Lecture 22 — Homomorphic Encryption 4: Construction of fully homomorphic encryption.

Boaz Barak

April 21, 2010

Review of mildly homomorphic scheme

(Recall that $\lfloor x \rceil$ denotes the integer closest to x, breaking ties, say, downwards.)

Assumption We'll make the following "learning divisor with noise" assumption: **LDN Assumption:** let P a random n bit prime, R a random n^4 bit prime, and let $N = PR$. A distinguisher that is given N and $X_1, \ldots, X_{\text{poly}(n)}$ cannot distinguish between case **(I)** X_i's are chosen independently at random from $[N]$, and **(II)** $X_i = PQ_i + 2E_i \pmod{N}$ where Q_i is chosen independently at random from $[R]$ and E_i is chosen independently at random from $[-2^{n^{0.1}}, +2^{n^{0.1}}]$.

Note: Following Sushant's suggestion, I changed the LDN assumption so that N is an exact multiple of P. This makes reducing ciphertext size much easier, since now reducing modulo N doesn't introduce any additional noise (can you see why?). I also changed the parameters a bit (set N to have n^5 bits rather than $100n$), since there are in fact some attacks if N is not big enough. This makes no difference in anything we discussed last time. Adding the $\pmod{N}$ in case **(II)** does not make any difference as with $1 - negl(n)$ probability X_i will be a number between 1 and N— it's just a bit cleaner this way.

Revision of last lecture's scheme: Below is a slight variant of the private key mildly homomorphic scheme we showed on Monday. As I mentioned in class and you'll show in an exercise, a fully homomorphic private key encryption implies a fully homomorphic public key encryption, so once we get a fully homomorphic encryption we'll be fine.

Key We choose P to be a random n bit prime, and R to be a random n^4 bit prime, $N = PQ$. We keep P secret, and can publish N as a public parameter. (One can also think of N as being concatenated to every encryption.)

Encryption $\mathsf{Enc}_P^{\mathbf{E}}(b)$ denotes encryption of b with key P and noise parameter $\mathbf{E}$. It's defined as follows: choose $Q \leftarrow_{\text{R}} [N/P]$ and $E \leftarrow_{\text{R}} [-\mathbf{E}, +\mathbf{E}]$, and output $X = QP + 2E + b \pmod{N}$. We set the parameter $\mathbf{E}$ to be $2^{\sqrt{n}}$.

Decryption To decrypt X, output $X - \lfloor X/P \rceil P \pmod{2}$.[1]

[1]This is a close variant to the decryption algorithm of outputting $(X + 2\lfloor P/4 \rfloor \pmod{P}) \pmod{2}$ I showed last time since $X - \lfloor X/P \rceil P$ is the same as $X - \lfloor X/P + 0.5 \rfloor P$, which in our case (where X/P is very close to an integer) equals $X + 2\lfloor P/4 \rfloor \pmod{P}$ up to an even number.

Security and correctness of scheme The choice $2^{\sqrt{n}}$ for the parameter $\mathbf{E}$ makes the scheme both correct and secure. More generally, as long as $\mathbf{E} \ll P$ (say $\mathbf{E} < 2^{0.9n}$) then decryption will succeed, since $X - \lfloor X/P \rceil P$ will equal $2E + b$. As long as $\mathbf{E} > 2^{n^{0.1}}$ then under the LDN assumption the scheme will be secure.

Mild homomorphism We have the operations Add and Mult defined simply as $\mathsf{Add}(X, X') = X + X' \pmod N$ and $\mathsf{Mult}(X, X') = X \cdot X' \pmod N$. Let $\mathcal{E}^{\mathbf{E}}(b)$ denote the set of possible encryptions of b with parameter $\mathbf{E}$. That is $\mathcal{E}^{\mathbf{E}}(b) = \{X : X = QP + 2E + b, Q \in [N/P], E \in [-\mathbf{E}, +\mathbf{E}]\}$. Then if $X \in \mathcal{E}^{\mathbf{E}}(b)$ and $X' \in \mathcal{E}^{\mathbf{E}'}(b')$ then the calculations we did show that
(i) $\mathsf{Add}(X, X') \in \mathcal{E}^{2(\mathbf{E}+\mathbf{E}')}(b \oplus b')$ and
(ii) $\mathsf{Mult}(X, X') \in \mathcal{E}^{5\mathbf{E}\mathbf{E}'}(bb')$.

