Handout 11: Forward Security

Boaz Barak

Total of 100 points + 20 bonus points.
Exercises due December 13th, 2005 1:30pm.
Last homework of the term!

Exercise 1 (60 points)**.** The notion of forward security has a natural generalization to *signature schemes*.

1. State informally the goals for a forward-secure digital signature scheme, and provide a formal definition for this notion.
2. Suppose that $(\mathsf{G}, \mathsf{Sign}, \mathsf{Ver})$ is a construction for standard (not forward-secure) chosen-message attack secure signature schemes, where the length of the private and public keys, and also the length of the signatures is n (and the length of messages to be signed can be arbitrary, i.e., works for any message $\{0,1\}^*$). Construct a forward-secure signature scheme with lengths of private and public keys is $O(n)$, and the length of each signature is at most $O(Tn)$, where T is the maximum number of time periods allows. (If it makes things simpler for you, the signature scheme you construct can be defined only for messages of length n.)
3. Under the same assumption, construct a signature scheme where the length of the public key and signatures is $O(n)$, but the length of the private key can be $O(T \cdot n)$.
4. (*20 point bonus question*) Can you construct a forward secure signature scheme with better parameters? (Ideally, we'd like private key, public key, and signature length to be at most $O(n \log T)$ or perhaps even $O(n + \log T)$).

Exercise 2 (40 points)**.** Another issue in protecting cryptographic keys, is how to make sure that they are generated at random. In a simplified scenario, suppose that you have two machines but one of them might be faulty — can you run an interaction between them that results in an unbiased coin toss?

More formally, a *secure coin-tossing protocol* is defined as follows: it is a two-party protocol with the two parties named Alice and Bob. There is a polynomial-time function *result* that takes as input the transcript of the protocol (i.e., the sequence of all messages exchanged between the two parties), and outputs a bit $b \in \{0,1\}$. Note that if the two parties are probabilistic, the transcript $\mathit{trans} = \mathsf{trans}\langle A(1^n), B(1^n)\rangle$ of the execution of the protocol (where both Alice and Bob are given as input the security parameter n) is a random variable. We require that as long as at least one party follows the protocol, for every $b \in \{0,1\}$, the probability that $\mathit{result}(\mathit{trans}) = b$ is between $\frac{1}{2} - \epsilon(n)$ and $\frac{1}{2} + \epsilon(n)$ where n is the security parameter for the protocol, and $\epsilon(\cdot)$ is a function such that $\epsilon(n) = n^{-\omega(1)}$.

An example for a simple protocol attempting to solve this problem would be for Alice to choose b at random and to send it to Bob, and for the result function to simply output b. This would work in the case that both Alice and Bob are honest (i.e., follow the protocol) and also if just Alice is honest, but not in the case that just Bob is honest. Hence this is not a secure coin-tossing protocol.

Construct a secure coin-tossing protocol.

See footnote for hint.[1]

[1]**Hint:** You might want to handle the case in which one party refuses to send a valid message, by defining the *result* function in such a way that in this case the other party gets to choose the output b.