MIT 6.5620/6.875/18.425: Cryptography and Cryptanalysis **Due: Dec. 4, 2023**

Problem Set 6

Total Number of Points: 60.

Collaboration Policy: Collaboration is allowed and encouraged in small groups of at most three students. You are free to collaborate in discussing answers, but you must write up solutions on your own and must specify in your submission the names of any collaborators. Do not copy any text from your collaborators; the writeup must be entirely your work. Do not write down solutions on a board and copy them verbatim into LaTeX; again, the writeup must be entirely your own words and your own work and should demonstrate a clear understanding of the solution. Additionally, you may make use of published material, provided that you acknowledge all sources used. Of course, scavenging for solutions from prior years is forbidden.

Problem 1. Hiding the Circuits in FHE (20 points)

Recall the definition of what it means for a fully homomorphic encryption (FHE) scheme to be compact and to be circuit-hiding. In this problem, we will think of FHE schemes capable of evaluating a class of Boolean circuits $\mathcal{C}$ computing functions f with k-bit inputs and a 1-bit output, for any $k = \mathsf{poly}(n)$.

Compactness. We say that a secret-key FHE scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ has compact ciphertexts if there is a polynomial p such that for any $C \in \mathcal{C}$, the bit-length of $\mathsf{Eval}(\mathsf{ek}, C, \mathsf{Enc}(\mathsf{sk}, x))$ is at most $p(n)$. That is, the size of the evaluated ciphertext depends neither on the input length $|x|$ nor on the circuit size $|C|$.

Function Hiding. We say that a secret-key FHE scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ is function-hiding if for every PPT adversary A, any input x, and any pair of circuits C_0 and C_1 of the same size with $C_0(x) = C_1(x)$:

$$\mathbf{Pr}[b' = b : (\mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^n); \psi \leftarrow \mathsf{Enc}(\mathsf{sk}, x); b \leftarrow \{0,1\}; c_b \leftarrow \mathsf{Eval}(\mathsf{ek}, C_b, \psi); b' \leftarrow A(\mathsf{sk}, \mathsf{ek}, \psi, c_b)] \leq 1/2 + \mathsf{negl}(n)$$

(a) (1 points) Show that compactness does not necessarily imply function hiding. For this problem, you can assume the existence of an FHE scheme that satisfies compactness.

(b) (1 points) Show that function hiding does not necessarily imply compactness. For this problem, you can assume the existence of an FHE scheme that satisfies function hiding.

(c) (18 points) Assume the existence of a function hiding (but not necessarily compact) secret-key FHE scheme $(\mathsf{Gen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1, \mathsf{Eval}_1)$ and that of a compact (but not necessarily function-hiding) secret-key FHE scheme $(\mathsf{Gen}_2, \mathsf{Enc}_2, \mathsf{Dec}_2, \mathsf{Eval}_2)$. Construct a secret-key FHE scheme that is (IND-CPA secure) and both compact and function hiding.

For the purposes of this problem, we only require that the decryption algorithm decrypts evaluated ciphertexts (as opposed to fresh ciphertexts produced by the encrytion algorithm) correctly.

Problem 2. Perfectly Binding Hash Functions: An Oxymoron? (20 points)

A hash function $\mathsf{H} : \{0,1\}^n \to \{0,1\}^m$ compresses its n-bit input into an m-bit output for some $m < n$. A perfectly binding function $\mathsf{H} : \{0,1\}^n \to \{0,1\}^m$ is injective, i.e. given $\mathsf{H}(x)$, x is uniquely determined. Could one have a perfectly binding hash function? Clearly not!

In this problem, we will construct hash functions that nevertheless achieve variants of perfect binding.

A *bit-wise perfectly binding* hash function is a pair of algorithms $(\mathsf{Gen}, \mathsf{H})$ where

- $k \leftarrow \mathsf{Gen}(1^n, i)$ takes as input an integer $i \in [n]$ and outputs a public hashing key k;
- $y = \mathsf{H}(k, x)$ is a deterministic poly-time algorithm that takes an n-bit input x and produces an m-bit output.

We require the following two properties:

Index Hiding: For every pair of indices $i_0, i_1 \in [n]$, the hash keys $k_0 \leftarrow \mathsf{Gen}(1^n, i_0)$ and $k_1 \leftarrow \mathsf{Gen}(1^n, i_1)$ are computationally indistinguishable.

Bit-wise Perfectly Binding: For any $i \in [n]$, $k \leftarrow \mathsf{Gen}(1^n, i)$, and $x, x' \in \{0,1\}^n$, if $\mathsf{H}(k, x) = \mathsf{H}(k, x')$, then $x[i] = x'[i]$.

Construct a bit-wise perfectly binding hash function, assuming the hardness of the quadratic residuosity problem. Prove that your construction is correct.

Problem 3. Two-Server PIR with Pre-Processing (20 Points)

Recall the notion of a private information retrieval (PIR) scheme. In this problem, we will consider the PIR with two servers, where each server has a copy of the database $x \in \{0,1\}^n$, and a user wants to know a single bit of x. The two servers *do not communicate* with each other and will answer users' queries separately, and the user does not want the servers to know which bit she wants. Consider the following two-server PIR protocols between user U and servers S_0 and S_1:

- The user chooses a random subset $T \subseteq \{1, 2, \ldots, n\}$, including each value independently with probability $1/2$. (Note that T can be represented as an n-bit string w_T by letting the bit in position j represent whether or not $j \in T$.)
- The user calculates the set $T \oplus i$, which is $T \cup \{i\}$ if T does not contain i, and $T \setminus \{i\}$ if T contains i.
- The user sends w_T to the server S_0 and $w_{T \oplus i}$ to the server S_1.
- Given a set, each server sends back the XOR of all the bits of x in that set.
- The user calculates the XOR of the two responses from S_0 and S_1.

(a) (4 points) Prove that the scheme is *information-theoretically private*; that is, for any two different values i and i', the view of each server S_0 and S_1 is perfectly indistinguishable, even if the servers are computationally unbounded. (It will be the case that the views of S_0 and S_1 together reveal i, but that is OK. Our assumption is that of non-collusion, that is S_0 and S_1 do not ever collude and put their views together.)

(b) (8 points) Unfortunately, the above protocol has a total communication complexity of $2(n+1)$ bits, because the user U sends an n-bit message and receives a one-bit response from each of the two servers. Note that there is a trivial, private protocol with communication complexity n: one of the servers simply sends the entire string x to the user U. We would like to reduce the amount of communication to significantly below n. Alter the scheme from part (a) in such a way as to reduce the amount of communication to $o(n)$.

(Hint: It might be useful to reorganise the database into a $\sqrt{n} \times \sqrt{n}$ table.)

(c) (8 points) Unfortunately, in the above protocol each server needs to look at the entire database to answer every query. That's too computationally expensive.

Your goal is to alter the scheme from (b) in such a way as to reduce the computation *per query* to $O(n/\log n)$, while keeping the communication complexity sublinear. In

order to do that, you are allowed to preprocess the database into a polynomially larger string in an *offline* phase, that is before receiving any queries. This preprocessing could be computationally expensive, i.e., it could take polynomial time in n. However, the computation in the *online* phase, after receiving a query from the client, should take time $o(n)$. In particular, **a solution that achieves online communication complexity $O(n/\log n)$ and $\mathsf{poly}(n)$ offline computation complexity will receive full points.**