Information Security Lab - AS20

Reductions

Handout: Oct 16, 2020 12:00 AM

Due: Nov 2, 2020 3:30 PM

Exercise 3a -- Rerandomization of DDH Challenges

Open Task

Exercise 3a - Rerandomizability of DDH

Again, we consider $\mathbb{G}$ to be a cyclic group of prime order p. In this assignment, we want to showcase the fact that DDH is rerandomizable. Given a DDH challenge (g, g^x, g^y, g^z), we want an algorithm that can efficiently sample tuples $(g, g^{d_1}, g^{e_1}, g^{f_1}), \ldots, (g, g^{d_n}, g^{e_n}, g^{f_n})$, such that:

- $(g, g^{d_1}, g^{e_1}, g^{f_1}), \ldots, (g, g^{d_n}, g^{e_n}, g^{f_n})$ are independent DDH tuples when (g, g^x, g^y, g^z) is a DDH tuple. (referred to as honestly generated DDH tuples in the source code.)
- $(g, g^{d_1}, g^{e_1}, g^{f_1}), \ldots, (g, g^{d_n}, g^{e_n}, g^{f_n})$ are independent uniformly random tuples when (g, g^x, g^y, g^z) is a uniformly random tuple:

If DDH is hard, this means that $D_{\text{n-DDH}}$ and $D_{\text{n-random}}$ defined below are computationally indistinguishable:

$$D_{\text{n-DDH}} = \{(g, (g^{d_i}, g^{e_i}, g^{d_i e_i})_{i \in [n]}) : ((d_i, e_i)_{i \in [n]}) \leftarrow_r \mathbb{Z}_p^{2n}\}.$$

$$D_{\text{n-random}} = \{(g, (g^{d_i}, g^{e_i}, g^{f_i})_{i \in [n]}) : ((d_i, e_i, f_i)_{i \in [n]}) \leftarrow_r \mathbb{Z}_p^{3n}\}$$

Your task is to implement the `rerandomize` method of the `Solution3a` class:

```
public class Solution3a implements IDDHRerandomizer {
    @Override
    public DDHChallenge[] rerandomize(int numberOfGames, DDHChallenge ddhChallenge, SecureRandom RNG) {
        DDHChallenge[] rerandomizedChallenges = new DDHChallenge[numberOfGames];
         /* your code */
        return rerandomizedChallenges;
    }
}
```

Important Use only the randomness from the RNG parameter. Your code should rerandomize a given DDH challenge. Given a challenge tuple (g, g^x, g^y, g^z), it should return tuples of the form (g, g^d, g^e, g^f) such that $f = ed$ if $z = xy$. If z was sampled uniformly at random and independently from x and y, then f should also be distributed uniformly at random and independently from d and e. The exponents d and e should always be distributed uniformly at random and independently of each other.

This rerandomization method must be state-less. Its output value should only depend on `ddhChallenge` and the randomness given by `RNG`.

Different output tuples should be independently distributed of each other

This method must not use other randomness from other sources. It is expected that two calls of this method return the same values when they are given the same ddhChallenge and RNG where in both cases `RNG` has been initialized with the same seed.

Evaluation

In order for your solution to be valid, your rerandomization algorithm should always correctly map DDH tuples to DDH tuples and uniformly random tuples to uniformly random tuples, with the following exception: It might be the case that rerandomizing an uniformly random sample yields a DDH tuple with probability 1/p (the same probability that a uniformly random tuple is a DDH one).