COS 433 — Cryptography — Homework 9.

Boaz Barak

Total of 110 points. Due December 6th, 2007.

Exercise 1 (30 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 $trans = \mathsf{trans}\langle A(1^n), B(1^n)\rangle$ 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 $result(trans) = b$ 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]

Exercise 2 (40 points)**.** A *private coin tossing* protocol is the following variant of a coin tossing protocol. Intuitively, it supposed to be a protocol where only Alice actually knows the result of the coin toss, but Bob is still guaranteed that this result is random. (You can think that Alice learns the output b while the transcript only contains a commitment to b.)

More formally, again, there are two parties Alice and Bob and a function *result* on the transcript of the protocol, but this time *result* is *not* a polynomial-time function. We require the following properties of the protocol:

- As before, if at least one of the parties is honest, for every $b \in \{0,1\}$,
$$\frac{1}{2} - n^{-\omega(1)} < \Pr[result(trans) = b] < \frac{1}{2} + n^{-\omega(1)}$$
- If Alice is honest, then she knows the result: there is a polynomial-time algorithm RES that on input the private randomness and view that Alice used, outputs the same output as *result*. That is, no matter what algorithm B^* Bob uses $RES(\mathsf{view}_A\langle A, B^*\rangle) = result(\mathsf{trans}\langle A, B^*\rangle)$ where $\langle A, B^*\rangle$ in both sides of the equation denotes an execution between the honest Alice algorithm A and a possibly cheating (but polynomial-time Bob algorithm B^*).

[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.

- Bob does not know the result. That is, consider the following attack: Bob participates in the protocol using an arbitrary algorithm B^* where Alice uses the honest algorithm A, and at the end Bob outputs a guess b' for $result(trans)$, where $trans$ is the transcript of the execution. Then the probability that $b' = result(trans)$ is at most $1/2 + n^{-\omega(1)}$.

Note that if Alice and Bob participated in an execution of a private coin-tossing protocol with transcript $trans$, and Alice knows $b = result(trans) = RES(view)$ (where $view$ is her view including the random tape she used) then Alice can *prove* to Bob that $result(trans) = b$ by simply sending to Bob the random tape she used (using this random tape and $trans$ Bob can reconstruct Alice's view $view$ in the execution and then compute $result(trans) = RES(view)$).

Construct a private coin-tossing protocol and prove its security under the assumptions we learned in class.

Exercise 3 (40 points)**.** In this question, you will construct a protocol to allow Alice and Bob to play (a simplified version of) Black-Jack over the phone or net, without access to any physical deck of cards, and without needing to trust one another to follow the protocol.

We'll use the following version of Black Jack:

- There is a deck of 44 cards, with each card having value c between 1 and 11 and a type t which is a number between 1 and 4. That is, a card is a pair $\langle t, c \rangle \in [11] \times [4]$.
- There are two players Alice and Bob.
- The deck is shuffled randomly, and each player gets one card that is public and another card that only he/she can see.
- Based on that information, each player decides whether to ask for another card or not.
- At this point each player has either two or three cards. Each player reveals his/her cards and the total values of the cards for each player is computed, where for each player if this total is more than 21 then it is considered to be zero. The player with higher total value wins. If both totals are equal then it is a draw.

A *strategy* for a player in this game is a function $s : ([11] \times [4])^3 \to [0, 1]$ that is interpreted as follows: if the two public cards are $\langle c_1, t_1 \rangle$ and $\langle c_2, t_2 \rangle$ and the secret card that only the player sees is $\langle c_3, t_3 \rangle$ then the player will ask for a third card with probability $s(c_1, t_1, c_2, t_2, c_3, t_3)$. For two strategies s_A, s_B, we denote by a *real blackjack game* the result of running the process above where Alice uses strategy s_A and Bob uses strategy s_B.

We define a *secure blackjack protocol* to be a triplet $(A, B, result)$ where A, B are polynomial-time interactive algorithms and $result : \{0, 1\}^* \to \{\mathsf{Awins}, \mathsf{Bwins}, \mathsf{draw}\}$ is a polynomial-time computable function satisfying the following:

Inputs Each of the algorithms A,B takes as input a strategy $s : ([11] \times [4])^3 \to [0, 1]$ for the blackjack game. (You can assume that the output of s is always an integer multiple of $1/100$, so this strategy can be represented by a string of constant size.) Each algorithm also takes as input a string 1^n which we refer to as the security parameter.

Honest output If Alice and Bob both run the prescribed algorithms A and B with inputs s_A, s_B respectively and 1^n for every n, then for every $r \in \{\mathsf{Awins}, \mathsf{Bwins}, \mathsf{draw}\}$ the probability p_r that $result(trans) = r$ is equal to the probability that the corresponding outcome happens

in a real blackjack game where Alice uses s_A and Bob uses s_B (where *trans* is the random variable representing the transcript of an execution between $A(s_A, 1^n)$ and $B(s_B, 1^n)$.[2]

Bob can't win by cheating Suppose that Alice runs the honest algorithm A with strategy s_A, and suppose that Bob runs an arbitrary polynomial-sized computable strategy B^*. Denote by p the probability that $result(\mathsf{trans}\langle A(s_A, 1^n), B^*\rangle) \neq \mathsf{Awins}$ (where we assume n is sufficiently large). Then, there exists a strategy s_B for Bob such that with probability at least $p - 10^{-6}$, Alice does not win in a real blackjack game where she uses s_A and Bob uses s_B. That is, Bob could have done just as well in a real blackjack game, and hence Alice did not suffer any loss in the probability of winning because of Bob's not following the protocol.[3]

Alice can't win by cheating This is exactly the symmetric requirement for Bob. Suppose that Bob runs the honest algorithm A with strategy s_B, and suppose that Alice runs an arbitrary polynomial-sized computable strategy A^*. Denote by p the probability that $result(\mathsf{trans}\langle A^*, B(s_B, 1^n)\rangle) \neq \mathsf{Bwins}$ (where we assume n is sufficiently large). Then, there exists a strategy s_A for Alice such that with probability at least $p - 10^{-6}$, Bob does not win in a real blackjack game with strategies s_A and s_B.

Construct a blackjack protocol and prove its security under the assumptions we learned in class.

If it makes your life easier, you may allow the result function to have an additional output fail. However, we require that in all cases (both honest, Alice cheating, Bob cheating) the probability that the output is fail is at most 1/3. In all the requirements, we consider the conditional probabilities conditioned on the result different than fail. That is, in the case where both parties are honest we let p_r denote the probability that the result is r conditioned on the result different from fail. In the cases where Bob is cheating we let p denote the probability that the result is not Awins conditioned on the result different from fail, and we change in a similar way the requirement in the case that Alice is cheating.

[2] If it makes your life easier, you can relax the requirement that these probabilities are equal to the requirement that the absolute value of their difference is at most 10^{-6} for large enough n.

[3] For simplicity we didn't add the requirement that the probability of a draw will also be similar.