TD 7: Collision-Resistant Hash Functions (corrected version)

Exercise 1.

Suppose $h_1 : \{0,1\}^{2n} \to \{0,1\}^n$ is a collision-resistant hash function.

1. Define $h_2 : \{0,1\}^{4n} \to \{0,1\}^n$ as follows: Write $x = x_1 \| x_2$ with $x_1, x_2 \in \{0,1\}^{2n}$; return the value $h_2(x) = h_1(h_1(x_1) \| h_1(x_2))$. Prove that h_2 is collision-resistant.

☞ Let $x \neq x'$ be a collision for h_2. Let us write $x = x_1 || x_2$ and $x' = x'_1 || x'_2$.

- If $h_1(x_1)||h_1(x_2) \neq h_1(x'_1)||h_1(x'_2)$, then this is a collision for h_1, as they both have the same image by h_1.
- Otherwise, notice there is a $b \in \{1,2\}$ such that $x_b \neq x'_b$ (since $x \neq x'$). Moreover $h_1(x_b) = h_1(x'_b)$. Then (x_b, x'_b) is a collision for h_1.

In the end, if we can find a collision for h_2 then we can find a collision for h_1 in polynomial time (we have four hashes to compute and two equalities to check). Then if h_1 is collision-resistant, so is h_2.

2. For $i \geq 2$, define $h_i : \{0,1\}^{2^i n} \to \{0,1\}^n$ as follows: Write $x = x_1 \| x_2$ with $x_1, x_2 \in \{0,1\}^{2^{i-1} n}$; return $h_i(x) = h_1(h_{i-1}(x_1) \| h_{i-1}(x_2))$. Prove that h_i is collision-resistant.

☞ **First method:** define the following induction hypothesis (H_i): "If we can find a collision for h_i in polynomial time then we can find a collision for h_1 in polynomial time".

This already holds for $i = 2$. Let $i \geq 2$ and assume that (H_i) is true.

Then the reduction procedes as follows: assume that we can find a collision $x \neq x'$ for h_{i+1} in polynomial time.

Let $x = x_1 || x_2$ and $x' = x'_1 || x'_2$. Then, by definition of h_{i+1}, either $h_i(x_1)||h_i(x_2) \neq h_i(x'_1)||h_i(x'_2)$ and we have a collision for h_1 by computing only four hashes, or it is equal. If it is, take any $b \in \{1,2\}$ such that $x_b \neq x'_b$: we have found a collision for h_i and can use the induction hypothesis to conclude and find a collision for h_1 in polynomial time.

Then under the collision-resistance (and assumption that i is such that h_i can still be computed in polynomial time) of h_1, it holds that h_i is collision-resistant.

Second method: Given an adversary $\mathcal{A}_{i+1}$ that finds a collision for h_{i+1} with advantage ε non-negligible and assuming that h_1 and h_i are collision-resistant, we build two adversaries:

- First, $\mathcal{A}_1$ is an adversary against the collision-resistance of h_1 that on input $x \neq x'$ from $\mathcal{A}_{i+1}$ such that $h_{i+1}(x) = h_{i+1}(x')$ returns $(h_i(x_1)||h_i(x_2), h_i(x'_1)||h_i(x'_2))$ if these two values are different. Otherwise it outputs $FAIL$.
- Second, $\mathcal{A}_i$ is an adversary against the collision-resistance of h_i that on input $x \neq x'$ from $\mathcal{A}_{i+1}$ such that $h_{i+1}(x) = h_{i+1}(x')$ returns x_b, x'_b if there eixsts a $b \in \{1,2\}$ such that $x_b \neq x'_b$ and $h_i(x_1)||h_i(x_2) = h_i(x'_1)||h_i(x'_2)$. Otherwise it returns $FAIL$

Notice that

$$\Pr(\mathcal{A}_{i+1} \text{ wins}) = \Pr(\mathcal{A}_i \text{ wins}) + \Pr(\mathcal{A}_1 \text{ wins}).$$

Since this corresponds to the advantages of the adversaries, it holds that the right hand side is negligible, under the security of h_1 and h_i, but the left hand side is non-negligible, which is a contradiction: h_{i+1} is collision-resistant.