As a consequence, if we start with m ciphertexts with noise parameter $2^{\sqrt{n}}$ then we can add and multiply them and as long as we don't take a product of more than say $n^{1/10}$ of them then we'll still get ciphertexts of noise $\ll 2^n$ (and hence we can decrypt them). In particular we can apply to these ciphertexts any m variable polynomial over $GF(2)$ that has degree at most $n^{1/10}$ and at most polynomially many monomials.

Note that if $\mathbf{E}' \geq \mathbf{E}$ then $\mathcal{E}^{\mathbf{E}}(b) \subseteq \mathcal{E}^{\mathbf{E}'}(b)$.

Making it fully homomorphic

Clean and ReRand To make the scheme above fully homomorphic we'll add two operations to it:

- $\mathsf{Clean}(X)$ will take as input a ciphertext in $\mathcal{E}^{2^{n^{0.9}}}(b)$ and output a ciphertext in $\mathcal{E}^{2^{n^{0.3}}}(b)$. That is, it reduces the noise of the ciphertext.
- $\mathsf{ReRand}(X)$ will take as input a ciphertext in $\mathcal{E}^{2^{n^{0.4}}}(b)$ and output a ciphertext that distributed statistically close to the uniform distribution over $\mathcal{E}^{2^{\sqrt{n}}}(b)$, that is, $\mathsf{ReRand}(X) \approx \mathsf{Enc}(b)$.

Fully homomorphic encryption Together Clean and ReRand imply a fully homomorphic encryption scheme: to evaluate $NAND$ on two encryption, express $NAND$ as additions and multiplications, by writing

$$\overline{(b \wedge b')} = 1 \oplus bb'$$

and so given two ciphertexts X, X' in the range of the encryption algorithm encrypting b and b' respectively (i.e., $X \in \mathcal{E}^{2^{\sqrt{n}}}(b)$ and $X' \in \mathcal{E}^{2^{\sqrt{n}}}(b')$) we can compute a ciphertext $Y \in \mathcal{E}^{2^{7\sqrt{n}}}(\overline{b \wedge b'})$ by writing $Y = 1 + XX'$. (Adding 1 to a ciphertext flips its encrypted value, though if you prefer to be more "modular", you can also include an encryption of 1 in the public parameters.)

We then output $\mathsf{ReRand}(\mathsf{Clean}(Y))$. Since $\mathsf{Clean}(Y)$ will be in $\mathcal{E}^{2^{n^{0.3}}}(\overline{b \wedge b'})$ the output of $\mathsf{ReRand}(\mathsf{Clean}(Y))$ will be statistically close to a random encryption of $\overline{b \wedge b'}$.

Note: As you can see, we have considerable "slackness" in the parameters of ReRand and Clean, I chose these values to demonstrate that the parameter choice here needs to be done somewhat carefully, but it's not extremely fragile.

Our goal is now to get both Clean and ReRand. Clean is really the important one among those— the rerandomization property can often be achieved for many encryption schemes.

Getting ReRand We'll briefly mention how one can get ReRand, leaving verifying the details to the homework exercise. Our input is a ciphertext of the form $X = QP + 2E + b$ where $|E| \leq 2^{n^{0.4}}$. We want to transform it into $X' = Q'P + 2E' + b$ where Q' is uniform in $[R] = [N/P]$ and E' is uniform in $[-2^{\sqrt{n}}, +2^{\sqrt{n}}]$.

