Homework 3

Problem 1: Annihilating More General Branching Programs. In this problem, we will see a nice way to extend the attack that we saw in class to a wider class of programs (recall that in class, the attack only worked for "trivial" branching programs). We will focus on the step of devising an annihilating polynomial for particular programs. Parts (a), (b), (c), and (e) are required. (d) is optional and extra credit.

Part (a). Annihilating constant-degree polynomials. Fix a constant d. Suppose there are n variables $X_1, \ldots, X_n$, t polynomials in $X_1, \ldots, X_n$, $p_i(X_1, \ldots, X_n)$ for $i = 1, \ldots, t$, where each p_i is a polynomial of degree d. $t > n$ is sufficient to guarantee an annihilating polynomial for the p_i. However, if $t \gg n$, it can potentially be much easier to compute an annihilating polynomial.

Show how, in polynomial time, to compute an annihilating polynomial for the p_i, provided t is sufficiently large. How large does t need to be?

Hint: use linear algebra.

Part (b). Let BP be an arbitrary branching program computing a function P. Consider the *single input* version of our first obfuscator (without the extra block consisting of a random matrix), where each column of BP reads only a single bit.

For an element x such that $P(x) = 1$ (so that zero testing gives 0), let T_x be the element obtained from the zero testing procedure. Let $\bar{x}$ be the bitwise complement of x.

Compute the product $T_x \times T_{\bar{x}}$. Notice something interesting happening?

Part (c). Combine Parts (a) and (b) to show how to efficiently compute annihilating polynomials for general programs (in the single input setting).

Part (d) (Optional — this part is actually an open question, and extra credit will be given to anyone who solves it). Generalizing to dual-input programs. Construct family F of subsets of $\{0,1\}^n$ such that

- There are exponentially many subsets in F

- There is a constant c (independent of n) such that for every $S \in F$, $|S| = c$.
- For every $S \in F$, the product $\prod_{x \in S} T_x$ will cause the α's to combine as in Part (b), even for dual input branching programs.

Use this family F to show how to efficiently compute annihilating polynomials for general programs in the dual-input setting.

Alternatively, show that such a family F is impossible.

Part (e). The above attacks all operate in the weak ideal multilinear map model discussed in class. We showed a modified obfuscator that is secure in this model. Yet, the attack above apparently works for all branching programs. **Explain this apparent contradiction.**

Note: don't focus on dual input vs single input — this is not the issue. You can assume that a positive solution to Part (d) has been found