Fundamentals of Cryptography Homework 7

周书予
2000013060@stu.pku.edu.cn

December 3, 2022

Problem 1

$$\mathsf{Aggregate}(m_1, \sigma_1, \cdots, m_\ell, \sigma_\ell) = \prod_{i=1}^{\ell} \sigma_i$$

$$\mathsf{Verify}(pk, m_1, \cdots, m_\ell, \sigma) = \left[\sigma^e \equiv \prod_{i=1}^{\ell} H(m_i) \bmod N\right]$$

Assume there is a PPT adversary (the forger) $\mathcal{A}$ who braks this aggregate signature scheme, we'll show that we can construct another PPT adversary $\mathcal{A}'$ who breaks RSA assumption.

$\mathcal{A}'$ works as follows:

- $\mathcal{A}'$ is given N, y, e as described in RSA assumption. The goal of $\mathcal{A}'$ is to find x such that $x^e \equiv y \bmod N$.
- $\mathcal{A}'$ prepares the answer for the oracle queries that $\mathcal{A}$ will ask, by choosing at random $r_1, \cdots, r_{p(n)} \in \mathbb{Z}_n^*$ and an index $j \in \{1, \cdots, p(n)\}$, and let the answer to the i-th query to H be
$$H(m_i) = \begin{cases} y \cdot r_i^e, & i = j \\ r_i^e, & i \neq j \end{cases}$$
where $p(n)$ is some polynomial.
Intuitively, $\mathcal{A}'$ is betting on the chance that $\mathcal{A}$ will include its j-th oracle query in its output (the forgery).
- $\mathcal{A}'$ prepares the answer for the signature queries that $\mathcal{A}$ will ask. Typically when asked $\mathsf{Sign}(sk, m_i)$ for some $i \neq j$, $\mathcal{A}'$ simply returns r_i. When asked $\mathsf{Sign}(sk, m_j)$, $\mathcal{A}'$ fails.
- $\mathcal{A}'$ runs the forger algorithm $\mathcal{A}$, and answer its both oracle and signature queries.
- $\mathcal{A}'$ retrieves the output $(m_{i_1}, \cdots, m_{i_\ell}, \sigma)$ of $\mathcal{A}$. If $j \notin \{i_1, \cdots, i_\ell\}$ it fails, otherwise $\mathcal{A}'$ knows that w.p. at least $1/\mathrm{poly}(n)$,
$$\sigma^e = \prod_{k=1}^{\ell} H(m_{i_k}) = y \cdot \prod_{k=1}^{\ell} r_{i_k}^e$$
then it simply outputs $x = \sigma \cdot \prod_{k=1}^{\ell} r_{i_k}^{-1}$ such that $x^e = y$.

One can see that

$$\Pr\left[\mathcal{A}' \text{ breaks RSA assumption}\right] \geqslant \frac{1}{p(n)}\Pr\left[\mathcal{A} \text{ breaks this aggregate signature scheme}\right] = \frac{1}{\text{poly}(n)}$$

So $\mathcal{A}'$ indeed breaks RSA assumption.

Problem 2

Part A

The signature algorithm Sign first finds $H(m)^{-1} \bmod \varphi(N)$, which is efficient via 辗转相除. Then it uses the fast power algorithm to calculate $g^{H(m)^{-1}} \bmod N$.

Notice that addition and multiplication operation of $\mathbb{Z}_N$ elements can be down in $\text{poly}(n)$ time, and only $\text{poly}(n)$ operations are needed in the above process, thus all thins can be done in $\text{poly}(n)$-time.

Part B

Assume there is a PPT adversary $\mathcal{A}$ who braks Gennaro-Halevi-Rabin signature scheme, we'll show that we can construct another PPT adversary $\mathcal{A}'$ who breaks RSA assumption.

