TD3: Security Assumptions

Exercise 1. *Attacking the DLG problem*

Let $\mathbb{G}$ be a cyclic group generated by g, of (known) prime order p, and let h be an element of $\mathbb{G}$. Let $F : \mathbb{G} \to \mathbb{Z}_p$ be a nonzero function, and let us define the function $H : \mathbb{G} \to \mathbb{G}$ by $H(\alpha) = \alpha \cdot h \cdot g^{F(\alpha)}$. We consider the following algorithm (called *Pollard ρ Algorithm*).

Pollard ρ Algorithm

Input: $h, g \in \mathbb{G}$

Output: $x \in \{0, \ldots, p-1\}$ such that $h = g^x$ or FAIL.

1. $i \leftarrow 1$
2. $x \leftarrow 0$, $\alpha \leftarrow h$
3. $y \leftarrow F(\alpha)$; $\beta \leftarrow H(\alpha)$
4. **while** $\alpha \neq \beta$ do
5. $\quad x \leftarrow x + F(\alpha) \bmod p; \alpha \leftarrow H(\alpha)$
6. $\quad y \leftarrow y + F(\beta) \bmod p; \beta \leftarrow H(\beta)$
7. $\quad y \leftarrow y + F(\beta) \bmod p; \beta \leftarrow H(\beta)$
8. $\quad i \leftarrow i + 1$
9. **end while**
10. **if** $i < p$ **then**
11. $\quad$ **return** $(x - y)/i \bmod p$
12. **else**
13. $\quad$ **return** FAIL
14. **end if**

To study this algorithm, we define the sequence (γ_i) by $\gamma_1 = h$ and $\gamma_{i+1} = H(\gamma_i)$ for $i \geqslant 1$.

1. Show that in the **while** loop from Steps 4 to 9 of the algorithm, we have $\alpha = \gamma_i = g^x h^i$ and $\beta = \gamma_{2i} = g^y h^{2i}$.

2. Show that if this loop terminates with $i < p$, then the algorithm returns the discrete logarithm of h in basis g.

3. Let j be the smallest integer such that there exists $k < j$ such that $\gamma_j = \gamma_k$. Show that $j \leqslant p + 1$ and that the loop ends with $i < j$.

4. Show that if F is a random function, then the average execution time of the algorithm is in $O(p^{1/2})$ multiplications in $\mathbb{G}$.

Exercise 2. *Around the DDH assumption*

We recall the definition of the DDH assumption.

Definition 1 (Decisional Diffie-Hellman distribution). *Let $\mathbb{G}$ be a cyclic group of (prime) order p, and let g be a public generator of $\mathbb{G}$. The* decisional Diffie-Hellman *distribution (DDH) is,* $D_{\text{DDH}} = (g^a, g^b, g^{ab}) \in \mathbb{G}^3$ *with a, b sampled independently and uniformly in $\mathbb{Z}/p\mathbb{Z} =: \mathbb{Z}_p$.*

Definition 2 (Decisional Diffie-Hellman assumption). *The decisional Diffie-Hellman assumption states that there exists no probabilistic polynomial-time distinguisher between D_{DDH} and (g^a, g^b, g^c) with a, b, c sampled independently and uniformly at random in $\mathbb{Z}_p$.*

1. Does the DDH assumption hold in $\mathbb{G} = (\mathbb{Z}_p, +)$ for $p = \mathcal{O}(2^\lambda)$ prime?

2. Same question for $\mathbb{G} = (\mathbb{Z}_p^\star, \times)$ of order $p - 1$, with p an odd prime.

Exercise 3. *LWE assumption*

We recall the Learning with Errors assumption.

Definition 3 (Learning with Errors). *Let $q \in \mathbb{N}$, $B \in \mathbb{N}$, $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{m \times n})$. The Learning with Errors (*LWE*) distribution is defined as follows: $D_{\text{LWE}} = (\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod q)$ for $\mathbf{s} \hookleftarrow U(\mathbb{Z}_q^n)$, $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{m \times n})$ and $\mathbf{e} \hookleftarrow U\left((-B, B]^m\right)$.*

In this setting, the vector $\mathbf{s}$ is called the secret, and $\mathbf{e}$ the noise.
Remark. If q and B are powers of 2, we are manipulating bits, contrary to the DDH-based PRG from the lecture.

The *LWE assumption* states that, given suitable parameters q, B, m, n, it is computationally hard to distinguish D_{LWE} from the distribution $U(\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m)$.
Let us propose the following pseudo-random generator: $G(\mathbf{A}, \mathbf{s}, \mathbf{e}) = (\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod q)$.

1. By definition, a PRG must have a bigger output size than input size. Give a bound on B that depends on the other parameters if we want G to satisfy this.

2. Given suitable B, q, n, m such that the LWE assumption and previous bound hold, show that G is a secure pseudo-random generator.