- *Rerandomizing noise:* if we just wanted to rerandomize the noise we could just choose E'' uniformly in $[-2^{\sqrt{n}}, +2^{\sqrt{n}}]$, and add $2E''$ to X. If we look at $E' = E + E''$ then this is distributed uniformly in the interval $[-2^{\sqrt{n}}, +2^{\sqrt{n}}] + E$ which is within $2^{n^{0.4}}/2^{\sqrt{n}} = negl(n)$ statistical distance to the uniform distribution over $[-2^{\sqrt{n}}, +2^{\sqrt{n}}]$.
- *Rerandomizing multiple:* rerandomizing Q is a bit more tricky. The idea is the following: suppose we have at our disposal many, say $X_1, \ldots, X_m$ for $m = n^6$, random encryptions of 0 with small noise (less than $2^{n^{0.4}}$). Then we will choose at random a subset $S \subseteq [m]$ and will look at the ciphertext $X'' = X + \sum_{i \in S} X_i$. This is still an encryption of 0 with at most $m2^{n^{0.4}}$ noise, and the corresponding multiple is just

  $$Q + \sum_{i \in S} Q_i \pmod R$$

  where $X_i = PQ_i + 2E_i$. We then use the following lemma (variant of what's known as "leftover hash lemma"):

  Lemma 1. *Let R be a k bit prime and suppose that $Q_1, \ldots, Q_m$ are chosen at random in $\mathbb{Z}_R$ where $m > 10k$. Then with probability at least $1 - 2^{-k/10}$ over the choice of $Q_1, ..., Q_m$, if we fix them and consider the random variable $Q = \sum_{i \in S} Q_i \pmod R$, where S is a random subset of $[m]$, then Q is within $2^{-k/10}$ statistical distance to the uniform distribution over $\mathbb{Z}_R$.*

- *Putting it all together:* We combine these to get ReRand as follows: as part of the public parameters (or concatenated to any encryption) we add ciphertexts $X_1, \ldots, X_m$ where $X_i = Q_iP + 2E_i$ with $Q_i \leftarrow_{\mathrm{R}} [R]$ and $E_i \leftarrow_{\mathrm{R}} [-2^{n^{0.4}}, +2^{n^{0.4}}]$. Then to rerandomize X we choose a random subset S of $[m]$, and E'' at random from $[-2^{\sqrt{n}}, +2^{\sqrt{n}}]$ and output

  $$X' = X + \sum_{i \in S} X_i + 2E'$$

Getting Clean using "wishful thinking" We now tackle the bigger problem - how to get the cleanup procedure Clean. This is very challenging, since up until this point it seems that any operation we do on ciphertexts, adding/multiplying/rerandomizing etc..., only increases the noise. In fact, it seems somewhat counterintuitive that you could decrease the noise without knowing the secret key, since if you could decrease it too much then you would be able to find out the plaintext!

Nevertheless, we'll show it may be possible to clean up the ciphertext, at least if we happened to be very lucky and the encryption scheme satisfies a certain property.

Let us consider the decryption algorithm Dec. The algorithm takes as input the secret key P and a ciphertext X and outputs the corresponding bit b. Since P and X are in the end represented by bits, Dec is just a function mapping $\{0,1\}^m$ to $\{0,1\}$ (where $m = n + n^5$ is the length of this description; this is not the same number m we used in the ReRand operation).

This is an *efficient* function, and so it can be computed by a polynomial size Boolean circuit, and we can assume that the gates of this circuit are only $\cdot$ and $\oplus$ (plus the constants $0, 1$)

since they are universal. Now lets say the *degree* of the circuit is largest number of its inputs that are ever multiplied together.[2] (For example, if the circuit has multiplicative depth ℓ then the degree is at most 2^ℓ.)

