Notes for Lecture 22

Scribed by Himanshu Sharma, posted May 6, 2009

Summary

In the last lecture we described a very complex signature scheme based on one-time signatures and pseudorandom functions. Unfortunately there is no known simple and efficient signature scheme which is existentially unforgeable under a chosen message attack under general assumptions.

Today we shall see a very simple scheme based on RSA which is secure in the *random oracle model*. In this model, all parties have oracle access to a random function $H : \{0,1\}^n \to \{0,1\}^m$. In implementations, this random function is replaced by a cryptographic hash function. Unfortunately, the proof of security we shall see today breaks down when the random oracle is replaced by hash function, but at least the security in the random oracle model gives some heuristic confidence in the design soundness of the construction.

1 The Hash-and-Sign Scheme

Our starting point is the "textbook RSA" signature scheme, in which a message M is signed as $M^d \bmod N$ and an alleged signature S for a message M is verified by checking that $S^e \bmod N = M$.

We discussed various ways in which this scheme is insecure, including the fact that

1. It is easy to generate random message/ signature pairs M, S by first picking a random S and then setting $M := S^e \bmod N$;
2. If S_1 is the signature of message M_1 and S_2 is the signature of M_2, then $S_1 \cdot S_2 \bmod N$ is the signature of $M_1 \cdot M_2 \bmod N$.

Suppose now that all parties have access to a good cryptographic hash function, which we will model as a completely random function $H : \{0,1\}^m \to \mathbb{Z}_N$, mapping every possible message M to an random integer $H(M) \in \mathbb{Z}_N$, and define a signature scheme $(Gen, Sign, Verify)$ as follows:

- Key generation: as in RSA

- Signature: the signature of a message M with secret key N, d is $H(M)^d \bmod N$
- Verification: given an alleged signature S, a message M, and a public key N, e, check that $S^e \bmod N = H(M)$.

That is, we use the textbook RSA method to sign $H(M)$.

Now it is not clear any more how to employ the previously mentioned attacks. If we first select a random S, for example, then to find a message of which S is a signature we need to compute $h := S^e \bmod N$ and then find a message M such that $H(M) = h$. This, however, requires exponential time if H is a random functions. Similarly, if we have two messages M_1, M_2 and know their signatures S_1, S_2, the number $S_1 \cdot S_2 \bmod N$ is a signature for any document M such that $H(M) = H(M_1) \cdot H(M_2) \bmod N$. Finding such an M is, however, again very hard.

2 Analysis

We provide a formal analysis of the signature scheme defined in the previous section, in the random oracle model.

Theorem 1 *Suppose that $(Gen, Sign, Verify)$, as defined in the previous section, is not (t, ϵ) existentially unforgeable under a chosen message attack in the random oracle model.*

Then RSA, with the key size used in the construction, is not a $(t \cdot O(r), \epsilon \cdot \frac{1}{t})$-secure family of trapdoor permutations, where r is the time taken by RSA computation with the selected key size.

PROOF: We will prove that, if A is an algorithm of complexity at most t that breaks existential unforgeability under chosen message attack with probability $\geq \epsilon$, then there is an algorithm A' that breaks RSA (finds X given $X^e \bmod N$) with probability $\geq \frac{\epsilon}{t}$ and complexity $\leq t \cdot O(r)$.

$$Pr[A^{H,Sign(N,d,\cdot)}(N, e) = (M, S) : (H(M)) = S^e] \geq \epsilon$$

Without the loss of generality we assume that:

- A never makes the same random oracle query twice.
- A queries $H(M)$ before it requests a signature on a message M.
- If A outputs (M, S) then it had previously queried $H(M)$

We construct an algorithm A' which on input (N, e, y) where $y = X^e \bmod N$, finds X.

Algorithm A' is defined as:

- Pick $i \leftarrow \{1, ..., t\}$ randomly.
- Initialise datastructure that stores triples, initially empty.
- Simulate A:
  - When A makes its jth random oracle query $H(M_j)$
    - If $j = i$, answer the oracle query with y.
    - Otherwise, randomly pick X_j, compute ${X_j}^e \bmod N$, store $(M_j, X_j, {X_j}^e \bmod N)$ in the datastructure and answer the oracle query with $y_j = {X_j}^e \bmod N$
  - When A requests $Sign(M_k)$
    - If $k = i$ abort.
    - If $k \neq i$ look for $(M_k, X_k, {X_k}^e \bmod N)$ in the datastructure and answer the oracle query with X_k.
      (Note that we had made the assumption that A queries $H(M)$ before it requests a signature on a message M.)
- After A finishes, it outputs (M, S). If $M = M_i$ and $S^e = y \bmod N$, then output S as the required output X.

For each random oracle query, we are doing a RSA exponentiation operation of complexity r. So the complexity of A' would be at most complexity of A multiplied by $O(r)$ i.e. $t \cdot O(r)$.

The index i chosen by A' in the first step represents a guess as to which oracle query of A will correspond to the eventual forgery output by A. When the guess is correct, view of A as part of A' is distributed identically to the view of A alone. When A' guesses correctly and A outputs a forgery, then A' solves the given instance of RSA problem (because $S^e = y \bmod N$ and thus S is *inverse* of y). Since A' guesses correctly with probability $1/t$ and A outputs a forgery with probability $\geq \epsilon$. So, Probability with which A' breaks RSA $\geq \epsilon \cdot \frac{1}{t}$, which is what we intended to prove. $\square$