$\mathcal{A}'$ works as follows:

- $\mathcal{A}'$ is given N, y, e as described in RSA assumption. The goal of $\mathcal{A}'$ is to find x such that $x^e \equiv y \bmod N$.
- $\mathcal{A}'$ prepares the answer for the oracle queries that $\mathcal{A}$ will ask, by choosing at random $e_1, \cdots, e_{p(n)} \in \mathbb{Z}_n^*$ (except e_j, it sets $e_j = e$ without sampling) and an index $j \in \{1, \cdots, p(n)\}$, and let the answer to the i-th query to H be $H(m_i) = e_i$. Let $E = \prod_{i \neq j} e_i$, $\mathcal{A}'$ sets $g = y^E \bmod N$ as public key and sends it to $\mathcal{A}$.

  For simplicity, we suppose that $\mathcal{A}'$ chooses such e_i-s that $\gcd(e_i, e) = 1$ for all $i \neq j$, which means that $\gcd(E, e) = 1$. In the real world, this succeeds with probability at least $1/\text{poly}(n)$.

  Intuitively, $\mathcal{A}'$ is betting on the chance that $\mathcal{A}$ will use its j-th oracle query as its output (the forgery).
- $\mathcal{A}'$ prepares the answer for the signature queries that $\mathcal{A}$ will ask. Typically when asked $\mathsf{Sign}(sk, m_i)$ for some $i \neq j$, $\mathcal{A}'$ simply returns y^{E/e_i} (notice that E/e_i is an integer when $i \neq j$). When asked $\mathsf{Sign}(sk, m_j)$, $\mathcal{A}'$ fails.
- $\mathcal{A}'$ runs the forger algorithm $\mathcal{A}$, and answer its both oracle and signature queries.
- $\mathcal{A}'$ retrieves the output (m, σ) of $\mathcal{A}$. If $m \neq m_j$ it fails, otherwise $\mathcal{A}'$ knows that w.p. at least $1/\text{poly}(n)$,

$$\sigma^e = y^E$$

now $\mathcal{A}'$ computes two integer a, b such that $ae + bE = 1$ by extended Euclidean algorithm, and outputs

$$x = \sigma^b \cdot y^a$$

such that

$$x^e = (\sigma^e)^b \cdot y^{ae} = y^{bE} \cdot y^{ae} = y$$

Part C

$$\mathsf{Verify}'(pk, m_1, \cdots, m_\ell, \sigma) = \left[\sigma^{\prod_i H(m_i)} = g^{\sum_i \prod_{j \neq i} H(m_j)}\right]$$

Part D

(We assume that $m_1, \cdots, m_\ell$ be a sequence of distinct messages such that all $H(m_i)$-s are pairwise-coprime.)

With aggregate signature σ and messages $m_1, \cdots, m_\ell$ given, one can calculate $\sigma^{\prod_{i \neq 1} H(m_i)}$ which gives the value

$$v_1 = \sigma^{\prod_{i \neq 1} H(m_i)} = g^{\frac{\prod_{i \neq 1} H(m_i)}{H(m_1)} + \sum_{j=2}^{\ell} \prod_{i \neq 1, i \neq j} H(m_i)}$$

Since for all $j \in \{2, \cdots, \ell\}$, the value $g^{\prod_{i \neq 1, i \neq j} H(m_i)}$ can be calculated efficiently, one can further obtain the value

$$v_2 = v_1 / \prod_{j=2}^{\ell} g^{\prod_{i \neq 1, i \neq j} H(m_i)} = g^{\frac{\prod_{i \neq 1} H(m_i)}{H(m_1)}}$$

Notice that by our assumption, $\gcd\left(H(m_1), \prod_{i \neq 1} H(m_i)\right) = 1$, thus we can compute two integer a, b such that $aH(m_1) + b\prod_{i \neq 1} H(m_i) = 1$, and it follows

$$v_3 = g^a v_2^b = g^{\frac{aH(m_1) + b\prod_{i \neq 1} H(m_i)}{H(m_1)}} = g^{1/H(m_1)}$$

which gives the signature of message m_1.

Problem 3

Notice that the equation

$$x^e = g \bmod N$$

has and only has one solution

$$x = g^d \bmod N$$

for some $g \in \mathbb{Z}_N^*$, and $ed = 1 \bmod \varphi(N)$. This suggests that Verify-s in the RSA signature and the Gennaro-Halevi-Rabin Signature are both canonical.

And since both Sign-s are deterministic, existential unforgeability implies strong unforgeability.