COS 433 - Cryptography - Final Take Home Exam

Boaz Barak

May 12, 2010

- 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 48 hours of your choice between Monday May 3rd 2010 and Friday May 14th 2010 5pm. (i.e., you need to submit the exam either 48 hours after you downloaded it from the website or by Friday May 14 5pm, whichever comes sooner.) *This is a strict deadline.* You may submit the exam earlier.

  Please *type up* the exam, and submit it (as a pdf, postscript, or word doc file) to Boaz, Sushant and Shi by email. If you have a problem with typing up the exam, then please let me know as soon as possible. In this case you should submit *two copies* of the handwritten exam to my mailbox, and also email me, Sushant and Shi at the same time to let us know that you've done so.
- **Restrictions, allowed texts, 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 the Boneh-Shoup, Katz-Lindell, Goldreich and Arora-Barak textbooks as well as Trevisan's lecture notes. You can also use any personal summaries and notes of the material that you prepare on your own or with a group before starting to work on the exam. *You should not use any other material while solving this exam.* When you submit the exam by email, you should also include in the email the honor pledge (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 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.
- **Assumptions:** Unless explicitly said otherwise, you may assume as true any of the axioms/assumptions that were given in class such as: existence of one-way functions and permutations, commitment schemes, pseudorandom generators, functions and 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 a fully homomorphic encryption scheme. 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 unless the question states otherwise, 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 public key 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 or given as a homework exercise. However, you should quote them precisely, and state the date and lecture number or the homework number and question. You *cannot* quote without proof any other results— this includes results that were only stated without proof in class, and results that are proven in the textbooks but not in class. If you do use a proof from another source such as a textbook, you should write the proof in full but also add a reference to the place it appears. 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 609-981-4982 between 11am and 10pm EST. You can also email Sushant and Shi as well. If there are any unresolved doubts, please write your confusion as part of the answer and maybe you will get partial credit.

This exam has 4 questions, with a total of 135 points.

Question 1 (30 points)**.** Consider the following primitives: **(1)** one-way function, **(2)** one-time secure digital signature scheme, **(3)** CPA secure public key encryption, **(4)** fully homomorphic CPA-secure private key encryption, **(5)** a chosen ciphertext secure (CCA) private key encryption that for a key of length n handles messages of length $10n$ and the encryption algorithm uses n bits of randomness.

For each pair $\{i, j\}$ of these primitives, say whether the existence of i implies the existence of j, and/or vice versa. If you do not know whether an implication holds, then say so. (This is OK, as some of these implications are open questions.)

For any implication you claim to hold, either give a proof or quote the lecture in class in which it was proven (and explain exactly how what was proven in class yields the implication). For this question you cannot use any unproven assumption. As a reminder, a polynomial-time computable function $F : \{0,1\}^* \to \{0,1\}^*$ is a *one-way function* if there does not exists a polynomial time algorithm A and a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for infinitely many n's, $\Pr_{x \leftarrow_R \{0,1\}^n}[A(F(x)) = x' \text{ s.t. } F(x') = F(x)] \geq 1/p(n)$. You can use the result we stated in class without proof that if there exists a one-way function then there exists a pseudorandom generator with output length larger than the input length.

As a hint, there are 6 basic implications known between the above primitives, and the rest of the known implications are derived from these using transitivity. One of these six is harder than the other five, and so you may want to look at the other questions first after you show five implications.

Question 2 (30 points)**.** (This question is very long since I tried to use very precise notation, but I don't think it's very hard.)

A *3CNF formula* is a formula on taking variables in $\{0,1\}$ that is of the form of an AND of OR's, where each OR depends on three variables or their negation. More formally, φ is a 3CNF variable on n variables $x_1, \ldots, x_n$ if it is of the form $C_1 \wedge \cdots \wedge C_m$ where each C_i (known as a *clause*) has the form $(y_{i,1} \vee y_{i,2} \vee y_{i,3})$ and each $y_{i,j}$ (known as a *literal*) is equal to either x_ℓ or $\neg x_\ell$ for some $\ell \in [n]$. (Recall that $\wedge$ is AND, $\vee$ is OR, and $\neg$ is NOT.) For $x \in \{0,1\}^n$, we denote by $\varphi(x) \in \{0,1\}$ the value of φ when its variables are assigned the value x (identifying 0 with "false" and 1 with "true").

