *vector* in $\mathbb{Z}_q^n$, and b is a *scalar* (i.e., just a number) in $\{0,1\}$. See Fig. 16.1 for a visual representation of the ciphertexts in this "naive" encryption scheme. Keeping track of the dimensions of all objects will become only more important in the rest of this lecture.

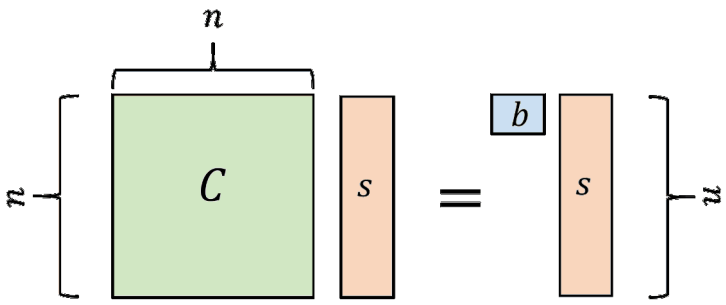

Figure 16.1: In the "naive" version of the GSW encryption, to encrypt a bit b we output an $n \times n$ matrix C such that $Cs = bs$ where $s \in \mathbb{Z}_q^n$ is the secret key. In this scheme we can transform encryptions C, C' of b, b' respectively to an encryption C'' of $NAND(b,b')$ by letting $C'' = I - CC'$.

Given C and s we can recover b by just checking if $Cs = s$ or $Cs = 0^n$. The scheme allows homomorphic evaluation of both addition (modulo q) 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 $\mathbb{Z}_q$).

Indeed, one can verify that both addition and multiplication

succeed since

$$(C + C')s = (b + b')s \tag{16.1}$$

and

$$CC's = C(b's) = bb's \tag{16.2}$$

where all these equalities are in $\mathbb{Z}_q$.

Addition modulo q is not the same as XOR, but given these multiplication and addition operations, we can implement the NAND operation as well. Specifically, for every $b, b' \in \{0,1\}$, $b \; NAND \; b' = 1 - bb'$. Hence we can take a ciphertext C encrypting b and a ciphertext C' encrypting b' and transform these two ciphertexts to the ciphertext $C'' = (I - CC')$ that encrypts $b \; NAND \; b'$ (where I is the identity matrix). Thus in a world without Gaussian elimination it is not hard to get a fully homomorphic encryption.

R **Private key FHE** We have not shown how to *generate* a ciphertext without knowledge of s, and hence 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.

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 actually know how to invert matrices. 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.[2]

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

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

$b \in \{0,1\}$ 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.[3]

[3] 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 try to check what adding and multiplying two matrices does to the noise. If $Cs = bs + e$ and $C's = b's + e'$ then

$$(C + C')s = (b + b')s + (e + e') \tag{16.3}$$

and

$$CC's = C(b's + e') + e = bb's + (b'e + Ce') \ . \tag{16.4}$$

I recommend you pause here and check for yourself whether it will be the case that if $C + C'$ encrypts $b + b'$ and CC' encrypts bb' up to small noise or not.

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. Looking at Eq. (16.4) we see that $(C + C')s$ equals $(b + b')s$ up to the "noise" vector $e + e'$, and 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.

However, if we consider Eq. (16.4), we see that CC' will be equal to $bb's$ plus the "noise vector" $b'e + Ce'$. The first component $b'e$ of this noise vector is "short" (after all $b \in \{0,1\}$ and e is "short"). However, the second component 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's and 1's"*

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 $\hat{x} \in \{0,1\}^{\log q}$.[4] What's even more surprising is that this seemingly trivial trick turns out to be immensely useful. We will define the *binary encoding* of a vector or matrix x over $\mathbb{Z}_q$ by $\hat{x}$. That is, $\hat{x}$ is obtained by replacing every coordinate x_i with $\log q$ coordinates $x_{i,0}, \ldots, x_{i,\log q-1}$ such that

$$x_i = \sum_{j=0}^{\log q-1} 2^j x_{i,j} \ . \tag{16.5}$$

[4] 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.

