19
Multiparty secure computation II: Construction using Fully Homomorphic Encryption

In the last lecture we saw the definition of secure multiparty computation, as well as the compiler reducing the task of achieving security in the general (malicious) setting to the passive (honest-but-curious) setting. In this lecture we will see how using fully homomorphic encryption we can achieve security in the honest-but-curious setting.[1] We focus on the two party case, and so prove the following theorem:

[1] This is by no means the only way to get multiparty secure computation. In fact, multiparty secure computation was known well before FHE was discovered. One common construction for achieving this uses a technique known as *Yao's Garbled Circuit*.

Theorem 19.1 — Two party honest-but-curious MPC. Assuming the LWE conjecture, for every two party functionality F there is a protocol computing F in the honest but curious model.

Before proving the theorem it might be worthwhile to recall what is actually the definition of secure multiparty computation, when specialized for the $k = 2$ and honest but curious case. The definition significantly simplifies here since we don't have to deal with the possibility of aborts.

Definition 19.2 — Two party honest-but-curious secure computation. Let F be (possibly probabilistic) map of $\{0,1\}^n \times \{0,1\}^n$ to $\{0,1\}^n \times \{0,1\}^n$. A *secure protocol for* F is a two party protocol such for every party $t \in \{1,2\}$, there exists an efficient "ideal adversary" (i.e., efficient interactive algorithm) S such that for every pair of inputs (x_1, x_2) the following two distributions are computationally indistinguishable:

- The tuple (y_1, y_2, v) obtained by running the protocol on inputs x_1, x_2, and letting y_1, y_2 be the outputs of the two parties and v be the *view* (all internal randomness, inputs, and messages received) of party t.

- The tuple (y_1, y_2, v) that is computed by letting $(y_1, y_2) = F(x_1, x_2)$ and $v = S(x_t, y_t)$.

That is, S, which only gets the input x_t and output y_t, can simulate all the information that an honest-but-curious adversary controlling party t will view.

19.1 CONSTRUCTING 2 PARTY HONEST BUT CURIOUS COMPUTATION FROM FULLY HOMOMORPHIC ENCRYPTION

Let F be a two party functionality. Lets start with the case that F is *deterministic* and that only Alice receives an output. We'll later show an easy reduction from the general case to this one. Here is a suggested protocol for Alice and Bob to run on inputs x, y respectively so that Alice will learn $F(x, y)$ but nothing more about y, and Bob will learn nothing about x that he didn't know before.

Protocol 2PC: (See Fig. 19.1)

- **Assumptions:** $(G, E, D, EVAL)$ is a fully homomorphic encryption scheme.
- **Inputs:** Alice's input is $x \in \{0,1\}^n$ and Bob's input is $y \in \{0,1\}^n$. The goal is for Alice to learn only $F(x, y)$ and Bob to learn nothing.
- **Alice->Bob:** Alice generates $(e, d) \leftarrow_R G(1^n)$ and sends e and $c = E_e(x)$.
- **Bob->Alice:** Bob computes define f to be the function $f(x) = F(x, y)$ and sends $c' = EVAL(f, c)$ to Alice.
- **Alice's output:** Alice computes $z = D_d(c')$.

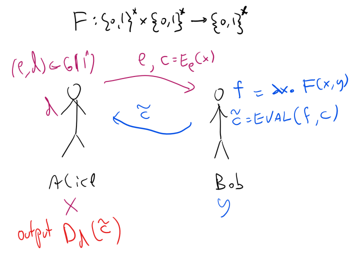

Figure 19.1: An honest but curious protocol for two party computation using a fully homomorphic encryption scheme with circuit privacy.

First, note that if Alice and Bob both follow the protocol, then indeed at the end of the protocol Alice will compute $F(x, y)$. We now claim that Bob does not learn anything about Alice's input:

Claim B: For every x, y, there exists a standalone algorithm S such that $S(y)$ is indistinguishable from Bob's view when interacting with Alice and their corresponding inputs are (x, y).

Proof: Bob only receives a single message in this protocol of the form (e, c) where e is a public key and $c = E_e(x)$. The simulator S will generate $(e, d) \leftarrow_R G(1^n)$ and compute (e, c) where $c = E_e(0^n)$. (As usual 0^n denotes the length n string consisting of all zeroes.) No matter what x is, the output of S is indistinguishable from the message Bob receives by the security of the encryption scheme. QED

(In fact, Claim B holds even against a *malicious* strategy of Bob- can you see why?)

We would now hope that we can prove the same regarding Alice's security. That is prove the following:

Claim A: For every x, y, there exists a standalone algorithm S such that $S(y)$ is indistinguishable from Alice's view when interacting with Bob and their corresponding inputs are (x, y).

P

At this point, you might want to try to see if you can prove Claim A on your own. If you're having difficulties proving it, try to think whether it's even true.

.
.
.
.
.
.
.
.
.
.

