TD 8: Public Key Cryptography

Exercise 1. [*HMAC*]

1. In the Merkle-Damgård transform, the message is split into consecutive blocks, and we add as a last block the binary representation of the length of this message. Suppose that we do not add this block: does this transform still lead to a collision-resistant hash function?

2. Before HMAC was invented, it was quite common to define a MAC by $\mathrm{Mac}_k(m) = H^s(k \parallel m)$ where H is a collision-resistant hash function. Show that this is not a secure MAC when H is constructed via the Merkle-Damgård transform.

Exercise 2. [*Pedersen's hash function*]

Pedersen's hash function is as follows:

- Given a security parameter n, algorithm `Gen` samples (G, g, q) where $G = \langle g \rangle$ is a cyclic group of cardinality q, a prime number. It then sets $g_1 = g$ and samples g_i uniformly in G for all $i \in \{2, \ldots, k\}$, where $k \geq 2$ is some parameter. Finally, it returns $(G, q, g_1, \ldots, g_k)$.
- The hash of message $M = (M_1, \ldots, M_k) \in (\mathbb{Z}/q\mathbb{Z})^k$ is $H(M) = \prod_{i=1}^k g_i^{M_i} \in G$.

1. Assume for this question that G is a subgroup of prime order q of $(\mathbb{Z}/p\mathbb{Z})^\times$, where $p = 2q + 1$ is prime. What is the compression factor in terms of k and p?

Definition 1. (Discrete Logarithm Problem (DLP)). *Given G, g, and $h \in G$ where $G = \langle g \rangle$ is a cyclic group of cardinality q, prime number. The DLP asks for $x \in \mathbb{Z}\backslash q\mathbb{Z}$ such that $g^x \equiv h \mod q$. The problem is hard if no efficient adversary can find such x with non-negligible advantage.*

2. Assume for this question that $k = 2$. Show that Pedersen's hash function is collision-resistant, under the assumption that the DLP is hard for G.

3. Same question as the previous one, with $k \geq 2$ arbitrary.

Exercise 3. [*Semantic security and CPA-security*]

Let us define the following experiments for $b \in \{0, 1\}$ and $Q = poly(\lambda)$. For $\mathrm{Exp}_b^{\text{many-CPA}}$

$\mathcal{C}$	$\mathcal{A}$
$(pk, sk) \hookleftarrow Keygen(1^\lambda)$	
sends pk to $\mathcal{A}$	
	sends $\left(m_0^i, m_1^i \right)_{i=1}^Q$ to $\mathcal{C}$
$\left(c_i^* = Enc_{pk}(m_b^{(i)})\right)_{i=1}^Q$	
sends $(c_i^*)_{i=1}^Q$ to $\mathcal{A}$	
	outputs a bit $b' \in \{0, 1\}$

The advantage of $\mathcal{A}$ in the many-time CPA game is defined as:

$$Adv^{\textit{many-CPA}}(\mathcal{A}) = \left| Pr_{(pk,sk)}[\mathcal{A} \to 1 | Exp_1^{\textit{many-CPA}}] - Pr_{(pk,sk)}[\mathcal{A} \to 1 | Exp_0^{\textit{many-CPA}}] \right|$$

1. Recall the "Semantic security" game given in the lecture. What is the difference?
2. Show that the two definitions are equivalent.
3. Do we have a similar equivalence in the private-key setting?

Exercise 4. [*Pollard-rho*]

Let $\mathbb{G}$ be a cyclic group generated by g, of (known) prime order q, and let h be an element of $\mathbb{G}$. Let $F : \mathbb{G} \to \mathbb{Z}_q$ 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: $x \in \{0, \dots, q-1\}$ such that $h = g^x$ of 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. $\quad x \leftarrow x + F(\alpha) \bmod q; \alpha \leftarrow H(\alpha)$
6. $\quad y \leftarrow y + F(\beta) \bmod q; \beta \leftarrow H(\beta)$
7. $\quad y \leftarrow y + F(\beta) \bmod q; \beta \leftarrow H(\beta)$
8. $\quad i \leftarrow i + 1$
9. **end while**
10. **if** $i < q$ **then**
11. $\quad$ **return** $(x - y)/i \bmod q$
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 lines 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 finishes with $i < q$, then the algorithm returns the discrete logarithm of h in basis g.
3. Let j be the smallest integer such that $\gamma_j = \gamma_k$ for $k < j$. Show that $j \leqslant q + 1$ 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 $O(q^{1/2})$ multiplications in $\mathbb{G}$.