In this question we consider the following protocol, where the public input is a 3CNF formula φ with m clauses and n variables, and the prover's private input is $x \in \{0,1\}^n$. You will prove that it is a zero knowledge protocol for the statement "$\exists_x$ s.t. $\varphi(x) = 1$".

Public input: 3CNF formula φ on n variables and m clauses. We assume every variable appears in at least one clause, and that $m \geq n$.

Prover's (Alice) private input: $x \in \{0,1\}^n$ such that $\varphi(x) = 1$.

Prover's first message: Loosely speaking, in this step the prover will encode every variable x_ℓ by a random string $a_\ell \in \{0,1\}^3$ such that $a_{\ell,1} \oplus a_{\ell,2} \oplus a_{\ell,3} = x_\ell$ and will send commitments to the encoding of (a randomly reordered version of) the literals of each clause. We now explain precisely what the prover does:

1. For every variable x_ℓ, the prover chooses at random $a_{\ell,1}, a_{\ell,2} \in \{0,1\}$ and sets $a_{\ell,3} = x_\ell \oplus a_{\ell,1} \oplus a_{\ell,2}$.

2. The prover reorders randomly the three literals of every clause. We let π be a string of length $100m$ that describes this reordering (we can think of π as a tuple of m permutations over $[3]$, where π_i denotes the i^{th} permutation). The prover computes $\hat{\pi} = \mathsf{Com}(\pi)$, where Com is a commitment scheme (see below for a reminder of definition and notation).
3. For every clause C_i, let $y_{i,1}, y_{i,2}, y_{i,3}$ be the three literals (in the new order) of C_i, where each literal is a variable or its negation. For $j \in [3], k \in [3]$, the prover lets $c_{i,j,k} = a_{\ell,k}$ if the literal $y_{i,j}$ is equal to the variable x_ℓ, and she lets $c_{i,j,k} = \neg a_{\ell,k}$ if the literal is the negation $\neg x_\ell$. For $i \in [m], j, k \in [3]$, the prover computes $\hat{c}_{i,j,k} = \mathsf{Com}(c_{i,j,k})$.
4. The prover sends the commitments $\hat{\pi}$ and $\{\hat{c}_{i,j,k}\}_{i \in [m], j,k \in [3]}$ to the verifier.

Verifier's (Bob) message The verifier tosses a coin $a \leftarrow_{\mathrm{R}} \{0,1\}$ and:

- If $a = 0$ then Bob chooses at random $\ell \leftarrow_{\mathrm{R}} [n]$ and chooses at random two literals $y_{i,j}$ and $y_{i',j'}$ (where $i, i' \in [m]$ and $j, j' \in [3]$) that refer to the variable x_ℓ. (These are literals of φ in the original ordering.) Bob then sends k at random in $[3]$ and sends $(0, i, j, i', j', k)$ to Alice. Intuitively, the meaning of this message is that Bob wants to verify that the encodings of the two literals are consistent.
- If $a = 1$ them Bob chooses at random $i \leftarrow_{\mathrm{R}} [m]$, and sends $(1, i)$ to Alice. Intuitively, the meaning of this message is that Bob wants to verify that the clause i has at least one literal with value 1.

Prover's (Alice) second message Upon receiving the message from Bob, Alice does the following:

- If the message was $(0, i, j, i', j', k)$ then Alice sends to Bob the randomness used in the commitment $\hat{\pi}$, and the randmoness used in the commitments $\hat{c}_{i,\pi_i(j),k}$ and $\hat{c}_{i',\pi_{i'}(j'),k}$ (where we denote by $\pi_i(j)$ the new order in $[3]$ of the j^{th} literal of the i^{th} clause under π).
- If the message was $(1, i)$ then Alice lets $j \in [3]$ be such that $y_{i,j} = 1$ (this is in the new ordering, note that there must be at least one such literal, since $\varphi(x) = 1$— if there is more than one then Alice chooses such a literal at random), and sends the randomness to the commitments $\hat{c}_{i,j,1}$, $\hat{c}_{i,j,2}$, $\hat{c}_{i,j,3}$.

