TD3: Security Assumptions (corrected version)

Exercise 1.

Let $(\mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme over $K \times P \times \{0,1\}^n$.

1. In this question, we assume that $(\mathsf{Enc}, \mathsf{Dec})$ is smCPA-secure. Prove that there exists a smCPA-secure encryption scheme $(\mathsf{Enc}', \mathsf{Dec}')$ such that $G : k \mapsto \mathsf{Enc}'(k, 0)$ is not a secure PRG. *Hint: try to concatenate constant bits to every ciphertext.*

☞ Define $\mathsf{Enc}' : (k, m) \mapsto 1^\ell || \mathsf{Enc}(k, m)$. The decryption algorithm Dec' ignores the first ℓ bits and calls Dec on the remaining ones. We have two things to prove:

- The pair $(\mathsf{Enc}', \mathsf{Dec}')$ is a smCPA-secure encryption scheme.
- $G : k \mapsto 1^\ell || \mathsf{Enc}(k, 0)$ is not a secure PRG.

We start with the first claim. If we assume by contradiction that there exists an efficient adversary $\mathcal{A}$ that breaks the smCPA-security of $(\mathsf{Enc}', \mathsf{Dec}')$, we build $\mathcal{A}'$ against the smCPA-security of $(\mathsf{Enc}, \mathsf{Dec})$ the following way. It starts by calling $\mathcal{A}$. When $\mathcal{A}$ outputs two messages m_0, m_1, $\mathcal{A}'$ outputs the same messages to the challenger. When the challenger outputs a ciphertext c, $\mathcal{A}'$ sends to $\mathcal{A}$ the ciphertext $1^\ell || c$. When $\mathcal{A}$ outputs a bit b', $\mathcal{A}'$ outputs the same. This is summed up in the following sketch:

$\mathcal{C}$	$\mathcal{A}'$	$\mathcal{A}$
$k \hookleftarrow U(K)$		
	Call $\mathcal{A}$	
		Choose and send $(m_0, m_1) \in P'$
	Send the same messages (m_0, m_1)	
Send $c := \mathsf{Enc}(k, m_b)$		
	Compute and send to $\mathcal{A}$: $c' := 1^\ell \| c$	
		Output b'
	Output b'	

In these games, the view of $\mathcal{A}$ is the same as in the previous question. This means that it behaves the same way as in the Exp_b games for the encryption scheme $(\mathsf{Enc}', \mathsf{Dec})$. By definition of the advantage, $\mathsf{Adv}(\mathcal{A}') = \mathsf{Adv}(\mathcal{A})$. Thus, this breaks the security of $(\mathsf{Enc}, \mathsf{Dec})$.

We move on to prove the second claim by exhibiting an efficient distinguisher $\mathcal{B}$. It does the following: upon receiving a sample from either $G(U(K))$ or the uniform distribution, it outputs 1 if the first ℓ bits are 1 and 0 otherwise. Its advantage is $1 - \frac{1}{2^\ell}$. It is non-negligible as soon as $\ell \geq 1$.

Exercise 2. *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. **return** $(x - y)/i \bmod p$
12. **else**
13. **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 $\beta = \gamma_{2i} = g^y h^{2i}$.

☞ We check these identities by induction on $i \geq 1$. For $i = 1$, they are satisfied since from lines 1 to 3 of the algorithm, we have $x = 0, \alpha = h, y = F(h)$, and $\beta = H(h) = g^y h^2$.

Now, let $i \geqslant 1$ and denote by $x_i, \alpha_i, y_i, \beta_i$ the values taken by x, α, y, β at the beginning of the i-th iteration of the **while** loop. We assume that the identities $\alpha_i = \gamma_i = g^{x_i} h^i$ and $\beta_i = \gamma_{2i} = g^{y_i} h^{2i}$ hold.

At the end of the i-th iteration (or the beginning of the $i+1$-th), we have $x_{i+1} = x_i + F(\alpha_i) \bmod p$, and $\alpha_{i+1} = H(\alpha_i) = \alpha_i \cdot h \cdot g^{F(\alpha_i)} = (g^{x_i} \cdot h^i) \cdot h \cdot g^{F(\alpha_i)} = g^{x_i + F(\alpha_i)} \cdot h^{i+1} = g^{x_{i+1}} \cdot h^{i+1}$. We also have $\beta_{i+1} = H(H(\beta_i)) = H(\beta_i \cdot h \cdot g^{F(\beta_i)}) = \beta_i \cdot h^2 \cdot g^{F(\beta_i)} \cdot g^{F(H(\beta_i))} = g^{y_i} \cdot g^{F(\beta_i)} \cdot g^{F(H(\beta_i))} \cdot h^{2i+2}$, and $y_{i+1} = y_i + F(\beta_i) + F(H(\beta_i))$, hence the identity $\beta_{i+1} = g^{y_{i+1}} h^{2i+2}$.

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

☞ When the loop finishes, we have $\alpha = \beta$ and according to Question 1, this gives $g^x h^i = g^y h^{2i}$, thus $h^i = g^{x-y}$. If furthermore the loop finishes with $i < p$ (note that $i > 0$), then since p is prime, i is invertible modulo p and $h = g^u$ where $u = (x - y)/i \bmod p$.

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 $i < j$.

☞ The sequence (γ_i) has its values in the finite group $\mathbb{G}$ of cardinality p. By the pigeonhole principle, there exist two indices $k < j \leqslant p + 1$ such that $\gamma_k = \gamma_j$; then, since (γ_i) is defined by $\gamma_{i+1} = H(\gamma_i)$, this sequence repeats with period a divisor of $j - k$.

Remark: we have $\gamma_{k+t} = \gamma_{j+t}$ for any integer $t \geqslant 0$. This leads to representing the values of the sequence in a shape which looks like the letter ρ, hence the name of the algorithm.

From Question 1, we see that the algorithm simultaneously computes the values of γ_i and γ_{2i} and returns the first index i for which $\gamma_i = \gamma_{2i}$. Since the sequence repeats with period $j - k$, considering the smallest multiple i of $j - k$ that is greater or equal to k, namely $i = (j-k)\lceil \frac{k}{j-k} \rceil$, we have that $\gamma_i = \gamma_{2i}$, since $i \geqslant k$ and $2i - i = i$ is a multiple of the period $j - k$. Besides, the sequence $k, k+1, \ldots, k + (j - k - 1)$ contains a multiple of $j - k$, so that $i \leqslant j - 1$ (we can also deduce it from the formula above for i).

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 $\mathbb{G}$.

☞ If $H : \mathbb{G} \to \mathbb{G}$ is a random function, according to the birthday paradox, the expected number of elements of the sequence (γ_i) needed to obtain two identical values is approximately $\sqrt{\pi p/2}$. Since every iteration of the **while** loop uses a constant number of multiplications in $\mathbb{G}$, the result follows.