CS 395T: Topics in Cryptography **Spring 2022**

Problem Set 2

Due: May 6, 2022 at 11:59pm (Submit on Gradescope) **Instructor:** David Wu

Instructions. You **must** typeset your solution in LaTeX using the provided template:

`https://www.cs.utexas.edu/~dwu4/courses/sp22/static/homework.tex`

You must submit your problem set via Gradescope (accessible through Canvas). **Last updated:** April 20, 2022.

Collaboration Policy. You may discuss your general approach with other students, but you may not share written documents. You should not search online for solutions to these problems. If you do consult external sources, you must cite them in your submission. You must include the names of all of your collaborators with your submission. Refer to the official course policies for the full details.

Instructions. We will add problems to the problem set throughout the semester (roughly 1 problem each week), with the last problem added at least a week in advance of the due date. All problems are weighted equally. You should submit solutions to at least 70% of the problems (rounded down). If you submit solutions to more than 70% of the problems, we will drop the lowest-scoring problems when computing your final homework score. If you scribed *at least* two lectures during the course of this semester, you only need to submit solutions to at least 50% of the problems (rounded down).

Problem 1: ABE Key Delegation. In an ABE scheme, secret keys are associated with functions $f\colon \{0,1\}^\ell \to \{0,1\}$ and ciphertexts are associated with an attribute-message pair (x,μ) where $x \in \{0,1\}^\ell$ and $\mu \in \{0,1\}$. A secret key sk_f for a function f successfully decrypts a ciphertext $\mathsf{ct}_{x,\mu}$ associated with (x,μ) if $f(x) = 0$ (using the conventions for the construction from lecture). We say that the scheme supports key delegation if the holder of a secret key sk_f can generate a key for the function $(f \wedge g)$ where $(f \wedge g)(x) = 0$ if and only if $f(x) = 0 = g(x)$. Show how to modify the key-generation algorithm for the ABE scheme from class so that it additionally supports key delegation. The key-delegation algorithm should only take the *master public key* mpk, the secret key sk_f associated with f, and the function g as input. It does *not* know the master secret key. Prove the correctness of your key-delegation procedure. You do *not* need to re-prove security of your modified ABE scheme, but the same type of argument as used in lecture should still apply to your modified scheme.

Problem 2: Predicate Encryption and Functional Encryption. In class, we showed how to construct a weak attribute-hiding predicate encryption scheme from lattices. In this model, we require that the attribute associated with a ciphertext be hidden only if the adversary does not possess any keys that allow it to decrypt the ciphertext. We say a predicate encryption scheme satisfies strong attribute-hiding if the attribute remains hidden even given decryption keys that successfully decrypt the ciphertext. Namely, in the semantic security game, the adversary's challenge is a tuple (x_0, μ_0, x_1, μ_1). The tuple is admissible if one of the following two properties hold:

- $\mu_0 = \mu_1$ and $f(x_0) = f(x_1)$ for all key queries f the adversary makes; or
- $f(x_0) = 1 = f(x_1)$ for all key queries f the adversary makes.

Shows that a strong attribute-hiding predicate encryption scheme for general circuit predicates implies a functional encryption scheme for general circuits. Your functional encryption scheme should support circuits with *multi-bit* outputs (e.g., $C\colon \{0,1\}^n \to \{0,1\}^m$) and be collusion resistant (i.e., the adversary can ask for any polynomial number of keys). You only need to provide a high-level sketch of the security proof of your construction.

Problem 3: NIZK Proofs of Knowledge. Recall that in a proof of knowledge, the soundness property is replaced by a stronger knowledge property, which requires that for every (possibly malicious) prover that convinces the verifier of a statement with probability $\varepsilon > 0$, there exist an efficient extractor that extracts the witness from the prover with probability $\varepsilon - \mathsf{negl}(\lambda)$. In the common reference string model (CRS), the extractor gets to sample the CRS, as long as it samples one that is computationally indistinguishable from an honestly-generated CRS. Show that you can combine any NIZK for NP in the CRS model together with a public-key encryption scheme to obtain a NIZK proof of knowledge for NP in the CRS model. Describe your construction and then give an *informal* sketch of the extractability and zero-knowledge properties.

Problem 4: Statistical Correlation Intractability from LWE. Recall the correlation-intractable hash function based on SIS. The hash key $\mathsf{hk} = (\mathbf{b}, \mathbf{B}_1, \ldots, \mathbf{B}_\ell)$ where $\mathbf{b} \xleftarrow{R} \mathbb{Z}_q^n$ and $\mathbf{B}_i \xleftarrow{R} \mathbb{Z}_q^{n \times k}$. The hash function is then defined as

$$\mathsf{H}(\mathsf{hk}, x) := \mathbf{G}^{-1}(\mathbf{b} + \mathbf{B}_{U_x}\mathbf{G}^{-1}(\tilde{\mathbf{g}})) \text{ where } \mathbf{B}_{U_x} \leftarrow [\mathbf{B}_1 \mid \cdots \mid \mathbf{B}_\ell] \cdot \mathbf{H}_{U_x},$$

and $\tilde{\mathbf{g}}$ is the vector consisting of the elements of $\mathbf{G}$. Show how to modify hk so the construction is *statistically* correlation-intractable for any fixed and efficiently-computable function f. Namely, show how to sample a hash key hk such that there does not *exist* any x where $f(x) = \mathsf{H}(\mathsf{hk}, x)$. The hash key hk should be computationally indistinguishable (under the LWE assumption) from the real hash key (where all components are uniform).[1] Prove both the statistical correlation intractability property and the hash key indistinguishability properties.

Problem 5: Correlation Intractability for Polynomial-Size Sets. In class, we showed how to construct a correlation-intractable hash function for any efficiently-searchable relation $\mathcal{R}$. Namely, the relation $\mathcal{R}$ has the property that for every instance x, there is a unique $y \leftarrow f(x)$ such that $\mathcal{R}(x, y) = 1$. Suppose instead that for every input x, there exists a set S_x of size $|S_x| = d$ where $\mathcal{R}(x, y) = 1$ for every $y \in S_x$ (and 0 otherwise). Here $d = \mathrm{poly}(\lambda)$. Suppose moreover that there exists an efficiently-computable function g that takes as input x and outputs the set S_x. Show how to use a correlation-intractable hash function for an efficiently-searchable relation to construct a correlation-intractable hash function for $\mathcal{R}$. As usual, the hash key for your construction should not depend on g (but could depend on the size/depth of g). Prove the security of your construction.

Problem 6: Multi-Key FHE and Homomorphic Secret Sharing. In class, we showed how to obtain a 2-party HSS scheme from a multi-key FHE scheme. Explain why the approach we described in class does not yield an HSS scheme for more than 2 parties. (**Note:** We can construct n-party HSS from multi-key FHE by boosting the 2-party construction, but this is beyond the scope of this question).

Optional Feedback. Please answer the following *optional* questions to help us design future problem sets. You do not need to answer these questions. However, we do encourage you to provide us feedback on how to improve the course experience.

(a) How long did you spend on this problem set?

(b) What was your favorite problem on this problem set? Why?

(c) What was your least favorite problem on this problem set? Why?

(d) Do you have any other feedback for this problem set?

(e) Do you have any other feedback on this course?

[1] We often refer to this property as *somewhere statistical correlation intractability.*