Let's suppose that we are lucky and the degree of the circuit Dec is at most $n^{0.01}$. We claim that in this case we can run the Clean operation as follows:

Recall that we're given an input $X = QP + 2E + b$ where $|E| \le 2^{n^{0.9}}$ and our goal is to come up with $X' = Q'P + 2E' + b$ such that $|E'| \le 2^{n^{0.3}}$. We are going to do the following:

- We change the scheme to include $Y_1, \dots, Y_n$ in the public parameters where $Y_i = \mathsf{Enc}_P^{2^{n^{0.1}}}(P_i)$ with P_i being the i^{th} bit of P. That is, we include an encryption of P in the public parameters, using noise value $2^{n^{0.1}}$ which is smaller than the standard parameter, but still big enough to ensure security.
- In the cleanup operation we define $Y_{n+1}, \dots, Y_m$, (where $m = n + n^5$) according to the bits of the ciphertext X. That is, Y_{n+1} is 1 if the first bit of X is 1 and 0 otherwise, etc.. Note that we can think of the number 1 also as an encryption of 1 (after all $1 = 0 \cdot P + 2 \cdot 0 + 1$) and similarly we can think of the number 0 also as an encryption of 0.

  Therefore, we now have ciphertexts $Y_1, \dots, Y_m$ that are encryptions of the $P \circ X$ where $\circ$ denotes concatenation. Moreover, these ciphertexts Y_i have very low noise! That is, each one of them has noise at most $2^{n^{0.1}}$. (Where our goal is at the end to get a ciphertext of noise $2^{n^{0.3}}$.)

  Now we know that $\mathsf{Dec}(P \circ X) = b$, and so if we run the circuit Dec on the encryptions $Y_1, \dots, Y_m$ we should get a ciphertext X' encrypting b which is exactly what we wanted! (This argument is so beautiful it deserves all the exclamation marks it gets...)

  The only thing left to verify is that the noise of X' is not that large. The idea is that because the circuit is simple, and we started from ciphertexts with small noise then we will get a ciphertext with not too large noise. Formally, we have the following lemma (naturally left as an exercise...)

  Lemma 2. *Suppose that C is a Boolean circuit with $\cdot, \oplus$ gates mapping $\{0,1\}^m \to \{0,1\}$ with degree at most d, and let $Y_1, \dots, Y_m$ be ciphertexts such that $Y_i \in \mathcal{E}^{\mathbf{E}}(b_i)$. Let Y be the result of applying C to $Y_1, \dots, Y_m$ (by replacing addition and multiplication with the corresponding* Add *and* Mult *operations— add and multiply modulo N). Then*

  $$Y \in \mathcal{E}^{\mathbf{E}^{(3d)^2}}\big(C(b_1 \cdots b_m)\big)$$

  Noting that in our case $\mathbf{E} = 2^{n^{0.1}}$ and $d = n^{0.01}$, and so $\mathbf{E}^{(3d)^2} = \left(2^{n^{0.1}}\right)^{9n^{0.02}} \le 2^{n^{0.2}}$, the proof is complete.

A relatively minor issue Unfortunately, it turns out that CPA security does not guarantee that it is secure to encrypt the secret key with itself (exercise...). We overcome this issue by using a common cryptographic technique— making an assumption: we'll assume that even given

[2] A more formal way to define degree is to note that the value of any gate of the circuit is always some m-variable polynomial $F : \mathrm{GF}(2)^m \to \mathrm{GF}(2)$, the degree of the circuit is the maximum degree of any polynomial that appears in its gates. In particular, if a circuit has degree d then the output of the circuit has to be a polynomial of degree at most d.

oracle access to the encryption oracle, one cannot distinguish an encryption of the secret key and an encryption of the all zero string.

This notion is called *circular security* and it is a subclass of a more general notion of *key dependent message (KDM) security.* While there are examples of CPA (and even CCA) secure schemes that are not circular secure, there are no known attacks against natural cryptosystems (e.g., El-Gamal etc..) and so it seems a reasonable assumption to assume that they are circular secure. In recent years, a few encryption schemes were proven to be circular secure (and satisfy some notions of KDM security as well) under relatively standard assumptions. It is also easy to construct KDM secure schemes in the random oracle model, and there are ways to try to combine this construction with the homomorphic scheme to make it even more likely it is circular secure.

In any case, we will assume this scheme is circular secure. Hopefully at some point someone will manage to prove that it is, and get rid of this assumption.[3]

As I already mentioned, the question of getting any plausible homomorphic encryption scheme, even with only heuristic security such as the random oracle model, was open for 30 years, so we shouldn't complain too much even if the solution uses somewhat non-standard assumptions.