So, it turns out that Claim A is *not* generically true. The reason is the following: the definition of fully homomorphic encryption only requires that $EVAL(f, E(x))$ decrypts to $f(x)$ but it does *not* require that it hides the contents of f. For example, for every FHE, if we modify $EVAL(f, c)$ to append to the ciphertext the first 100 bits of the description of f (and have the decryption algorithm ignore this extra information) then this would still be a secure FHE.[2] Now we didn't exactly specify how we describe the function $f(x)$ defined as $x \mapsto F(x, y)$ but there are clearly representations in which the first 100 bits of the description would reveal the first few bits of the hardwired constant y, hence meaning that Alice will learn those bits from Bob's message.

[2] It's true that strictly speaking, we allowed $EVAL$'s output to have length at most n, while this would make the output be $n + 100$, but this is just a technicality that can be easily bypassed, for example by having a new scheme that on security parameter n runs the original scheme with parameter $n/2$ (and hence will have a lot of "room" to pad the output of $EVAL$ with extra bits).

Thus we need to get a stronger property, known as *circuit privacy*. This is a property that's useful in other contexts where we use FHE. Let us now define it:

Definition 19.3 — Perfect circuit privacy. Let $\mathcal{E} = (G, E, D, EVAL)$ be an FHE. We say that $\mathcal{E}$ satisfies *perfect circuit privacy* if for every (e, d) output by $G(1^n)$ and every function $f : \{0,1\}^\ell \rightarrow \{0,1\}$ of $poly(n)$ description size, and every ciphertexts $c_1, \ldots, c_\ell$ and $x_1, \ldots, x_\ell \in \{0,1\}$ such that c_i is output by $E_e(x_i)$, the distribution of $EVAL_e(f, c_1, \ldots, c_\ell)$ is identical to the distribution of $E_e(f(x))$. That is, for every $z \in \{0,1\}^*$, the probability that $EVAL_e(f, c_1, \ldots, c_\ell) = z$ is the same as the probability that $E_e(f(x)) = z$. We stress that these probabilities are taken only over the coins of the algorithms $EVAL$ and E.

Perfect circuit privacy is a strong property, that also automatically implies that $D_d(EVAL(f, E_e(x_1), \ldots, E_e(x_\ell))) = f(x)$ (can you see why?). In particular, once you understand the definition, the following lemma is a fairly straightforward exercise.

Lemma 19.4 If $(G, E, D, EVAL)$ satisfies perfect circuit privacy then if $(e, d) = G(1^n)$ then for every two functions $f, f' : \{0,1\}^\ell \rightarrow \{0,1\}$ of $poly(n)$ description size and every $x \in \{0,1\}^\ell$ such that $f(x) = f'(x)$, and every algorithm A,

$$|\Pr[A(d, EVAL(f, E_e(x_1), \ldots, E_e(x_\ell))) = 1] - \Pr[A(d, EVAL(f', E_e(x_1), \ldots, E_e(x_\ell))) = 1]| < negl(n). \tag{19.1}$$

P Please stop here and try to prove Lemma 19.4

The algorithm A above gets the *secret key* as input, but still cannot distinguish whether the $EVAL$ algorithm used f or f'. In fact, the expression on the lefthand side of (19.1) is equal to *zero* when the scheme satisfies perfect circuit privacy.

However, for our applications bounding it by a negligible function is enough. Hence, we can use the relaxed notion of "imperfect" circuit privacy, defined as follows:

Definition 19.5 — Statistical circuit privacy. Let $\mathcal{E} = (G, E, D, EVAL)$ be an FHE. We say that $\mathcal{E}$ satisfies *statistical circuit privacy* if for every (e, d) output by $G(1^n)$ and every function $f : \{0,1\}^\ell \rightarrow \{0,1\}$ of $poly(n)$ description size, and every ciphertexts $c_1, \ldots, c_\ell$ and $x_1, \ldots, x_\ell \in \{0,1\}$ such that c_i is output by $E_e(x_i)$, the distribution of $EVAL_e(f, c_1, \ldots, c_\ell)$ is equal up to $negl(n)$ total variation distance to the distribution of $E_e(f(x))$.

That is,

$$\sum_{z \in \{0,1\}^*} |\Pr[EVAL_e(f, c_1, \ldots, c_\ell) = z] - \Pr[E_e(f(x)) = z]| < negl(n) \tag{19.2}$$

where once again, these probabilities are taken only over the coins of the algorithms $EVAL$ and E.

If you find Definition 19.5 hard to parse, the most important points you need to remember about it are the following:

- Statistical circuit privacy is as good as perfect circuit privacy for all applications, and so you can imagine the latter notion when using it.
- Statistical circuit privacy can easier to achieve in constructions.

(The third point, which goes without saying, is that you can always ask clarifying questions in class, Piazza, sections, or office hours...)

Intuitively, circuit privacy corresponds to what we need in the above protocol to protect Bob's security and ensure that Alice doesn't get any information about his input that she shouldn't have from the output of $EVAL$, but before working this out, let us see how we can construct fully homomorphic encryption schemes satisfying this property.