Specifically, if $s \in \mathbb{Z}_q^n$, then we denote by $\hat{s}$ the $n \log q$-dimensional vector with entries in $\{0,1\}$, such that each $\log q$-sized block of $\hat{s}$ encodes a coordinate of s. Similarly, if C is an $m \times n$ matrix, then we denote by $\hat{C}$ the $m \times n \log q$ matrix with entries in $\{0,1\}$ that corresponds to encoding every n-dimensional row of C by an $n \log q$-dimensional row where each $\log q$-sized block corresponds to a single entry. (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.)

While encoding in the binary basis is not a linear operation, the *decoding* operation is linear as one can see in Eq. (16.5). We let Q be the $n \times (n \log q)$ "decoding" matrix that maps an encoding vector $\hat{v}$ back to the original vector v. Specifically, every row of Q is composed of n blocks each of $\log q$ size, where the i-th row has only the i-th block nonzero, and equal to the values $(1, 2, 4, \ldots, 2^{\log q-1})$. It's a good exercise to verify that for every vector v and matrix C, $Q\hat{v} = v$ and $\hat{C}Q^\top = C$. (See Fig. 16.2 amd Fig. 16.3.)

In our final scheme the ciphertext encrypting b will be an $(n \log q) \times (n \log q)$ matrix C with small coefficients 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 $Cv = bv + e$ and $C'v = b'v + e'$ we get that

$$(C \oplus C')v = (C + C')v = (b + b')v + (e + e') \tag{16.6}$$

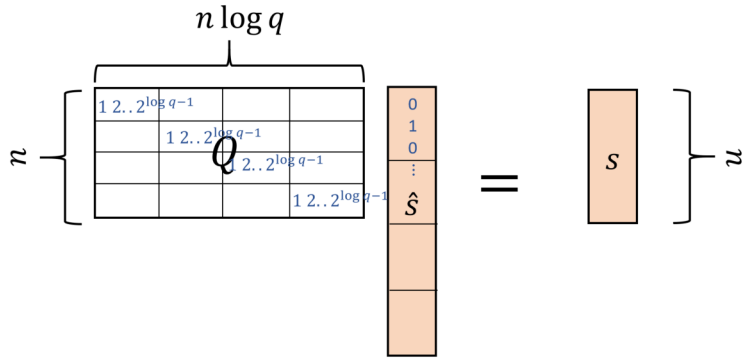

Figure 16.2: We can encode a vector $s \in \mathbb{Z}_q^n$ as a vector $\hat{s} \in \mathbb{Z}_q^{n \log q}$ that has only entries in $\{0,1\}$ by using the binary encoding, replacing every coordinate of s with a $\log q$-sized block in $\hat{s}$. The decoding operation is *linear* and so we can write $s = Q\hat{s}$ for a specific (simple) $n \times (n \log q)$ matrix Q.

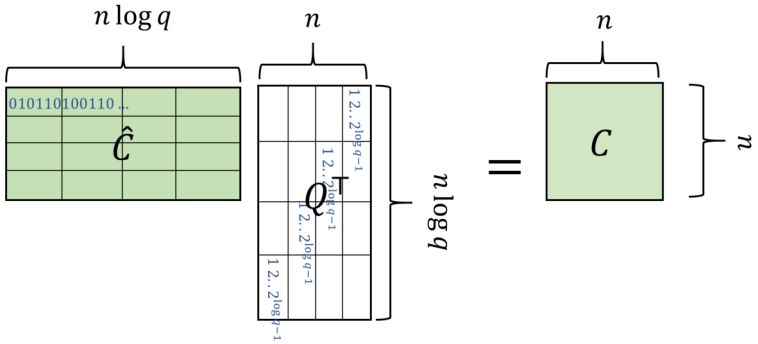

Figure 16.3: We can encode an $n \times n$ matrix C over $\mathbb{Z}_q$ by an $n \times (n \log q)$ matrix $\hat{C}$ using the binary basis. We have the equation $C = \hat{C}Q^\top$ where Q is the same matrix we use to decode a vector.