A major issue The major issue is that we had no reason to believe that our circuit will be of degree at most $n^{0.01}$. Generally a circuit over $\{0,1\}^m$ is expected to have degree about $m \sim n^5$, and indeed I believe one can verify that the decryption circuit actually computes a polynomial of degree at least $n/100$ (and hence we cannot implement it by a circuit with degree smaller than that). So we're off by a polynomial factor.

We will tackle this issue by making an additional tweak to the encryption scheme, intended to "squash" the decryption circuit and make it of smaller degree.

Getting the Clean operation— squashing decryption

While at a high level, what we do amounts to squashing the decryption circuit, and in the papers is described in this way, our goal is just to get the Clean operation in some way. Thus, we will just show what we do to implement Clean. We will not change the actual encryption and decryption algorithms at all, just add some public parameters. Thus, all the properties of our scheme (correctness, homomorphism, and rerandomization) will be preserved.

Sparse subset sum assumption We'll need to use another cryptographic assumption. The *subset sum* problem is the question, given m n-bit numbers $\alpha_1, \ldots, \alpha_m$ and a target number β, whether there exists a subset $T \subseteq [m]$ such that $\sum_{i \in T} \alpha_i = \beta$. When m is sufficiently large as a function of n then this is considered a hard problem. We will assume hardness of a average-case decision variant of this problem where the set T is relatively small (of size n^ϵ for some $\epsilon > 0$).

Sparse Subset Sum (SSE) Assumption: For any $n, \epsilon > 0$ and $\beta \in [-2^n, +2^n]$, the following two distributions on $\alpha_1, \ldots, \alpha_m$ are computationally indistinguishable: Case **(I)**: $\alpha_1, \ldots, \alpha_m$ are chosen randomly and independently in $[-2^n, +2^n]$ and Case **(II)** we choose

[3]Even without this assumption one can get a limited homomorphic encryption scheme, where the public key grows with the depth of the circuit, see the papers and Gentry's thesis.

a set $T \subseteq [m]$ of size n^ϵ at random, for $i \notin T$ choose α_i randomly and independently from $[-2^n, +2^n]$, while on the other hand we ensure that $\sum_{i\in T}\alpha_i = \beta$. (Technically we do so by setting for every $i \in T$, α'_i be a random number in $[-2^n, +2^n]$, letting $\beta' = \sum_{i\in T}\alpha'_i$ and setting $\alpha_i = \alpha'_i - \beta' + \beta/|T|$.)

Actually we'll use the SSE assumption for n bit fractions in $(-1, +1)$. This is clearly equivalent by just scaling by a factor of 2^n.

Implication of SSE assumption The subset sum assumption implies that we will not harm security if we include in the public parameters numbers $\alpha_1, \ldots, \alpha_m \in (-1, +1)$ chosen such that there is a set T of size $k = n^{1/1000}$ satisfying that $\sum_{i\in T}\alpha_i = 1/P$.

(We assume here exact equality but in reality we'll truncate the numbers after say the n^5 digit and that will ensure this equality holds with precision as good we need.)

Now, instead of an encryption of P, we will provide in the public parameters an encryption of T. Specifically, we will set for $i \in [m]$ Y_i to be an encryption (with noise parameter $2^{n^{0.1}}$) of 1 if $i \in T$ and of 0 if $i \notin T$.

We also make the circular security assumption that this is safe to publish in the public parameters.