Bob's decision On receiving Alice's message, Bob does as follows depending on the value a he selected earlier:

- If $a = 0$, Bob checks that the randomness he receives is consistent with the commitment $\hat{\pi}$, that π encodes a proper reordering (i.e., m permutations of $[3]$) and that Alice did indeed send valid openings to the commitments $\hat{c}_{i,\pi_i(j),k}$ and $\hat{c}_{i',\pi_{i'}(j'),k}$. Let c and c' be respectively the decommitted values of these commitments. Then if the literals $y_{i,j}$ and $y_{i,k}$ have the same sign (i.e., they are both of the form x_ℓ or they are both of the form $\neg x_\ell$), then Bob checks that $c = c'$. Otherwise he checks that $c =\neq c'$. If all checks pass then Bob accepts the proof, otherwise he rejects it.
- If $a = 1$, Bob checks that the randomness he receives is consistent with the commitments $\hat{x}_{i,j,1}, \hat{x}_{i,j,2}, \hat{x}_{i,j,3}$ (where $i \in [m]$ is the clause index Bob chose in his message, while j is a number in $[3]$), and that the XOR of the decommitted values is equal to 1. If all checks pass then Bob accepts the proof, otherwise he rejects it.

Prove that the above is a zero knowledge protocol for 3SAT statements. That is, prove:

1. (5 points) *(Completeness)* If $\varphi(x) = 1$ and both verifier and prover follow the protocol, then the verifier will accept the proof with probability one.

2. (15 points) *(Soundness)* If φ is a formula such that $\varphi(x) = 0$ for all $x \in \{0,1\}^n$, then no matter what the prover strategy is, the verifier will reject the proof with probability at least $1/(100m^2)$.

3. (5 points) (*(Honest verifier zero knowledge)* There is a probabilistic polynomial time algorithm SIM such that if we let $R(\varphi, x)$ be the random variable consisting of the verifier's random tape and messages received in a real interaction where both verifier and prover follow the protocol on inputs φ, x then $R(\varphi, x)$ is computationally indistinguishable from the random variable $SIM(\varphi)$.

4. (10 points) *(Full-fledged zero knowledge)* For every polynomial-time (possibly cheating) strategy V^* for the verifier, there is a probabilistic polynomial time algorithm $SIM*$ such that if we let $R_{V^*}(\varphi, x)$ be the random variable consisting of the verifier's random tape and messages received in a real interaction where both the prover follows the protocol and the verifier uses V^* on inputs φ, x then $R_{V^*}(\varphi, x)$ is computationally indistinguishable from the random variable $SIM^*(\varphi)$. (Note that this subsumes the previous item, so if you answer this item, you can skip the previous one.)

Reminder— definition of commitment: for simplicity one can think of $\mathsf{Com} : \{0,1\}^{n+1} \times \{0,1\}^{n+1}$ as a function with the following properties: **Binding:** Com is one to one and **hiding:** $\mathsf{Com}(b, U_n)$ is computaitonally indistinguishable from $\mathsf{Com}(b', U_n)$ for every $b, b' \in \{0,1\}$, where U_n denotes the uniform distribution over $\{0,1\}^n$. For $b \in \{0,1\}$, we denote by $\mathsf{Com}(b)$ the random variable $\mathsf{Com}(b, U_n)$ and for $z \in \{0,1\}^m$ we denote by $\mathsf{Com}(x)$ the concatenation $\mathsf{Com}(x_1) \cdots \mathsf{Com}(x_m)$.