and

$$(C \otimes C')v = \widehat{(CQ^\top)}C'v = \widehat{(CQ^\top)}(bv + e') \; . \tag{16.7}$$

But since $v = Q^\top s$ and $\hat{A}Q^\top = A$ for every matrix A, the right-hand side of Eq. (16.7) equals

$$\widehat{(CQ^\top)}(b'Q^\top s + e') = b'CQ^\top s + \widehat{(CQ^\top)}e' = b'Cv + \widehat{(CQ^\top)}e' \tag{16.8}$$

but since $\widehat{B}$ is a matrix with small coefficients for every B and e' is short, the righthand side of Eq. (16.8) equals $b'Cv$ up to a short vector, and since $Cv = bv + e$ and $b'e$ is short, we get that $(C \otimes C')v$ equals $b'bv$ plus a short vector as desired.

If we keep track of the parameters in the above analysis, we can see that

$$C\overline{\wedge}C' = (I - C \otimes C') \tag{16.9}$$

then if C encrypts b and C' encrypts b' with noise vectors e, e' satisfying $\max |e_i| \leq \mu$ and $\max |e'_i| \leq \mu'$ then $C\overline{\wedge}C'$ encrypts $b \; NAND \; b'$ up to a vector of maximum magnitude at most $O(\mu + n \log q\mu')$.

16.4 *Putting it all together*

We now describe the full scheme. We are going to use a quantitatively stronger version of LWE. Namely, 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.

FHEENC:

- **Key generation:** As in the scheme of last lecture the secret key is $s \in \mathbb{Z}_s^n$ 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 $|x| = \min\{x, q - x\}$. As before, we can assume that $s_1 = \lfloor q/2 \rfloor$ which implies that $(Q^\top s)_1$ is

also $\lfloor q/2 \rfloor$ since (as can be verified by direct inspection) the first row of $Q^\top$ is $(1, 0, \ldots, 0)$.

- **Encryption:** To encrypt $b \in \{0,1\}$, let $d_1, \ldots, d_{(n \log q)} \leftarrow_R G_s(1^n)$ output $C = \widehat{(bQ^\top + D)}$ where D is the matrix whose rows are $d_1, \ldots, d_{n \log q}$ generated from G_s. (See Fig. 16.4)
- **Decryption:** To decrypt the ciphertext C, we output 0 if $|(CQ^\top s)_1| < 0.1q$ and output 1 if $0.6q > |(CQ^\top s)_1| > 0.4q$, see Fig. 16.5. (It doesn't matter what we output on other cases.)
- **NAND evaluation:** Given ciphertexts C, C', we define $C \overline{\wedge} C'$ (sometimes also denoted as $NANDEVAL(C, C')$) to equal $I - \widehat{(CQ^\top)}C'$, where I is the $(n \log q) \times (n \log q)$ identity matrix.

P Please take your time to read the definition of the scheme, and go over Fig. 16.4 and Fig. 16.5 to make sure you understand it.