Implementation of Clean: Now we can show the implementation of Clean. Again, we are given a ciphertext $X = QP + 2E + b$ where $|E| \le 2^{n^{0.9}}$ and our goal is to come up with $X' = Q'P + 2E' + b$ such that $|E'| \le 2^{n^{0.3}}$. We are going to do the following:

1. Let $a = X \pmod 2$. Note that $b = (X - \lfloor X/P \rceil P) \pmod 2$ and hence, (since P is odd) $b = a \oplus \lfloor X/P \rceil \pmod 2$. Our goal will be to produce a ciphertext X'' encrypting $a' = \lfloor X/P \rceil \pmod 2$. We can then set $X' = X'' + a$.
2. We compute the numbers $Z_1, \ldots, Z_m$ where $Z_i = X\alpha_i$. Note that $\sum_{i\in T} Z_i = X/P$. We let $\tilde{Z}_i$ denote the truncation of number $Z_i + 2$ where we throw away all the binary digits corresponding to 2^i for $i > 0$ or $i \le -2\log k$. We claim that

$$\left\lfloor \sum_{i\in T} \tilde{Z}_i \right\rceil \pmod 2 = \lfloor X/P \rceil \pmod 2$$

   this is because **(1)** throwing the digits corresponding to $2^{-2\log k}$ or smaller can introduce at most $k/k^2 < 1/k < 1/100$ difference to the sum of the Z_i's. But since X/P was very close to an integer, after changing it by adding a number in $[-1/100, +1/100]$ it will still round to the same number and **(2)** changing digits corresponding to 2^1 or larger will only add or substract an even number from X/P, and that will not change its value modulo 2.

   Adding 2 clearly doesn't change the value $\pmod 2$ and just allows us to assume that the $\tilde{Z}_i$'s are numbers between 0 and 2 rather than between -1 and 1. That is, they are numbers that written in binary have the form $x.xxxxx$— one binary digit to the left of the "binary point" and $2\log k - 1$ digits to its right.
3. Let y_i be equal to 1 if $i \in T$ and to 0 if $i \notin T$. Let $z_{i,1}, \ldots, z_{i,2\log k}$ denotes the binary digits of the number $y_i\tilde{z}_i$. That is, if $y_i = 0$ then $z_{i,j} = 0$ for all j, and otherwise $z_{i,j}$ is the j^{th} digit of $\tilde{Z}_i$.

We can obtain $Z_{i,j}$ that is a ciphertext for $z_{i,j}$ by multiplying the ciphertext Y_i from the public parameter with the j digit of $\tilde{Z}_i$.

We now consider the following two functions:

- $SUM : \{0,1\}^{m(2\log k)} \to \{0,1\}^{3\log k}$ takes as inputs m numbers of $2\log k$ bits in $[0,2)$ of the form $x.xxxxx$, and computes their sum, where we have the guarantee that at most k of these numbers are not zeroes.
- $ROUND : \{0,1\}^{2\log k} \to \{0,1\}$ takes a number of the form $x.xxxx$, rounds it to the nearest integer (either 0, 1 or 2) and outputs the parity of that integer.

We claim that both these functions can be implemented by circuits of $\mathrm{poly}(m,k)$ size and degree at most k^3. This claim immediately implies the Clean operation, since we can just combine these circuits (this at most multiplies the degrees), and get a circuit of degree less than k^6 that given the $z_{i,j}$ inputs outputs the value $a' = \lfloor X/P \rceil \pmod 2$. We can then apply this circuit to the $Z_{i,j}$ ciphertexts to get an encryption of a' with noise at most $2^{n^{0.01}9k^{12}} \leq 2^{n^{0.3}}$ for our choice of $k = n^{1/1000}$. (The above calculations have a lot of slackness.)

For $ROUND$ the claim is straightforward (and even with degree $2\log k$)— any function mapping $\{0,1\}^{\ell}$ to $\{0,1\}$ can be expressed as a polynomial of at most 2^{ℓ} terms and degree ℓ.

For SUM the claim follows by somewhat tedious calculations showing that if you just follow the gradeschool algorithm for addition, and are careful to note that you never need more than $O(\log k)$ carry bits (because there are at most k non zero numbers) then the resulting circuit has degree at most k^2 or so.

Bottom line: We have obtained a fully homomorphic private key encryption!! This is already good enough for our applications such as cloud computing, zero knowledge, and multiparty computation (where one side generates the keys and the other just applied $EVAL$), but it's also very easy to transform it to a public key encryption.