Exercise 2.

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?

☞ No. Take for instance x of length $B\ell(n) - 1$ for some $B \geq 2$, and $y = x \| 0$. In the transform, we start by padding x with one zero so that its length is a multiple of $\ell(n)$: we obtain y. In the rest of the process, the only thing that differs between x and y is that their "length blocks" are not the same; without this length block, x and y form a collision.

2. Before HMAC was invented, it was quite common to define a MAC by $\mathrm{Mac}_k(m) = H^s(k \| m)$ 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.

☞ The goal is to construct (m, t) with $\mathrm{Verify}_k(m, t) = 1$, having oracle access to Mac_k but without querying $\mathrm{Mac}_k(m)$ itself.

With Merkle-Damgård, the function H^s divides the message $k \| m$ in p blocks $x_1, \ldots, x_p$ of size ℓ (padding the last block x_p with a Padding Block PB so that $x_p \| \mathrm{PB}$ has size ℓ) and then adding a new block x_{p+1} of length ℓ depending on the bit length of $k \| m$. Then the Merkle-Damgård construction uses a (fixed-length) collision-resistant hash function h to compute its output as follows:

$$H^s(k \| m) = h^s(x_{p+1}, h^s(x_p \| \mathrm{PB}, h^s(x_{p-1}, h^s(\ldots, h^s(x_1, \mathrm{IV}))))).$$

Given $H^s(k \| m)$, anyone can compute $H^s(k \| m \| \mathrm{PB} \| x_{p+1} \| \omega)$ for any ω; for instance, if ω is of size ℓ, using $h^s(x'_{p+2}, h^s(\omega, H^s(k \| m)))$ where x'_{p+2} only depends on the length of $k \| m \| PB \| x_{p+1} \| \omega$ and can be publicly computed.

Exercise 3.
Let $m \geq n \geq 2$, $q \geq 2$ and $B > 0$ such that $mB \leq q/4$, with q prime. Recall that the $\mathsf{LWE}_{m,n,q,B}$ hardness assumption states that the distribution $(\mathbf{A}, \mathbf{As} + \mathbf{e})$, where $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{m\times n})$, $\mathbf{s} \hookleftarrow U(\mathbb{Z}_q^n)$ and $e \hookleftarrow U((-B, B]^m)$ is computationally indistinguishable from $U(\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m)$. Define the following hash function:

$$H_{\mathbf{A}} : \{0,1\}^m \to \mathbb{Z}_q^n$$
$$\mathbf{x} \mapsto \mathbf{x}^\top \cdot \mathbf{A} \bmod q$$

1. **(a)** Recall the definition of the compression factor, and compute it for H.

   ☞ The compression factor is the ratio of the bitsize of the input over the bitsize of the output. Here, the compression factor is $\frac{m}{n \log_2 q}$.

   (b) Show how to break the $\mathsf{LWE}_{m,n,q,B}$ assumption given a vector $\mathbf{x} \in \{-1,0,1\}^m$ such that $\mathbf{x}^\top \mathbf{A} = \mathbf{0} \bmod q$ and $\mathbf{x} \neq \mathbf{0}$.

   ☞ Let $\mathbf{u} \hookleftarrow U(\mathbb{Z}_q^m)$. Then $\mathbf{x}^\top \mathbf{u} \bmod q$ is uniform over $\mathbb{Z}$, because q is prime and the coefficients of $\mathbf{u}$ are independently sampled. However, $\mathbf{x}^\top(\mathbf{As} + \mathbf{e}) = \mathbf{x}^\top \mathbf{e} \bmod q$, and this has absolute value $\leq m \cdot B \leq q/4$ (we take representatives in $(-q/2, q/2]$).

