Information Security Lab - AS20

Reductions

Handout: Oct 16, 2020 12:00 AM

Due: Nov 2, 2020 3:30 PM

Exercise 2 -- Ind-CPA Security of El Gamal PKE

Open Task

Exercise 2 - Hardness of DDH implies ElGamal PKE is IND-CPA-secure

Again, we consider $\mathbb{G}$ to be a cyclic group of prime order p. In this assignment, we want to show that if DDH holds, then the ElGamal PKE scheme is IND-CPA-secure. Your task is to implement the `decideDDH` method in the `Solution2` java class:

```
public class Solution2 implements IDDHIndCPAElgamalReduction {

    @Override
    public boolean decideDDH(DDHChallenge ddhChallenge, IElgamalIndCPAAdversary adversary) {
        //your code here
        return true if the ddhChallenge is an honestly generated DDH tuple, false otherwise;
    }

}
```

As in Exercise 1, you will have to initialize and invoke the `IElgamalIndCPAAdversary`. Recall that the DDH problem means you are given a generator g and you are asked to distinguish between (g, g^x, g^y, g^{xy}) and (g, g^x, g^y, g^z), where $x, y, z \leftarrow_r \mathbb{Z}_p$. If the challenge is an honestly generated DDH tuple (i.e. of the form (g, g^x, g^y, g^{xy})), you should output true. Otherwise, output false.

As in Exercise 1, your reduction is only allowed to call the method `adversary.solveIndCPAChallenge(ciphertext)` at most **once** (meaning that we ask that your reduction is tight).

The `init` and `solve` methods of the adversary are defined similarly to Exercise 1. One key difference is that now we have a new method:

```
public CandidateMessagePair<IGroupElement> getCandidateMessages();
```

This method should be called after you have already called `init`. It models the stage in the IND-CPA game when the adversary picks its plaintext challenges. Here, the adversary will generate two plaintexts (of type `IGroupElement`) and return them to your reduction. If `init` has not been called before, `getCandidateMessages` will return null. Calling this method will cause this instance to save copies of both messages. Calling this method a second time will cause this instance to overwrite its last saved values.

The output value is a `CandidateMessagePair` which contains two `IGroupElements`. The first entry is message_0 (the first message the adversary has chosen) and the second one is message_1 (the second message the adversary has chosen).

To retrieve the adversary's guess on whether message_0 or message_1 have been encrypted, you should run the following method of the adversary:

```
    public int solveIndCPAChallenge(ElgamalCiphertext ciphertext);
```

You should only call this method after you have called init and getCandidateMessages and supplied a ciphertext of an instance of ElGamal's PKE scheme whose public key and group generator are consistent with the parameters of your init call.

If the ciphertext is a valid encryption of message_0, the first message returned by getCandidateMessages, then this method should return 0.

If ciphertext is a valid encryption of message_1, the second message returned by getCandidateMessages, then this method should return 1.

Note If neither of the above cases does occur, then this method will return 0 or 1 (without any guarantees on which value will be returned). If init or getChallengeMessages has never been called before, this method will also return 0 or 1 (without any guarantees which value will be returned).

Evaluation

When you submit your solution, our tests will run the reduction 350 times per adversary. Your reduction should win, on average, 75% of the games (meaning that the theoretical maximum advantage of your reduction should be 1/2). Your reduction passes our test if the advantage we measure is at least 25% (i.e. if it wins at least 62,5% of all games against each adversary).