Problem Set 2
Cryptographic Hash Functions
(Summer Term 2017)

Bauhaus-Universität Weimar, Chair of Media Security
Prof. Dr. Stefan Lucks, Eik List
URL: `http://www.uni-weimar.de/de/medien/professuren/mediensicherheit/teaching/`

Due Date: 02 May 2017, 1:30 PM, via email to `eik.list(at)uni-weimar.de`.

Goal of this Problem Set: Basics of hash functions.

Task 1 – Proof of Storage (2 Credits)
Recall the Merkle-Damgård (MD) construction from the lecture. Let $H : \{0,1\}^* \to \{0,1\}^n$ be a collision-resistant hash function based on the Merkle-Damgård paradigm. Assume, Eve offers a cloud service where she advertises storage of her clients' data. Alice submits her large data message M for storage. However, Eve wants to sell her service to more clients than she has storage for. To prove to Alice that Eve still stored the data, consider the following two methods:

a) Alice sends a random n-bit challenge N to Eve. Eve responds with $Y = H(M \,\|\, N)$.

b) Alice sends a random n-bit challenge N to Eve. Eve responds with $Y = H(N \,\|\, M)$.

For each of the methods, show if Eve can trick Alice or not.

Task 2 – Merkle-Damgård (6 Credits)
Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a compression function and $H : \{0,1\}^* \to \{0,1\}^n$ a hash function based on the MD structure that uses F as internal compression function. Give *either* a proof sketch *or* contradict the following claim: If F is collision-resistant, then H is collision-resistant. *Hint: Show that every algorithm that efficiently finds a collision for H can be used to extract a collision for F with about the same success probability and complexity.*

Task 3 – Multi-Collisions (3 Credits)
Let $H : (\{0,1\}^n)^* \to \{0,1\}^n$ be a collision-resistant hash function based on the MD structure. We call a c-collision the event that one finds c disjoint messages $M^1, M^2, \ldots, M^c$ that are all mapped to the same hash value.

a) Describe an attack for finding a 4-collision for $m = 2$ that does not add more than a constant factor to the complexity of finding a 2-collision.

b) Describe the general expected complexity for c being a power of two: $c = 2^{c'}$.

Task 4 – Birthday Paradox and Hamming Weight (5 Credits)
Consider the hash function $H : \{0,1\}^{256} \to \{0,1\}^{256}$ with the following property:

$$w(x) = w(H(x)), \qquad \text{for all } x \in \{0,1\}^{256}, \tag{1}$$

where $w(x) := \sum_{i=0}^{n-1} x_i$ is the hamming weight of an n-bit string $x = (x_{n-1}, \ldots x_1, x_0)$.

a) Describe how one can use Property 1 for finding a collision on H and provide the complexities for a success probability of 0.5 and 0.99, respectively.

b) Describe how one can use Property 1 for finding an preimage $x \in \{0,1\}^{256}$ to a hash value $y \in \{0,1\}^{256}$, chosen by the adversary, and provide the lowest possible complexity.

c) Calculate the expected number of queries for finding a preimage $x \in \{0,1\}^{256}$ to a uniformly random sampled $y \twoheadleftarrow \{0,1\}^{256}$ for a success probability of 0.5.

Task 5 – Uninstantiability (2 Credits)
Let H be the hash function from Task 4, and let w denote the hamming- weight function. Let $\mathcal{SK}, \mathcal{VK}, \mathcal{S}$ be sets and $\mathcal{M} = \{0,1\}^n$. Let $\Pi = (\textsc{Gen}, \textsc{Sign}[H], \textsc{Verify}[H])$ denote a secure signature scheme (secure = it is infeasible to find a new forgery (M, S) without the signing key sk) with

- key-generation algorithm $\textsc{Gen}(1^k)$ that generates a pair of signing and verification key $(sk, vk) \in \mathcal{SK} \times \mathcal{VK}$,
- signature algorithm $\textsc{Sign}[H] : \mathcal{SK} \times \mathcal{M} \to \mathcal{S}$, which uses H internally, and
- verification algorithm $\textsc{Verify}[H] : \mathcal{VK} \times \mathcal{M} \times \mathcal{S} \to \{\textsc{true}, \textsc{false}\}$, which also uses H internally.

Let a relation $\mathcal{R} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ and a signature scheme $\Pi' = (\textsc{Gen}', \textsc{Sign}'[H], \textsc{Verify}'[H])$ be defined as follows:

11: **function** $\textsc{Gen}'(1^k)$	31: **function** $\mathcal{R}(X, Y)$
12: Store $(sk, vk) \leftarrow \textsc{Gen}(1^k)$	32: **return** $w(X) = w(Y)$
21: **function** $\textsc{Sign}'[H](M)$	41: **function** $\textsc{Verify}'[H](M, S)$
22: **if** $\mathcal{R}(M, H(M))$ **then return** sk	42: **if** $\mathcal{R}(M, H(M))$ **then return** $\textsc{true}$
23: **return** $\textsc{Sign}[H]_{sk}(M)$	43: **return** $\textsc{Verify}[H]_{vk}(M, S)$

Assume, you have black-box access to the $\textsc{Sign}'$ and $\textsc{Verify}'$ oracles (you do not have the keys). So, you can obtain signatures for messages M of your choice, and you can verify any message-signature pairs (M, S). Describe briefly an efficient forgery attack on the internal signature scheme Π; this means, find a tuple of message M' and valid corresponding signature S' for which you have not asked $\textsc{Sign}'[H](M)$ before.