CS395T Advanced Cryptography 2/26/2009

Lecture 11: RSA Signatures

Instructor: Brent Waters *Scribe: Sanjam Garg*

1 Review of Schnorr Signature

Schnorr Signature Scheme is based on the discrete log problem. Let $\mathbb{G}$ be a large prime order subgroup of $\mathbb{Z}_p^*$, the multiplicative group of integers modulo p for some prime p. The order of $\mathbb{G}$ is q such that $q \mid p-1$.

- $(SK, VK) \stackrel{R}{\leftarrow} Setup(\lambda)$ - Setup algorithm takes the security parameter λ as input and generates the secret key $SK \stackrel{R}{\leftarrow} x$ where $x \in \mathbb{G}$ and the verification key $VK = g^x$.
- $\sigma \stackrel{R}{\leftarrow} Sign(m, SK)$ - Pick $k \stackrel{R}{\leftarrow} \mathbb{G}$ and compute $r = g^k$. Set $h = H(m, r)$ where H is a hash function and $m \in \{0,1\}^*$ is the message being signed. Compute $s = k + h \cdot x$ and set $\sigma = (s, h)$.
- $\{0,1\} \leftarrow Verify(m, \sigma, VK)$ - Set $(s, h) \leftarrow \sigma$ and output 1 if $H(m, \frac{g^s}{y^h}) = h$ else return 0.

Review of Proof

The unforgeability of the schnorr signatures against a adaptive choosen message attack is based on the discrete log problem. The scheme is proved secrure in the random oracle model. The proof is based on the technique of the oracle replay attack. The two key challenges for a discrete log solver trying to use a schnorr forger are-

- *Generating Signatures* - The schnorr forger requires signatures on messages of its choice for its correct working. This can be achieved by carfully making use of the random oracle.
- *Using the forgery* - In order to solve the discrete log problem two forgeries with the same r as part of the signature are required and this can be achieved by rewinding the oracle.

2 RSA Signatures

2.1 RSA problem

Given $N = p \cdot q$, where p and q are large primes (of ℓ bits, based on the security parameter) it is hard to factor N by the factorization problem. Consider $\mathbb{Z}_N^*$ the group of all numbers relatively prime to p and q. The order of the group is $\phi(N) = (p-1)(q-1)$. N is called as a RSA modulus.

According to the RSA problem given (N, e) and $h \in \mathbb{Z}_N^*$ such that N is a RSA modulus and e is a random number less than $\phi(N)$ it is hard to evaluate $h^{e^{-1}} mod N$. RSA problem is not as hard as factoring. In other words, given an algorithm that can solve factoring we can easily solve for the RSA problem but it is not clear if the converse is true.

2.2 RSA Based Signature Scheme

- $(SK, VK) \xleftarrow{R} Setup(\lambda)$ - Setup algorithm takes the security parameter λ as input, publishes the verification key $VK = (e, N)$ and sets the $SK = e^{-1} mod \phi(N)$.
- $\sigma \xleftarrow{R} Sign(m, d)$ - Compute $\sigma = H(m)^d$ where H is a hash function and $m \in \{0,1\}^*$ is the message being signed.
- $\{0,1\} \leftarrow Verify(m, \sigma, e)$ - Output 1 if $H(m) = \sigma^e$ else return 0.

Theorem 2.1 (Security of RSA Signature Scheme) *The RSA based signature scheme is secure under the RSA assumption. In other words, for an adversary $\mathcal{A}^{ef-cma}$ making atmost q_s sign queries and q_h hash queries to the random oracle, that succedes in generating an existential forgery with $Adv_{\mathcal{A}} \geq \varepsilon$ in time τ, we construct a reduction $\mathcal{R}$ in the Random Oracle Model that solves the RSA problem with an advantage $Adv_{\mathcal{R}} \geq \varepsilon'$ and time τ' such that $\varepsilon \approx \frac{\varepsilon}{q_h}$ and $\tau' \approx \tau$.*

Proof. The reduction $\mathcal{R}$ receieves (N, e) and h as input and it is required to compute $h^{e^{-1}} mod N$. $\mathcal{R}$ sends (N, e) to $\mathcal{A}$. $\mathcal{A}$ will now make q_s sign queries and q_h hash queries to $\mathcal{R}$. The reduction $\mathcal{R}$ maintains a table with entries $(m, h(m), h(m)^d)$ and simulates the sign and hash queris as follows. It populates the entries in the table as it runs. The reduction also chooses a random $i^* \in [1 \dots q_h]$. And it maintains the variable i that records the number of hash query that is going to be asked. This variable is initiallized to $i = 1$. H is modeled as a random oracle.

Simulation of SIGN(m)

If there already exists an entry in the table (m, y^e, y) then it returns y as the signature. If there exists no entry with the message m then the reduction $\mathcal{R}$ pics a random $y \xleftarrow{R} \mathbb{Z}_N^*$. It sets $H(m) = y^e$ and adds the entry (m, y^e, y) to the table. It can be seen that y is a valid signature for m. It sends this value y as the response signature. If the entry corresponds to the i^{th} hash query then $\mathcal{R}$ quits.

Simulation of Hash(m)

If $i \neq i^*$ then $\mathcal{R}$ pics a random $y \xleftarrow{R} \mathbb{Z}_N^*$. It sets $H(m) = y^e$ and adds the entry (m, y^e, y) to the table. It can be seen that y is a valid signature for m. It then sends this value y^e as the hash response. But if, $i = i^*$ then it returns the hash response as h.

If the adversary $\mathcal{A}$ queries for a signature on the i^{*th} hash query then the reduction $\mathcal{R}$ quits. Reduction $\mathcal{R}$ also quits if the final forgery is not generated on the i^{*th} hash query.

The forgery generated corresponding to the i^{*th} hash query is $h^{e^{-1}} mod N$, which is the desired result required from $\mathcal{R}$.

It can be seen that with a probability $\frac{1}{q_h}$ the reduction $\mathcal{R}$ will not query signature on the i^{*th} hash query and will be able to use the forgery generated by $\mathcal{A}$ in breaking RSA problem. Hence $Adv_{\mathcal{R}} \geq \frac{\varepsilon}{q_h}$.

□

3 IBE based Signature Scheme

Consider the bilinear group $\mathbb{G}$ which has a bilinear map $e : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{G}_T$.

- $(SK, VK) \xleftarrow{R} Setup(\lambda)$ - Setup algorithm takes the security parameter λ as input, sets the $SK = x$ and publishes the verification key $VK = g^x$. Here $g^x \in \mathbb{G}$.
- $\sigma \xleftarrow{R} Sign(m, x)$ - Compute $\sigma = H(m)^x$ where H is a hash function and $m \in \{0,1\}^*$ is the message being signed.
- $\{0,1\} \leftarrow Verify(m, \sigma, g^x)$ - Output 1 if $e(\sigma, g) = e(H(m), g^x)$ else return 0.