Notes for Lecture 21

Summary

Today we show how to construct an inefficient (but efficiently verifiable) signature scheme starting from a one-time signature scheme.

Next time we shall see how to make it efficient using a pseudorandom function.

From One-Time Signatures to Fully Secure Signatures

Assume we have a (t, ϵ)-secure one-time signature scheme (G, S, V) such that if m is the length of messages that can be signed by S, then the length of public keys generated by $G()$ is at most $m/2$.

(Lamport's signatures do not satisfy the second property, but in Lecture 20 we described how to use a collision-resistant hash function to turn Lamport's scheme into a scheme that can sign longer messages. We can arrange the parameters of the construction so that the hash-and-sign scheme can sign messages at least twice as long as the public key.)

We describe a scheme in which the key generation and signing have exponential complexity; later we will see how to reduce their complexity.

- Key Generation: run $G()$ $2^{m+1} - 1$ times, once for every string $a \in \{0,1\}^*$ of length at most m, and produce a public key / secret key pair (pk_a, sk_a).

  It is convenient to think of the strings a of length at most m as being arranged in a binary tree, with a being the parent of $a0$ and $a1$, and the empty string ϵ being the root.

  - Public Key: pk_ϵ (where ϵ is the empty string)
  - Secret Key: the set of all pairs (pk_a, sk_a) for all a of length $\leq m$.

- Sign: given a message M of length m, denote by $M_{|i}$ the string $M_1, \ldots, M_i$ made of the first i bits of M. Then the signature of M is composed of $m + 1$ parts:

- $pk_M, S(sk_M, M)$: the signature of M using secret key sk_M, along with the value of the matching public key pk_M
- $pk_{M_{|m-1}}, pk_{M_{|m-1}0}||pk_{M_{|m-1}1}, S(sk_{M_{|m-1}}, pk_{M_{|m-1}0}||pk_{M_{|m-1}1})$ the signature of the public keys corresponding to M and its sibling, signed using the secret key corresponding to the parent of M, along with the matching public key
- $\cdots$
- $pk_{M_{|i}}, pk_{M_{|i}0}||pk_{M_i1}, S(sk_{M_{|i}}, pk_{M_{|i}0}||pk_{M_i1})$
- $\cdots$
- $pk_0, pk_1, S(sk_\epsilon, pk_0||pk_1)$

- Verify. The verification algorithm receives a public key pk_ϵ, a message M, and a signature made of $m+1$ pieces: the first piece is of the form (pk_m, σ_m), the following $m-1$ pieces are of the form $(pk_j, pk'_j, pk''_j, \sigma_j)$, for $j = 1, \ldots, m-1$, and the last piece is of the form $(pk'_0, pk''_0, \sigma_0)$.

  The verification algorithm:

  1. checks $V(pk_m, M, \sigma_m)$ is valid;
  2. For $j = 1, \ldots, m$, if $M_j = 0$ it checks that $pk_j = pk'_{j+1}$, and if $M_j = 1$ it checks that $pk_j = pk''_{j+1}$;
  3. For $j = 0, \ldots, m$, it checks that $V(pk_j, pk'_j||pk''_j, \sigma_j)$ is valid. (For the case $j = 0$, we take $pk_0 := pk_\epsilon$.)

Theorem 1 *Suppose that the scheme described in this section is not (t, ϵ) existentially unforgeable against a chosen message attack.*

Then (G, S, V) is not a $(t \cdot O(r \cdot m), \epsilon \cdot (2tn + 1))$-secure one time signature scheme, where r is the running time of S.