Today’s Agenda

- Digital Signatures Review
- RSA Signatures
- Random Oracle Methodology

1 Digital Signature Review

**Definition 1.** A digital signature scheme for a message space \( M \) consists of PPT algorithms \((\text{KeyGen}, \text{Sign}, \text{Verify})\) such that:

1. **Correctness:** \( \forall (vk, sk) \in \text{KeyGen}(1^k), \forall m \in M \) and for all \( \sigma \in \text{Sign}(sk, m) \),
   \[ \text{Verify}(vk, m, \sigma) = \text{Accept} \]

2. **Security:** For all PPT \( A \exists \negl. \nu() \) such that
   \[ \Pr[(vk, sk) \in \text{KeyGen}(1^k) ; (Q, m', \sigma') \leftarrow A^{\text{Sign}(sk, \cdot)}(vk) : m' \notin Q \text{ and } \text{Verify}(vk, m', \sigma') = \text{Accept}] = \nu(k) \]
   
   Our adversary \( A \) has access to the signing oracle \( \text{Sign}(sk, \cdot) \) and can get signatures \( \sigma_1, \sigma_2, \ldots, \sigma_n \) on his choice of messages \( m_1, \ldots, m_n \). This list of message-signature pairs is outputted as \( Q \). This cannot be tampered with and is fixed by \( A \)’s queries.

Let us see the original RSA signature:

1. **KeyGen(1\(^k\)):** Let \( N = p.q \), product of two \( k \)-bit primes. Output \( vk = (N, e) \) and \( sk = d \) for message space \( M = \mathbb{Z}_N^* \)
2. **Sign(sk, m):** \( m^d \mod N \)
3. **Verify(vk, m, \sigma):** Check if \( \sigma^e = m \mod N \)

2 RSA Signature

The above mentioned signature scheme restricts the size of message and also has various attacks. So what is actually used in practice is a modified version of the original signature called the RSA Full-domain Hash Signature:

1. **KeyGen(1\(^k\)):** Let \( N = p.q \), product of two \( k \)-bit primes. Output \( vk = (N, e) \) and \( sk = d \) for message space \( M = \{0, 1\}^* \)
2. **Sign(sk, m):** \( (H(m))^d \mod N \) where \( H : \{0, 1\}^* \rightarrow \mathbb{Z}_N^* \) is a “magic” hash function.
3. **Verify(vk, m, \sigma):** Check if \( \sigma^e = H(m) \mod N \)
But this scheme is provably NOT provably secure meaning that you can design a very contrived hash function [Canetti-Goldreich-Halevi’98] which contradicts the security notion of signatures as we have defined it. What we do know is the following:

1. For all algorithmic instantiations of $H$, there exists some contrived TDP $f$ such that full-domain hash signature using $f$ is broken.

2. There exists a contrived CRHF $H$ such that full-domain hash signature using $f$ is broken for $H$.

But it is provably secure in what is called the Random Oracle Model [Bellare-Rogaway’96].

3 Random Oracle Model

This model states that each party has oracle access to a truly random function $R()$. For the purposes of a specific protocol, this random oracle can be instantiated with a specific function, say $H()$. Security in the Random Oracle model means that for all PPT $A$ that do not have the description for $H$ but only oracle access to it, there exists a negligible $\gamma()$ such that when $H : \{0,1\}^* \mapsto \mathbb{Z}_N^*$ is a random function. It seems like the ROM is sort of adhoc and may not really be a formal way of proving security, but it is still a reasonable way of proving.

For example, let $H$ be a program such that without its description it is truly indistinguishable from a truly random function. Let $A$ be a program attacking Full-domain hash (FDH)-RSA that tries to attack the signature without knowing how $H$ works algorithmically and using $H$ as a black-box. Every time $A$ needs to evaluate $H$ it makes explicit subroutine calls to $H$ and still works if the subroutine is replaced by truly random function. Then $A$ has only negligible probability of breaking FDH-RSA (if RSA assumption holds). Another reason why proof in the ROM is acceptable is that most of the cryptanalysis and attacks on RSA-signatures do not care about structure of $H$ and in fact treats it as a black-box.

Let us prove the security of RSA-signatures in the ROM model: Or reduction $B$ (trying to break TDP $f_{pk}$) will work as follows:

On input $f_{pk}, y$:

1. Set $vk = f_{pk}$ as input to $A$

2. To answer queries to $H$: On input $s_i$, return random $t_i \leftarrow \text{Domain of } f_{pk}$ which is $\mathbb{Z}_N$. Also remember the pairs $(s_i, t_i)$ in case $A$ asks the same query again.

3. To answer signature queries: On input $m_i$:
   
   (a) Pick random $\sigma_i \leftarrow \text{Domain of } f_{pk}$
   
   (b) Set $H(m_i) = f_{pk}(\sigma_i)$
   
   (c) Output $\sigma_i$

The problem with this is that $A$ can easily get $B$ stuck by asking $H(m)$ and then signature on $m$. Since $B$ is choosing a random element in the domain as the signature this wouldn’t necessarily match the hash value. So we make the following modifications:

On input $f_{pk}, y$: choose a random $r \leftarrow \{1, \ldots, q\}$ is the number of queries $A$ makes to $H$. So $B$ chooses a position in $A$’s queries to put $y$ for $A$ to invert it for him.

1. Set $vk = f_{pk}$ as input to $A$
2. To answer queries to $H$: On input $s_i$, pick random $t_i \leftarrow \text{Domain of } f_{pk}$ which is $\mathbb{Z}_N^*$. Also remember the pairs $(s_i, t_i)$ in case $A$ asks the same query again. Output $f_{pk}(t_i)$. For query number $r$, return $y$.

3. To answer signature queries: On input $m_i$:
   (a) Pick random $\sigma_i \leftarrow \text{Domain of } f_{pk}$ or if $H(m_i)$ has been queried in the past then $\sigma_i = t_j$
   (b) Set $H(m_i) = f_{pk}(\sigma_i)$
   (c) Output $\sigma_i$

4. When $A$ outputs a forgery $(m, \sigma)$: $f_{pk}(\sigma) = H(m) = y$. If $m \neq s_r$, fail else output $\sigma$.

Note that whatever forgery $A$ outputs, it has to query the $H$-oracle on that message. This modified working of the reduction $B$ proves security if the signature scheme in the random-oracle model.