COS 433 - Cryptography - Final Take Home Exam

Boaz Barak

December 23, 2005

- Read these instructions carefully *before* starting to work on the exam. If any of them are not clear, please email me before you start to work on the exam.
- **Schedule:** You can work on this exam in a period of 96 hours of your choice between December 15^{th}, 2005 and January 17^{th}, 2005 at 12:00 pm. The exam needs to be submitted to my mail box by January 17^{th}, 2005 noon. *This is a strict deadline.* You may submit the exam earlier. If you typed up your exam, I would appreciate it if you also email me a copy of it at the same time. If you submit it through the mail box earlier than the deadline, please email me to let me know you did so, and email again if I don't acknowledge receipt.
- **Restrictions , honor code:** You should work on the exam alone. You can use your notes from the class, the homework exercises and their solutions, and the handouts I gave in class or put on the webpage. You can also use any personal summaries and notes of the material that you prepare before starting to work on the exam. *You should not use any other material while solving this exam.* You should write and sign the honor pledge on your submitted exam (the pledge is "I pledge my honor that I did not violate the honor code during this exam and followed all instructions").
- **Writing:** You should answer all questions *fully*, *clearly* and *precisely*. When describing an algorithm or protocol, state clearly what are the inputs, operation, outputs, and running time. When writing a proof, provide clear statements of the theorem you are proving and any intermediate lemmas or claims. I recommend that you first write a draft solution of all questions before writing (or preferably, typing) up your final submitted exam.
- **Partial solutions:** If there is a question you can not solve fully, but you can solve a partial/relaxed version or a special case, then please state clearly what is the special case that you can solve, and the solution for this case. You will be given partial credit for such solutions, as long as I feel that this special case captures a significant part of the question's spirit.
- **Polynomial security:** Whenever in this exam we say that a primitive is *secure* under some attack, we mean that it is $(T(n), \epsilon(n))$ secure for $T(n) = n^{\omega(1)}$ and $\epsilon(n) = n^{-\omega(1)}$, where n is the key size/security parameter of the scheme . That is, any adversary that is implementable by polynomial-sized circuit, and attacks the primitive has probability less than $1/poly(n)$ of success (for every polynomial $poly(\cdot)$ and large enough n). The conditions of the attack and meaning of success are of course determined by particular security definition.
- **Assumptions:** You may assume as true any of the axioms/assumptions that were given in class: existence of one-way permutations, commitment schemes, pseudorandom generators,

pseudorandom permutations, hardness of factoring random Blum integers, hardness of inverting the RSA permutation, decisional Diffie Hellman, existence of chosen-message (CMA) secure signature schemes, existence of collision-resistant hash functions, existence of chosen ciphertext (CCA) secure encryption schemes, and existence of secure oblivious transfer (OT) and private information retrieval (PIR) protocols. Whenever you use such an assumption, state it clearly and precisely. It is recommended that you review these assumptions and definitions before you start working on this test.

Note on the random oracle model: You can use the random oracle model, but it is preferred that you avoid doing so if possible (you will get at least the majority of points for a valid solution in the random oracle model). If you use as a black-box a CCA secure encryption scheme this does not count as using the random oracle model (even though the only construction we saw for this in class used the random oracle model).

- **Quoting results:** You can quote without proof theorems that were proven in class. However, you should quote them precisely, and state the date and lecture number where the theorem was proven. You can also use the hardcore theorem of Goldreich and Levin, even though it was not fully proved in class. When solving a question, you can use the results of a previous question as given, even if you did not manage to solve it.

- **Clarifications:** I have made an effort to make the questions as clear and unambiguous as possible. In case any clarifications are needed, I will try to be always available by email. You can also email me with your number and good times to call, and I will call you back. If you need me more urgently, you can call me at my cell phone 917-674-6110 between 11am and 10pm EST. You can also reach Dave at his email. Feel free to email me before starting to work on the exam to check my availability in that 96 hour period. If there are any unresolved doubts, please write your confusion as part of the answer and maybe you will get partial credit.

Turn the page only when you are ready to start working on the exam.

Question 1 (45 points). For each of the following statements, say whether it is *true* or *false*, and prove your assertion. You can consider as true all the assumption given in class as mentioned in the instructions on the first page. For example, if the statement is that there exists an object with some properties, then either prove that such an object exists by giving a construction based on the assumptions together with a proof that the construction satisfies the properties, or prove that such an object cannot exist.

