TD 10: Digital Signatures, pt2 (corrected version)

Exercise 1. *Plenty of Fish in the Sea*

Let $H : \{0,1\}^{2n} \mapsto \{0,1\}^n$. We say that H is second-preimage resistant if for all efficient adversary $\mathcal{A}$, the probability that $\mathcal{A}$ succeeds in the following experiment is negligible. It is given $x \leftarrow U(\{0,1\}^{2n})$ and it has to find $x' \neq x$ such that $H(x') = H(x)$.

1. Recall the definition of collision resistance. Show that collision resistance implies second-preimage resistance.

   ☞ See lecture notes for the definition of collision resistance.

   Assume $\mathcal{A}$ breaks second-preimage resistance. We build $\mathcal{B}$ breaking collision resistance as follows. Algorithm $\mathcal{B}$ samples x uniformly and gives it as input to $\mathcal{A}$. If $\mathcal{A}$ returns x' such that $x' \neq x$ and $H(x') = H(x)$. Then algorithm $\mathcal{B}$ returns (x, x'). The success probability of $\mathcal{B}$ is exactly the same as the success probability of $\mathcal{A}$.

2. Assume that there exists a second-preimage resistant $H : \{0,1\}^{2n} \mapsto \{0,1\}^n$. Show that there exists a second-preimage resistance H' that is not collision-resistant.

   ☞ Define $H' : \{0,1\}^{2n} \mapsto \{0,1\}^{n+1}$ as $H'(x) = H(x)\|0$ if $x \notin \{0^{2n}, 1^{2n}\}$, and $H'(x) = 1^{n+1}$ else. The function H' is not collision resistant, as $(0^n, 1^n)$ is a collision.

   There remains to show that second-preimage resistance of H implies second-preimage resistance of H'. Assume that $\mathcal{A}$ is an efficient algorithm breaking the second-preimage resistance of H'. We claim that $\mathcal{A}$ also breaks second-preimage resistance of H. Indeed, its input $x \leftarrow U(\{0,1\}^{2n})$ is 1^{2n} or 0^{2n} with probability 2^{2n-1}. Assume it is not the case and that $\mathcal{A}$ succeeds. Then it finds $x' \neq x$ such that $H'(x') = H'(x)$. The last bit of $H'(x') = H'(x)$ must be 0, by construction of H'. This implies that $x' \notin \{0^{2n}, 1^{2n}\}$, and hence that $H(x) = H(x')$.

Exercise 2. *PRF and ROM*

Let $H : \{0,1\}^{2n} \to \{0,1\}^n$ be a random oracle. For $x \in \{0,1\}^n$ and $k \in \{0,1\}^n$, we define F_k as follows:

$$F_k(x) = H(k\|x).$$

The security of a PRF F_k is defined by the following game:

- A random function H, a random $k \in \{0,1\}^n$ and a uniform bit b are chosen.
- If $b = 0$, the adversary $\mathcal{A}$ is given access to an oracle for evaluating $F_k(\cdot)$. If $b = 1$ then $\mathcal{A}$ is given access an oracle for evaluating a random function mapping n-bit inputs to n-bit outputs (which is independent of H).
- $\mathcal{A}$ outputs a bit b', and succeeds if $b = b'$.

Note that during the second step, $\mathcal{A}$ can access H in addition to the function oracle provided by the experiment.
The function F_k is a PRF if for any polynomial-time adversary $\mathcal{A}$, the success probability of $\mathcal{A}$ in the preceding experiment is at most negligibly greater than $1/2$.

1. Show that F_k is a PRF.

   ☞ A random oracle is sampled uniformly over the random functions, AND this randomness is taken into account while evaluating the winning probability (which is taken into account in the exercice by the definitino stating that a random function H is sampled at the beginning of the game).

   The proof follows from the fact that $\Delta(H(k\|\cdot); f(\cdot)$ for f random mapping in $\{0,1\}^n \to \{0,1\}^n) = 0$, due to the randomness of H and k, which implies that the advantage of a distinguisher between this two distributions has zero-advantage.

Exercise 3. $(blblbl)\triangleright$ ⋈

In this exercise, we assume we have two cyclic groups G and G_T of the same known prime cardinality p, and a generator g of G. We also assume we have a pairing function $e : G \times G \to G_T$, with the following properties: It is non-degenerate, i.e., $e(g,g) \neq 1$; It is bilinear, i.e., $e(g^a, g^b) = e(g,g)^{ab}$ for all $a, b \in \mathbb{Z}/q\mathbb{Z}$; It is computable in polynomial-time. Note that the bilinearity property implies that $e(g^a, g) = e(g, g^a) = e(g,g)^a$ holds for all $a \in \mathbb{Z}/p\mathbb{Z}$.

