Reductions

Exercise 1. *RSA to Factoring*

1. Show (formally) that the problem RSA reduces to the factoring problem, both introduced in the second lecture. In other words, show that if there exists an algorithm $\mathcal{A}$ which solves the factoring problem with non-negligible probability, one can construct an algorithm $\mathcal{B}$ which solves the RSA problem with non-negligible probability.

   Hint: Think about what information you need to solve RSA. The extended Euclidean algorithm applied to e and $\varphi(N)$ might be of use.

Solution: Recall the definition of the **factoring problem**: Let p, q be two random distinct primes, defining the λ-bit integer N. Given as input N, the problem asks to find p and q.

Recall the definition of the **RSA problem**: Let p, q be two random distinct primes, defining the λ-bit integer N. Let φ be Euler's totient function. Further, sample $e \leftarrow (\mathbb{Z}/\varphi(N)\mathbb{Z})^\times$ and sample $y \leftarrow (\mathbb{Z}/N\mathbb{Z})^\times$. Given as input (N, e, y), the problem asks to find x such that $x^e = y \bmod N$.

I Description: Let $\mathcal{A}$ be a PPT algorithm solving the factoring problem with probability $> \mathsf{negl}(\lambda)$. Our goal is to construct a PPT algorithm $\mathcal{B}$ solving the RSA problem with probability $> \mathsf{negl}(\lambda)$.

Let (N, e, y) be the input given to $\mathcal{B}$. The algorithm $\mathcal{B}$ takes N and gives it as input to $\mathcal{A}$. Let p, q be the output of $\mathcal{A}$. The algorithm $\mathcal{B}$ first verifies if $N = pq$ (that means, $\mathcal{A}$ was successful). Then, they compute $\varphi(N) = (p-1)(q-1)$ (the formula has been shown in previous exercises). Now, the algorithm $\mathcal{B}$ can use the extended Euclidean algorithm (learned in previous lectures) to compute $d \in (\mathbb{Z}/\varphi(N)\mathbb{Z})^\times$ such that $e \cdot d = 1 \bmod \varphi(N)$. The algorithm $\mathcal{B}$ then computes $x = y^d \bmod N$ and outputs x as their solution.

II Analysis: We first check that the view of $\mathcal{A}$ has the correct form. Their view only consists of the number N given as input. The N provided to $\mathcal{A}$ by $\mathcal{B}$ has the correct form as it is the same in the factoring problem and in the RSA problem.
Now, we check that the output of $\mathcal{B}$ is a solution to the RSA problem. Indeed, it holds

$$x^e = (y^d)^e = y^{de} = y^{de \bmod \varphi(N)} = y \bmod N.$$

Here, we used the properties of the group $(\mathbb{Z}/N\mathbb{Z})^\times$ of order $\varphi(N)$.
Finally, we observe that computing $\varphi(N)$ and d can be done in polynomial time.
Overall $\mathcal{B}$ is PPT as long as $\mathcal{A}$ is PPT. And

$$\Pr[\mathcal{B} \text{ wins the RSA game}] > \Pr[\mathcal{A} \text{ wins the factoring game}] > \mathsf{negl}(\lambda).$$

This concludes the reduction.

Exercise 2. *CDH to Discrete Log*

1. Show (formally) that the problem CDH reduces to the DLog problem, both introduced in the first lecture. In other words, show that if there exists an algorithm $\mathcal{A}$ which solves the DLog problem with non-negligible probability, one can construct an algorithm $\mathcal{B}$ which solves the CDH problem with non-negligible probability.

Solution: Recall the definition of the **DLog problem**: Let G be a cyclic group with g a generator and λ-bit order $\mathsf{ord}(G)$. Sample $t \leftarrow \{0, \ldots, \mathsf{ord}(G) - 1\}$ and compute $h = g^t \in G$. Given as input (G, g, h), find t.

Recall the definition of the **CDH problem**: Let G be a cyclic group with g a generator and λ-bit order $\mathsf{ord}(G)$. Sample $t_1, t_2 \leftarrow \{0, \ldots, \mathsf{ord}(G) - 1\}$ (identically distributed, but independent) and compute $h_1 = g^{t_1}$ and $h_2 = g^{t_2}$ in G. Given as input (G, g, h_1, h_2), find $h = g^{t_1 t_2} \in G$.

I Description: Let $\mathcal{A}$ be a PPT algorithm solving the DLog problem with probability $> \mathsf{negl}(\lambda)$. Our goal is to construct a PPT algorithm $\mathcal{B}$ solving the CDH problem with probability $> \mathsf{negl}(\lambda)$.

Let (G, g, h_1, h_2) be the input given to $\mathcal{B}$. The algorithm $\mathcal{B}$ takes (G, g, h_1) and gives it as input to $\mathcal{A}$. Let t_1 be the output of $\mathcal{A}$. The algorithm $\mathcal{B}$ does make use of $\mathcal{A}$ a second time, by taking (G, g, h_2) and giving it as input to $\mathcal{A}$. Let t_2 be the second output of $\mathcal{A}$. The algorithm $\mathcal{B}$ first verifies if $h_1 = g^{t_1}$ and $h_2 = g^{t_2}$ (that means, $\mathcal{A}$ was successful in both cases). Then, they compute $h = g^{t_1 t_2}$ and output h as their solution.

II Analysis: We first check that the view of $\mathcal{A}$ has the correct form. Their view consists of the triplets (G, g, h_1) and (G, g, h_2) given as input. Both triplets have the correct form as the choice of G and g are the same in the DLog problem and in the CDH problem. Moreover h_1 and h_2 are computed by choosing random elements t_1 and t_2 from the correct set.
Now, we check that the output of $\mathcal{B}$ is a solution to the CDH problem. Indeed, it holds $h = g^{t_1 t_2}$ where $h_1 = g^{t_1}$ and $h_2 = g^{t_2}$. Finally, we observe that computing h can be done in polynomial time. Overall $\mathcal{B}$ is PPT as long as $\mathcal{A}$ is PPT. And

$$\Pr[\mathcal{B} \text{ wins the CDH game}] > \Pr[\mathcal{A} \text{ wins the DLog game}] > \mathsf{negl}(\lambda).$$

This concludes the reduction.