1. There exists a pseudorandom generator $G = \{G_n\}$ where for every n, $G_n : \{0,1\}^n \to \{0,1\}^{2n}$ such that for every $x \in \{0,1\}^n$, the first $n/3$ bits of $G_n(x)$ are zero.

2. There exists a pseudorandom generator $G = \{G_n\}$ where for every n, $G_n : \{0,1\}^n \to \{0,1\}^{2n}$ such that for every $x \in \{0,1\}^n$, there exist $n/3$ bits of $G_n(x)$ that are zero.

3. There exists a pseudorandom generator $G = \{G_n\}$ where for every n, $G_n : \{0,1\}^n \to \{0,1\}^{2n}$ such that for every $x \in \{0,1\}^n$, if the first $n/3$ bits of x are zero then all the bits of $G_n(x)$ are zero (i.e., $G_n(x) = 0^{2n}$).

4. There exists a pseudorandom function collection $\{f_s\}_{s\in\{0,1\}^*}$ where, letting $n = |s|$, $f_s : \{0,1\}^n \to \{0,1\}^n$ that satisfies the following: for every $x \in \{0,1\}^n$, $f_{0^n}(x) = 0^n$ (i.e., $f_{0^n}(\cdot)$ is the constant zero function).

5. There exists a pseudorandom function collection $\{f_s\}_{s\in\{0,1\}^*}$ where, letting $n = |s|$, $f_s : \{0,1\}^n \to \{0,1\}^n$ that satisfies the following: for every $s \in \{0,1\}^n$, $f_s(0^n) = 0^n$.

6. For $\ell \geq 2$ and a string $x \in \{0,1\}^\ell$, let *cnot* : $\{0,1\}^\ell \to \{0,1\}^\ell$ be the following function: $\mathit{cnot}(x_1,\ldots,x_\ell) = x_1, x_2 \oplus x_1, x_3, \ldots, x_\ell$. (That is, *cnot* flips the second bit of x according to whether or not the first bit is one.) There exists a CPA-secure public key encryption scheme $(\mathsf{Gen}, \mathsf{E}, \mathsf{D})$ and a polynomial time algorithm A, such that for every n, if $(e,d) = \mathsf{Gen}(1^n)$ then for every $x \in \{0,1\}^\ell$ (where ℓ is the message size of the encryption scheme for security parameter n) it holds that
$$\mathsf{D}_d\Big(A\big(e, \mathsf{E}_e(x)\big)\Big) = \mathit{cnot}(x)$$

7. Same statement as Item 6 but with CPA-secure replaced with CCA-secure.

8. There exists a *commitment scheme* $C : \{0,1\} \times \{0,1\}^n \to \{0,1\}^m$ (i.e., a commitment for messages of length one bit, where to commit to the bit b, one chooses $r \leftarrow_{\mathrm{R}} \{0,1\}^n$ and sends $C(b,r)$) and a polynomial time algorithm $A : \{0,1\}^m \to \{0,1\}$, such that for both $b = 0$ and $b = 1$,
$$\Pr_{U_n}[A\big(C(b,U_n)\big) \oplus b = 1] \geq 2/3$$
where U_n denotes the uniform distribution over $\{0,1\}^n$.

9. Same statement as Item 8 but with 2/3 replaced with 1/2.

Question 2 (25 points)**.** Recall the definition of a secure coin-tossing protocol: 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 $\epsilon(n) = n^{-\omega(1)}$.

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 $b \in \{0,1\}$,
$$\tfrac{1}{2} - n^{-\omega(1)} < \Pr[result(trans) = b] < \tfrac{1}{2} + n^{-\omega(1)}$$
- 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^*).
- 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 $result(trans) = RES(view)$).

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

Question 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 $\mathit{result} : \{0,1\}^* \to \{\mathsf{Awins}, \mathsf{Bwins}, \mathsf{draw}\}$ 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 $\mathit{result}(\mathit{trans}) = r$ 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 $B(s_B, 1^n)$.[1]

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 $\mathit{result}(\mathsf{trans}\langle A(s_A, 1^n), B^* \rangle) \neq \mathsf{Awins}$ (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.[2]

[1] 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.

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

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.

The question: 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.