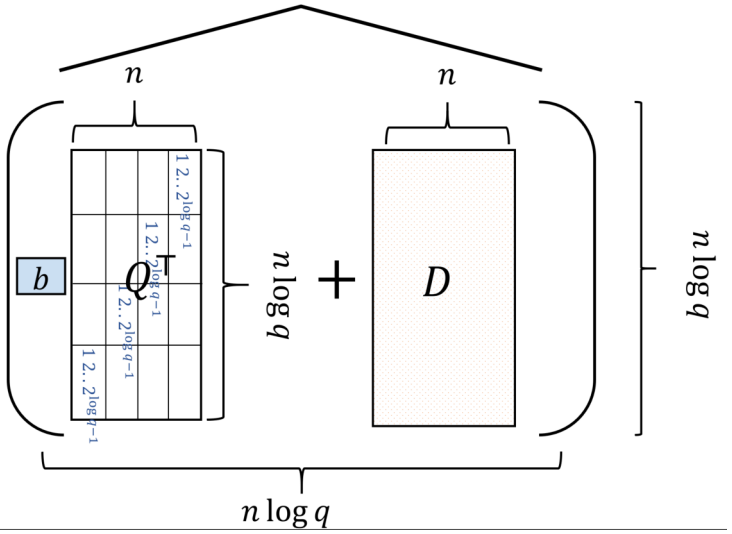

Figure 16.4: In our fully homomorphic encryption, the public key is a trapdoor generator G_s. To encrypt a bit b, we output $C = \widehat{(bQ^\top + D)}$ where D is a $(n \log q) \times n$ matrix whose rows are generated using G_s.

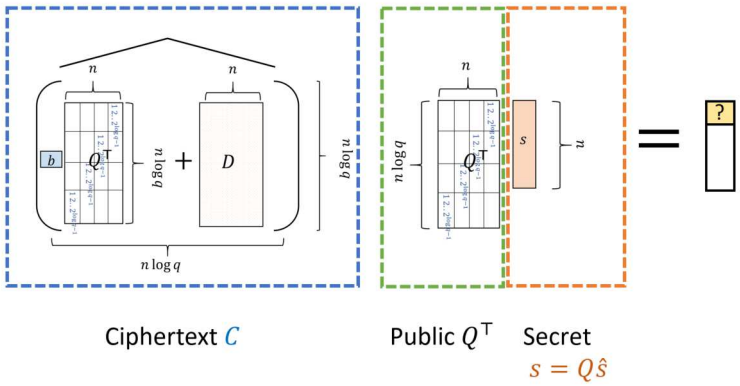

Figure 16.5: We decrypt a ciphertext $C = (\widehat{bQ^\top + D})$ by looking at the first coordinate of $CQ^\top s$ (or equivalently, $CQ^\top Q\hat{s}$). If $b = 0$ then this equals to the first coordinate of Ds, which is at most $\sqrt{q}$ in magintude. If $b = 1$ then we get an extra factor of $Q^\top s$ which we set to be in the interval $(0.499q, 0.51q)$. We can think of either s or $\hat{s}$ as our secret key.

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. **Homomorphism:** 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 can plug FHEENC into the Bootstrapping Theorem (Theorem 15.5) and thus complete the proof of existence of a fully homomorphic encryption scheme. We now address those points one by one.

16.5.1 Correctness

Correctness of the scheme will follow from the following stronger condition:

Lemma 16.2 For every $b \in \{0,1\}$, if C is the encryption of b then it is an $(n \log q) \times (n \log q)$ matrix satisfying

$$CQ^\top s = bQ^\top s + e \tag{16.10}$$

where $\max |e_i| \ll \sqrt{q}$.

Proof. For starters, let us see that the dimensions make sense: the encryption of b is computed by $C = (\widehat{bQ^\top + D})$ 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 $bQ^\top$ (i.e. either $Q^\top$ or the all-zeroes matrix, depending on whether or not $b = 1$) to 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 C does to the vector $v = Q^\top s$. Using the fact that $\hat{M}Q^\top = M$ for every matrix M, we get that

$$Cv = (bQ^\top + D)s = bv + Ds \tag{16.11}$$

but by construction $|(Ds)_i| \leq \sqrt{q}$ for every i. ∎

Lemma 16.2 implies correctness of decryption since by construction we ensured that $(Q^\top s)_1 \in (0.499q, 0.5001q)$ and hence we get that if $b = 0$ then $|(Cv)_1| = o(q)$ and if $b = 1$ then $0.499q - o(q) \leq |(C_v)_1| \leq 0.501q + o(q)$.

