Tutorial 9: Digital signatures

Exercise 1. *Secure pairing-based signature in the ROM*

In this exercise, we assume that we have two cyclic groups G and G_T of the same cardinality q, and a generator g of G. We also assume that 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 $a \in \mathbb{Z}/q\mathbb{Z}$.

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

2. Generalize the Diffie-Hellman key exchange protocol to derive a secure 1-round key exchange protocol between three parties. Formalize the underlying hardness assumption.

3. We consider the following signature scheme (due to Boneh, Lynn and Shacham):

   - `KeyGen` takes as inputs a security parameter and returns G, g, q, 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 $\mathbb{Z}/q\mathbb{Z}$. 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 $e(\sigma, g) = e(H(M), vk)$.

   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 (full-domain hash) random oracle. Recall that the CDH problem asks to compute g^{ab} given g^a and g^b.

Exercise 2. *Chameleon hash functions*

A chameleon hash function is a regular hash function with an additional algorithm `Trap_Coll` that computes collisions when given as input a trapdoor information. More formally, a chameleon hash function is a triple of probabilistic polynomial-time algorithms (`Gen, Hash, Trap_Coll`) with the following specifications:

- `Gen` takes as input a security parameter and returns a public key pk and a trapdoor $trap$.
- `Hash` is deterministic; it takes as inputs a public key pk, a message M and an r that can be viewed as a random string, and returns $\texttt{Hash}(pk; M, r)$.
- `Trap_Coll` takes as inputs pk, $trap$, a pair (M_1, r_1) and a message M_2, and returns r_2 such that $\texttt{Hash}(pk; M_1, r_1) = \texttt{Hash}(pk; M_2, r_2)$. Intuitively, it finds a collision by modifying the random string used to hash. Moreover, we want that if r_1 is uniform and independent of M_1 and M_2, then so is r_2.
- **Collision resistance**: Given pk (but not $trap$), it must be hard to find $(M_1, r_1) \neq (M_2, r_2)$ such that $\texttt{Hash}(pk; M_1, r_1) = \texttt{Hash}(pk; M_2, r_2)$.
- **Uniformity**: For any two messages M_1, M_2, the distributions $\texttt{Hash}(pk; M_1, r)$ and $\texttt{Hash}(pk; M_2, r)$ for r uniform must be identical.

We consider the following chameleon hash function H_{cham}:

- 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 samples x uniformly in $(\mathbb{Z}/q\mathbb{Z})^\times$ and computes $h = g^x$. It returns $pk = (G, q, g, h)$ and $trap = x$.
- To hash $M \in \mathbb{Z}/q\mathbb{Z}$ with the random string $r \in \mathbb{Z}/q\mathbb{Z}$, return $H_{cham}(pk; M, r) = g^M \cdot h^r$.

1. Show that H_{cham} is collision-resistant, under the assumption that the Discrete Logarithm Problem (DLP) is hard for G.

2. Describe a correct algorithm `Trap_Coll`.

3. Show that h is a generator of G. Derive that H_{cham} satisfies the uniformity property.

Chameleon hashing is used to transform a signature scheme that is existentially unforgeable under static chosen message (stat-EU-CMA) into a signature scheme that is existentially unforgeable under adaptive chosen message (EU-CMA). Stat-EU-CMA security of a signature scheme (`KeyGen`, `Sign`, `Verify`) is defined by the following game:

- The adversary gives to the challenger the messages $(M_1, \ldots, M_q)$ he is querying;
- The challenger replies with a verification key vk and valid signatures $(S_1, \ldots, S_q)$, i.e., satisfying $\texttt{Verify}(vk; M_i, S_i) = 1$ for all i;
- The adversary sends a pair (M^*, S^*) to the challenger;
- The adversary wins the game if $M^* \notin \{M_1, \ldots, M_q\}$ and $\texttt{Verify}(vk; M^*, S^*) = 1$.

The scheme is stat-EU-CMA-secure if no probabilistic polynomial-time adversary wins this game with non-negligible probability. We recall that in the EU-CMA security game, the message queries are sent from the adversary to the challenger **after** the challenger has made the verification key vk available to the adversary.
We now assume that we have a stat-EU-CMA-secure signature scheme (`KeyGen`, `Sign`, `Verify`) and a secure chameleon hash (`Gen`, `Hash`, `Trap_Coll`). Our goal is to build a signature scheme (`KeyGen'`, `Sign'`, `Verify'`) that is EU-CMA-secure. We define:

- `KeyGen'`: Run `KeyGen` to get a verification key vk and a secret key sk; Run `Gen` to get a public key pk and a trapdoor $trap$. Return $vk' = (vk, pk)$ and $sk' = sk$.
- `Sign'`: To sign M using $sk' = sk$, sample a uniform r, compute $h = \texttt{Hash}(pk; M, r)$, and return $S = (r, \texttt{Sign}(sk; h))$.

4. Give a (non-trivial) polynomial-time algorithm `Verify'` that accepts properly generated signatures.

5. Show that if (`KeyGen`, `Sign`, `Verify`) is stat-EU-CMA-secure and (`Gen`, `Hash`, `Trap_Coll`) is a secure chameleon hash function, then (`KeyGen'`, `Sign'`, `Verify'`) is EU-CMA-secure.