   It is then possible to distinguish between these two distributions with advantage $1/2$.

   (c) Conclude on the collision-resistance of H.

   ☞ Assume that an adversary $\mathcal{A}$ can find collisions in polynomial time with non-negligible probability.
   We build a distinguisher $\mathcal{B}$ that does the following: on input $(\mathbf{A}, \mathbf{b})$, it sends $\mathbf{A}$ to $\mathcal{A}$. If $\mathcal{A}$ fails, it returns a random bit. When it finds a collision $(\mathbf{x}, \mathbf{x}')$, adversary $\mathcal{B}$ computes $(\mathbf{x} - \mathbf{x}')^\top \mathbf{b}$ and returns LWE if it has absolute value $\leq q/4$, otherwise it returns $UNIF$.

   Then the advantage of $\mathcal{B}$ is $\mathsf{Adv}(\mathcal{A})/2$, which is non-negligible.

Exercise 4.
Pedersen's hash function is as follows:

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

1. Bound the cost of hashing, in terms of k and the number of multiplications in G.

   ☞ Here is a simple algorithm (the algorithm could be more adaptive and, before exponentiation, group together M_i's that are close to each other... we really don't care about that here!). First, use fast exponentiation to compute the powers of g_i, and then multiplies them together. This is done in, roughly, $\mathcal{O}(k \log(p))$ multiplications in the group G (more precisely, $\lceil \log_2(M_1) \rceil + \cdots + \lceil \log_2(M_k) \rceil + k - 1$).

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

   ☞ An element of G is represented with $\|p\|$ bits, where $\|p\|$ stands for the bitsize of q as an element of $\mathbb{Z}/p\mathbb{Z}$ is represented with $\|p\| = \|q - 1\| - 1$ bits, and since q is odd, $\|p\| = \|q\| - 1$. Thus, the compression factor of this function $(\mathbb{Z}/p\mathbb{Z})^k \to G$ is $k\|p\|/\|p\| = k$.
   Now, we choose k which minimizes the ratio "computation cost / compression factor" (we want the hashing to be as fast as possible and to compress as much as possible). The computation cost, in this specific context, is of $k\|p\|$ multiplications in G. Then the ratio is $\|q\|$ which is constant: any k is good.

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

   ☞ Let $\mathcal{A}$ be a PPT algorithm which finds a collision for H with probability $\varepsilon(n)$. We will use $\mathcal{A}$ to solve the DLP. More precisely, we show that the following PPT algorithm $\mathcal{A}'$ solves the DLP with probability of success $\varepsilon(n)$.

   Algorithm $\mathcal{A}'$:
   Input: G, p, g, h.
   Output: $x \in \mathbb{Z}/p\mathbb{Z}$.

1. Run $\mathcal{A}$ on (G, p, g, h) and obtain $M = (M_1, M_2)$ and $M' = (M'_1, M'_2)$.
2. If $M \neq M'$ and $H(M) = H(M')$ (collision):
   (a) If $h = 1$ then return 0.
   (b) Otherwise, return $(M_1 - M'_1)(M'_2 - M_2)^{-1} \mod p$.
3. Otherwise, `fail`

By construction, the input (G, p, g, h) is distributed exactly as in the collision experiment for $\mathcal{A}$, so that the probability of having a collision (satisfying the assertion of the first if statement) is $\varepsilon(n)$. Then, if (M, M') is indeed a collision, we show that $\mathcal{A}'$ solves the DLP, that is, returns $\log_g(h)$. This is obvious if $h = 1$, since then $\mathcal{A}'$ returns 0.

Now, if $h \neq 1$, we have $g^{M_1} h^{M_2} = g^{M'_1} h^{M'_2}$ with necessarily $M_2 \neq M'_2$ (otherwise, $g^{M_1} = g^{M'_1}$ and since g generates the group we would have $M = M'$), and therefore $M_2 - M'_2$ is invertible modulo the prime number p. Thus, writing $x = \log_g(h)$, we obtain $g^{M_1 + xM_2} = g^{M'_1 + xM'_2}$, so that $x = (M_1 - M'_1)(M'_2 - M_2)^{-1}$ (in $\mathbb{Z}/p\mathbb{Z}$).