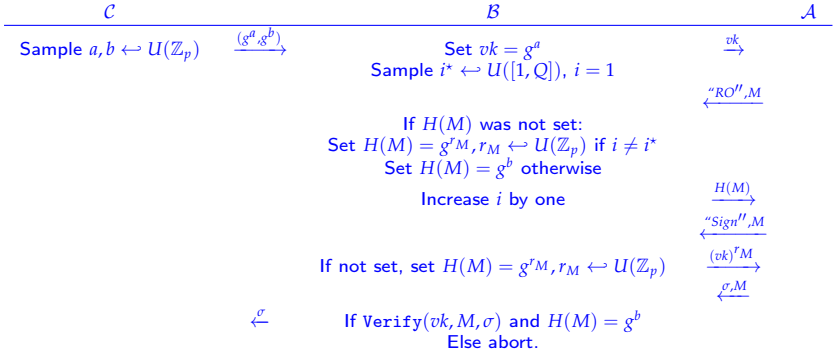
1. Show that the Decision Diffie-Hellman problem (DDH) on G can be solved in polynomial-time.

   ☞ Given g^a, g^b and g^c, test whether $e(g^a, g^b) = e(g^c, g)$. If $c = ab$, then equality holds. If c is uniform, then $e(g^c, g)$ is uniform, and as the pairing is non-degenerate, equality holds with probability $1/p$.

We consider the BLS signature scheme (due to Boneh, Lynn and Shacham), which is as follows:

- `KeyGen` takes as inputs a security parameter and returns G, g, p, G_T and a description of $e : G \times G \to G_T$ satisfying the properties above. All these are made publicly available. Sample x uniformly in $\mathbb{Z}/p\mathbb{Z}$. The verification key is $vk = g^x$, whereas the signing key is $sk = x$.
- `Sign` takes as inputs sk and a message $M \in \{0,1\}^*$. It computes $h = H(M) \in G$ where H is a hash function, and returns $\sigma = h^x$.
- `Verify` takes as inputs the verification key $vk = g^x$, a message M and a signature σ, and returns 1 if and only if $e(\sigma, g) = e(H(M), vk)$.

2. Show that this signature scheme is EU-CMA secure under the Computational Diffie-Hellman assumption (CDH) relative to G, when $H(\cdot)$ is modeled as a random oracle. ☞ Let $\mathcal{A}$ be an adversary against the EU-CMA seucrity of the signature scheme. Let Q be an upper bound on the number of (unique) random oracle queries made by $\mathcal{A}$. We build the following reduction $\mathcal{B}$:



   If both conditions at the end are verified, it holds that $e(g^b, g^a) = e(\sigma, g)$, meaning that $\sigma = g^{ab}$, and we win. The answers of $\mathcal{B}$ to RO queries are well distributed, as g is a generator of G of order p.

   The answers of $\mathcal{B}$ to signing queries are also well simulated, except if $\mathcal{A}$ queries M such that we set $H(M) = g^b$. In that case, since sign is deterministic, even if we could answer the query correctly, we would fail at the end (ie never get a forgery for M). So in that case, we can abort. Note that outputting a valid forgery for a message M without querying the Random Oracle first is highly improbable, as $H(M)$ is not yet set: the adversary only has probability $1/p$ to guess the correct value of $H(M)$, which is negligible.

   Then, assuming that $\mathcal{A}$ has non-negligible probability of winning, it has non-negligible probability of winning by forging a signature for a message it queried the RO for. Since we try to guess which message will be attacked, it holds that $\mathrm{Adv}(\mathcal{B}) \geq \Pr(\mathcal{A} \text{ wins with a forgery queried to the RO})/Q$, which is still non-negligible.

In cryptographic applications in which signing is performed very frequently (such as for cryptocurrencies), it is interesting to aggregate many signatures for multiple messages into significantly smaller space than required to store all these signatures.

3. Show that that the BLS signature scheme supports aggregation. ☞

   If we have two signature σ_1, σ_2 for messages M_1, M_2 respectively, we can compute their product $\sigma' = \sigma_1\sigma_2$. To verify that this aggregation is valid, one can check that $(H(M_1)H(M_2))^x = \sigma'$.

   We store m signatures in only 1 element of G. Of course, this comes at the price of security.

4. Propose formal definitions for the functionality and security of an aggregate signature scheme. ☞ An aggregate signature scheme is a tuple `Gen, Sign, Aggregate, Verify` such that

$\texttt{Gen}(1^\lambda)$: Outputs (vk, sk), a verification and secret keys.

$\texttt{Sign}(sk, M)$: Outputs σ, a signature for message M.

$\texttt{Aggregate}(\{\sigma_i, M_i\}_i, vk)$: Outputs σ', an aggregated signature for $\{M_i\}_i$. In particular, σ' must have smaller size than $\{\sigma_i\}_i$.

$\texttt{Verify}(\sigma', \{M_i\}_i, vk)$: Outputs 1 if σ' is an aggregated (or, if there is only one message, simply a) signature for $\{M_i\}_i$. Outputs 0 otherwise.

An aggregated signature scheme is secure if no adversary with access to a signing oracle can forge a valid aggregated signature, such that at least one message was not queried to the signing oracle.