CS 294 – Special Topics in Cryptography April 5, 2016

Lecture 21: Applications of Fully Homomorphic Encryption

Instructor: Sanjam Garg *Scribe: Urmila Mahadev*

1 Fully Homomorphic Encryption

A fully homomorphic encryption scheme consists of functions KeyGen, Enc, Dec and Eval such that:

- $(pk, sk) \leftarrow \textbf{KeyGen}(1^k)$
- $e \leftarrow \textbf{Enc}(pk, m)$
- $m \leftarrow \textbf{Dec}(e, sk)$

For a circuit C, assume C' is the circuit output by $\textbf{Eval}_{pk}(C)$. Then $C'(e)$ is an encryption of $C(m)$ (where e is an encryption of m). The size of $C'(e)$ is independent of the size of C. We will cover 3 applications of fully homomorphic encryption:

1. Non interactive zero knowledge: the size of the proof is the size of the witness + $poly(k)$
2. Multiparty computation: communication cost grows polynomial in input length and security parameter
3. Delegation schemes

2 Non Interactive Zero Knowledge

Recall the setup. We have algorithms (K, P, V), where $\sigma \leftarrow K(1^k)$, $\pi \leftarrow P(\sigma, x, w)$ and $0/1 \leftarrow V(\sigma, x, \pi)$.

2.1 Initial Solution

We will start with a solution which is not yet linear (the size of the proof is $poly(|W|, k)$) but gives an idea of the final solution. Let

$$C_x(w) = R(x, w)$$

be the relationship which is being tested. Let $C' = \textbf{Eval}(C_x)$. The prover generates pk, sk and encrypts the witness w to obtain e. The prover then computes $e' = C'(e)$ and generates

$$\pi \leftarrow P(\sigma, (pk, e'), sk)$$

using a NIZK protocol. The proof sent to the verifier is

$$\pi' = (k, e, \pi)$$

In other words, the prover uses a NIZK protocol to prove that the ciphertext e' is a valid encryption of 1, and the verifier can check that $e' = C'(e)$.

2.2 Linear Solution

The description of the linear solution follows. Let

$$C_{x,u}(\cdot) = R(x, u \oplus G(\cdot, |w|)) \text{ and } C'(x,u) = \mathbf{Eval}(C_{x,u})$$

where G is a pseudorandom generator. The idea here is to use the pseudorandom generator to hide the witness. If the prover sets

$$u = w \oplus G(s, |w|)$$

and s is hidden, the prover can reveal u to the verifier. This is the trick that makes the solution linear in the size of the witness. The prover can then homomorphically encrypt s (obtaining $\bar{s}$) and send $\bar{s}$ to the verifier, who can now compute $C'_{x,u}(\bar{s})$. The prover also provides a NIZK proof that the seed s is correctly encrypted and that $v = C'_{x,u}(\bar{s})$ decrypts to 1. The relationship that we are testing with the NIZK proof system is:

$$R^F = \{((pk, \bar{s}, v), (\rho, s, \bar{r})) :$$

$$\rho \in \{0,1\}^{l_{\text{KeyGen}}} \wedge (pk, sk) = \mathbf{KeyGen}(1^k, \rho) \wedge \bar{r} \in (\{0,1\}^{l_{\text{Enc}}})^{|s|} \wedge \bar{s} = \mathbf{Enc}(pk, (s, \bar{r})) \wedge \mathbf{Dec}(v, sk) = 1\}$$

The size of the proof is $|w| + poly(k)$.

3 Multiparty Computation

Two parties, P_1 and P_2, hold inputs x_1 and x_2. The goal is for each to be able to compute $f(x_1, x_2)$ and learning nothing more about the other party's input. P_1 homomorphically encrypts his input after generating (pk, sk):

$$e = \mathbf{Enc}(pk, x_1)$$

Let $C_{x_2}(\cdot) = f(\cdot, x_2)$, $C'_{x_2} = \mathbf{Eval}(C_{x_2})$. Given e and pk, P_2 can compute

$$e' = C'_{x_2}(e)$$

P_1 and P_2 then run a multiparty computation where the function f is **Dec** and the inputs are $(x_1, x_2) = (sk, e')$. Note that the decryption circuit is independent of the input length. The size of ciphertext e is $poly(|X_1|, k)$, e' is $poly(|f(x_1, x_2)|, k)$ and $|sk| = poly(k)$.

4 Delegation Schemes

In this setting, a delegator wants to assign work to a worker. The delegator gives the worker x and asks the worker to compute $f(x)$, for some function f. We are working in the online/offline setting, in which the delegator is allowed to do work proportional to the size of f (in the offline phase), but independent of x.

4.0.1 Known Input Distribution

Assume that in the offline phase, the delegator knows the distribution E from which x is drawn. The following protocol achieves a soundness parameter of $\frac{1}{2}$. In the offline phase, the delegator draws a random value r from the distribution E and computes $f(r)$. In the online phase, the delegator sends both x and r to the worker in a random order and asks the prover to apply f to both inputs. To verify, the delegator verifies that the worker computed $f(r)$ correctly. Since the worker cannot distinguish between x and r, he must compute $f(x)$ correctly with probability at least $\frac{1}{2}$ in order to pass the verifier's test.

4.0.2 Unknown Input Distribution

In this case, the delegator does not know the distribution E from which x is drawn. If he chooses a random value r from a different distribution F in the offline phase, the worker may be able to distinguish between r and x in the online phase.

Instead, the delegator homomorphically encrypts both r and x, obtaining $\bar{r}$ and $\bar{x}$, before sending both to the worker in a random order. The worker can then apply $f' = \mathbf{Eval}(f)$ on top of the encryption and send the results back to the delegator, who decrypts and accepts if $f(r)$ is correct. This protocol is not sound; the delegator can cheat inside the encryption. To do this, the delegator applies $g' = \mathbf{Eval}(g)$ instead of f'. The function g first checks which distribution the input is from, and then applies the function f only if the distribution is F.

To protect against this type of cheating, the delegator computes $f'(\bar{r})$ instead of $f(r)$ (in the offline phase). Now instead of decrypting the worker's answer and checking it, the delegator just checks if the worker's answer is $f'(\bar{r})$.

Intuitively, this is sound because the function that the delegator is checking is now deterministic, and the inputs given to the worker are computationally indistinguishable. To prove soundness, we can show that if the worker passes the verifier's test and is incorrect on f with probability more than $\frac{1}{2}$, he can break the security of the homomorphic encryption scheme. To amplify soundness, parallel repetition can be used.

This protocol is secure only for one time use. To make it reusable, another independent layer of homomorphic encryption is added in the online phase. To show soundness, we can argue that since the encryption keys are independent, you could have encrypted 0 instead of r.