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

☞ Let $\mathcal{A}$ be a PPT algorithm which finds a collision for H with probability $\varepsilon(n)$. We will use $\mathcal{A}$ to solve the DLP. More precisely, we show that the following PPT algorithm $\mathcal{A}'$ solves the DLP with good probability of success (close to $\varepsilon(n)$).

Algorithm $\mathcal{A}'$:
Input: G, p, g, h.
Output: $x \in \mathbb{Z}/p\mathbb{Z}$.

1. Choose uniformly $\alpha_2, \beta_2, \ldots, \alpha_k, \beta_k$ in $\mathbb{Z}/p\mathbb{Z}$, set $\alpha_1 = 1, \beta_1 = 0$ and set $g_i = g^{\alpha_i} h^{\beta_i}$ for all $i \in \{1, \ldots, k\}$.
2. Run $\mathcal{A}$ on $(G, p, g_1, \ldots, g_k)$ and obtain $M = (M_1, \ldots, M_k)$ and $M' = (M'_1, \ldots, M'_k)$.
3. If $M \neq M'$ and $H(M) = H(M')$ (collision):
   (a) If $\sum_i \beta_i (M'_i - M_i) \neq 0$, return $\sum_i \alpha_i (M_i - M'_i) \left(\sum_i \beta_i (M'_i - M_i)\right)^{-1} \mod p$.
   (b) Otherwise, `fail`
4. Otherwise, `fail`

By construction, the input $(G, p, g_1, \ldots, g_k)$ is distributed exactly as in the collision experiment for $\mathcal{A}$ Thus the probability of having a collision is $\varepsilon(n)$. Then, if (M, M') is indeed a collision, we show that $\mathcal{A}'$ returns $\log_g(h)$ with probability close to 1.

Writing $x = \log_g(h)$, we have $g^{\sum_i \alpha_i M_i + x\beta_i M_i} = g^{\sum_i \alpha_i M'_i + x\beta_i M'_i}$. Thus, $\sum_i \alpha_i (M_i - M'_i) = x \left(\sum_i \beta_i (M'_i - M_i)\right)$. Moreover, if $M \neq M'$, there exists an index i such that $M'_i - M_i \neq 0 \bmod p$. Since β_i is uniform over $\mathbb{Z}/p\mathbb{Z}$, it holds that $\sum_i \beta_i (M'_i - M_i)$ is also uniformly distributed and thus invertible with probability $\frac{p-1}{p}$. This holds because the distribution of the g_i is independent from β_i for $i \geq 2$. Indeed,

$$\begin{aligned} \Pr(\beta_i = k \cap \alpha_i + x \cdot \beta_i = \ell) &= \sum_{m \in \mathbb{Z}/p\mathbb{Z}} \Pr(x = m) \cdot \Pr(\beta_i = k \cap \alpha_i = \ell - mk) \\ &= \sum_{m \in \mathbb{Z}/p\mathbb{Z}} \Pr(x = m) \cdot \Pr(\beta_i = k) \cdot 1/p \\ &= \Pr(\beta_i = k) \cdot \Pr(\alpha_i + x\beta_i = \ell), \end{aligned}$$

as $\alpha_i + x\beta_i$ is uniformly distributed over $\mathbb{Z}/p\mathbb{Z}$, because α_i is independent from x and β_i.

Assuming that $\sum_i \beta_i (M'_1 - M_i)$ is invertible, then we directly obtain that $\mathcal{A}'$ indeed returns $x = \sum_i \alpha_i (M_i - M'_i) \left(\sum_i \beta_i (M'_i - M_i)\right)^{-1}$.