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 trapdoor 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 matrix plus a random

matrix yields a random matrix) 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 $\mu(C)$ to be the maximum of $|e_i|$ over all $i \in [n \log q]$. We make the following lemma, which we'll call the "noisy homomorphism lemma":

Lemma 16.3 Let C, C' be ciphertexts encrypting b, b' respectively with $\mu(C), \mu(C') \leq q/4$. Then $C'' = C \overline{\wedge} C'$ encrypts $b \ NAND \ b'$ and satisfies

$$\mu(C'') \leq (2n \log q) \max\{\mu(C), \mu(C')\} \tag{16.12}$$

Proof. This follows from the calculations we have done before. As we've seen,

$$\widehat{CQ^\top} C'v = \widehat{CQ^\top}(b'v + e') = b'\widehat{CQ^\top} Q^\top s + \widehat{CQ^\top} e' = b'(Cv) + \widehat{CQ^\top} e' = bb'v + b'e + \widehat{CQ^\top} e' \tag{16.13}$$

But since $\widehat{CQ^\top}$ is a $0/1$ matrix with every row of length $n \log q$, for every i $(\widehat{CQ^\top} e')_i \leq (n \log q) \max_j |e_j|$. We see that the noise vector in the product has magnitude at most $\mu(C) + n \log q \mu(C')$. Adding the identity for the NAND operation adds at most $\mu(C) + \mu(C')$ to the noise, and so the total noise magnitude is bounded by the righthand side of Eq. (16.12). ■

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 $f : \{0,1\}^{n \log q} \to \{0,1\}$ defined as

$$f(d) = D_d(C) \ NAND \ D_d(C') \tag{16.14}$$

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

In our case we can think of the secret key as the binary string $\hat{s}$ which describes 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 $CQ^\top$ (or, equivalently, the dot product of $\hat{q}$ with $CQ^\top Q$) and outputs 0 (respectively 1) if the resulting number is small (respectively large).

By repeatedly applying the noisy homomorphism lemma (Lemma 16.3), 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 $\ell + 1$.)

P Please make sure you understand the above argument.

If $c = (c_1, \ldots, c_{n \log q})$ is a vector then to compute its inner product with a 0/1 vector $\hat{s}$ we simply need to sum up the numbers c_i where $\hat{s}_i = 1$. Summing up m numbers can be done via the obvious recursion in depth that is $\log m$ times the depth for a single addition of two numbers. However, the naive way to add two numbers in $\mathbb{Z}_q$ (each represented by $\log q$ bits) will have depth $O(\log q)$ which is too much for us.

P Please stop here and see if you understand why the natural circuit to compute the addition of two numbers modulo q (represented as $\log q$-length binary strings) will require depth $O(\log q)$. As a hint, one needs to keep track of the "carry".

Fortunately, because we only care about accuracy up to $q/10$, if we add m numbers, we can drop all but the first $100 \log m$ most

significant digits of our numbers, since including them can change the sum of the n numbers by at most $m(q/m^{100}) \ll q$. Hence we can easily do this work in $poly(\log m)$ depth, which is $poly(\log n)$ since $m = poly(n)$.

Let us now show this more formally:

Lemma 16.4 For every $c \in \mathbb{Z}_q^m$ there exists some function $f : \{0,1\}^m \to \{0,1\}$ such that:

1. For every $\hat{s} \in \{0,1\}^n$ such that $|\langle \hat{s}, c \rangle| < 0.1q$, $f(\hat{s}) = 0$
2. For every $\hat{s} \in \{0,1\}^n$ such that $0.4q < |\langle \hat{s}, c \rangle| < 0.6q$, $f(\hat{s}) = 1$
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 $\tilde{x} \leq x \leq \tilde{x} + q/m^{10}$. 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 $[0.3\tilde{q}, 0.7\tilde{q}]$. ■

This completes the proof that our scheme can fit into the bootstrapping theorem (i.e., of Theorem 16.1), hence completing the description of the fully homomorphic encryption scheme.

Now would be a good point to go back and see you understand how all the pieces fit together to obtain the complete construction of the fully homomorphic encryption scheme.

16.6 Example application: Private information retrieval

To be completed