19.2 ACHIEVING CIRCUIT PRIVACY IN A FULLY HOMOMORPHIC ENCRYPTION

We now discuss how we can modify our fully homomorphic encryption schemes to achieve the notion of circuit privacy. In the scheme we saw, the encryption of a bit b, whether obtained through the encryption algorithm or $EVAL$, always had the form of a matrix C over $\mathbb{Z}_q$ (for $q = 2^{\sqrt{n}}$) where $Cv = bv + e$ for some vector e that is "small" (e.g., for every i, $|e_i| < n^{polylog(n)} \ll q = 2^{\sqrt{n}}$). However, the $EVAL$ algorithm was *deterministic* and hence this vector e is a function of whatever function f we are evaluating and someone that knows the secret key v could recover e and then obtain from it some information about f. We want to make $EVAL$ probabilistic and lose that information, and we use the following approach

To kill a signal, drown it in lots of noise

That is, if we manage to add some additional random noise e' that has magnitude much larger than e, then it would essentially "erase" any structure e had. More formally, we will use the following lemma:

Lemma 19.6 Let $a \in \mathbb{Z}_q$ and $T \in \mathbb{N}$ be such that $aT < q/2$. If we let X be the distribution obtained by taking $x (\mod q)$ for an integer x chosen at random in $[-T, +T]$ and let X' be the distribution obtained by taking $a + x (\mod q)$ for x chosen in the same way, then

$$\sum_{y \in \mathbb{Z}_q} |\Pr[X = y] - \Pr[X' = y]| < |a|/T \tag{19.3}$$

Proof. This has a simple "proof by picture": consider the intervals $[-T, +T]$ and $[-T + a, +T + a]$ on the number line (see Fig. 19.2). Note that the symmetric difference of these two intervals is only about a a/T fraction of their union. More formally, X is the uniform distribution over the $2T + 1$ numbers in the interval $[-T, +T]$ while X' is the uniform distribution over the shifted version of this interval $[-T + a, +T + a]$. There are exactly $2|a|$ numbers which get probability zero under one of those distributions and probability $(2T + 1)^{-1} < (2T)^{-1}$ under the other.

■

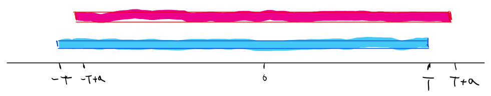

Figure 19.2: If $a \ll T$ then the uniform distribution over the interval $[-T, +T]$ is statistically close to the uniform distribution over the interval $[-T + a, +T + a]$, since the statistical distance is proportional to the event (which happens with probability a/T) that a random sample from one distribution falls inside the symmetric difference of the two intervals.

We will also use the following lemma:

Lemma 19.7 If two distributions over numbers X and X' satisfy $\Delta(X, X') = \sum_{y \in \mathbb{Z}} |\Pr[X = x] - \Pr[Y = y]| < \delta$ then the distributions X^m and X'^m over m dimensional vectors where every entry is sampled independently from X or X' respectively satisfy $\Delta(X^m, X'^m) \leq m\delta$.

P

We omit the proof of Lemma 19.7 and leave it as an exercise to prove it using the hybrid argument. We will actually only use Lemma 19.7 for distributions above; you can obtain intuition for it by considering the $m = 2$ case where we compare the rectangles of the forms $[-T, +T] \times [-T, +T]$ and $[-T + a, +T + a] \times [-T + b, +T + b]$. You can see that their union has size roughly $4T^2$ while their symmetric difference has size roughly $2T \cdot 2a + 2T \cdot 2b$, and so if $|a|, |b| \leq \delta T$ then the symmetric difference is roughly a 2δ fraction of the union.

We will not provide the full details, but together these lemmas show that $EVAL$ can use bootstrapping to reduce the magnitude of the noise to roughly $2^{n^{0.1}}$ and then add an additional random noise of roughly, say, $2^{n^{0.2}}$ which would make it statistically indistinguishable from the actual encryption. Here are some hints on how to make this work: the idea is that in order to "re-randomize" a ciphertext C we need a very noisy encryption of zero and add it to C. The normal encryption will use noise of magnitude $2^{n^{0.2}}$ but we will provide an encryption of the secret key with smaller magnitude $2^{n^{0.1}/polylog(n)}$ so we can use bootstrapping to reduce the noise. The main idea that allows to add noise is that at the end of the day, our scheme boils down to LWE instances that have the form (c, σ) where c is a random vector in $\mathbb{Z}_q^{n-1}$ and $\sigma = \langle c, s \rangle + a$ where $a \in [-\eta, +\eta]$ is a small noise addition. If we take any such input and add to σ some $a' \in [-\eta', +\eta']$ then we create the effect of completely re-randomizing the noise. However, completely analyzing this requires non-trivial amount of care and work.

19.3 BOTTOM LINE: A TWO PARTY HONEST BUT CURIOUS TWO PARTY SECURE COMPUTATION PROTOCOL

We can now prove the following theorem:

Theorem 19.8 — Two party . If $(G, E, D, EVAL)$ is a statistically circuit private fully homomorphic encryption then Protocol 2PC is a secure two party computation protocol with respect to honest but curious adversaries.