Question 3 (45 points)**.** In the construction of fully homomorphic encryption we needed the notion of *circular security*. In this question we explore it further:

1. (10 points) Say that a private key encryption (E, D) with key size = plaintext size = n is *CPA'-secure* if for every polynomial-time adversary A there is a negligible function μ such that A cannot distinguish with success better than $1/2 + \mu(n)$ between the following two cases: **(i)** k is chosen at random in $\{0,1\}^n$ and A is given 1^n as input and access to a box that on input $x \in \{0,1\}^n$ outputs $E_k(x)$ and **(ii)** k is chosen at random in $\{0,1\}^n$ and A is given 1^n as input and access to a box that on input $x \in \{0,1\}^n$ outputs $E_k(0^n)$. (We stress that in both cases, the box uses fresh randomness in each invocation.)

   Prove that (E, d) is CPA' secure if and only if it is CPA secure.

2. Say that a private key encryption (E, D) with key size = plaintext size = n is *KDM-secure* (for *key dependent messages*) if for every polynomial-time adversary A there is a negligible function μ such that A cannot distinguish with success better than $1/2 + \mu(n)$ between the following two cases: **(i)** k is chosen at random in $\{0,1\}^n$ and A is given 1^n as input and access to a box that on input a Boolean circuit C mapping n bits to n bits outputs $E_k(C(k))$ and **(ii)** k is chosen at random in $\{0,1\}^n$ and A is given 1^n as input and access to a box that on input a Boolean circuit C mapping n bits to n bits outputs $E_k(C(0^n))$.

We say that (E, D) is *circular secure* if it is CPA secure and in addition satisfies KDM security as above in the following restricted sense— we only require indistinguishability for the case that the adversary A is restricted to send to its box a Boolean circuit C that is either the identity function (i.e., $C(k) = k$ for all k), or a constant function (i.e. there is some $x \in \{0,1\}^n$ such that $C(k) = x$ for every $k \in \{0,1\}^n$). (In both cases we fix some canonical polynomial size representation of the circuits for the identity and constant functions, and A must use these representations.)

(a) (7 points) Show that there exists a CPA secure encryption scheme that is not circular secure.

(b) (8 points) Show that there exists a CPA circular secure encryption scheme that is not KDM secure.

(c) (10 points) Show that the following encryption scheme is KDM secure *in the random oracle model*: we have a random oracle $H : \{0,1\}^{2n} \to \{0,1\}^n$ and the key is a random string $k \in \{0,1\}^n$. To encrypt $x \in \{0,1\}^n$ we set $E_k(x) = (r, H(k \circ r) \oplus x)$, where r is chosen at random in $\{0,1\}^n$ and $\circ$ denotes concatenation. To decrypt (r, y) we set $D_k(r, y) = y \oplus H(k \circ r)$. KDM security in the random oracle model is defined in the same way, except that the adversary also has black-box access to the random oracle H.

(d) (10 points) Show that if (E, D) is a circular secure private key encryption scheme that is also *fully homomorphic* then it is also KDM secure.

For this question we'll make a slightly simplified and strengthened definition of fully homomorphic encryption: say that a CPA-secure private key scheme (E, D) with key-size = plaintext-size = n is *fully homomorphic* if there exists a probabilistic polynomial-time algorithm $EVAL$ such that for every key $k \in \{0,1\}^n$ and Boolean circuit $C : \{0,1\}^n \to \{0,1\}^n$ and every ciphertext X such that X is in the range of $E_k(x)$ for some $x \in \{0,1\}^n$ (i.e., there is some randomness r s.t. $X = E_k(x; r)$), $EVAL(C, X)$ is distributed identically to the distribution of $E_k(C(x))$ with independent randomness. We stress that $EVAL$ does *not* get the private key k as input. Also note that $EVAL(C, X)$ is a random variable depending on the randomness used $EVAL$, while C, X are fixed.

Question 4 (30 points)**.** Consider a key exchange protocol where the client has the public keys of a server, chooses a key $k \leftarrow_{\mathrm{R}} \{0,1\}^n$ for a private key scheme, interacts with the server, and at the end decides whether or not to accept the key as valid. For simplicity in this question we restrict ourselves to *sequential* attacks— the adversary can modify the messages from the client to server, but cannot start new interactions in the middle. That is, the attack game has the following form— we choose $b \leftarrow_{\mathrm{R}} \{0,1\}$ and do the following:

1. For a polynomial number of interactions, the client and server run the protocol sequentially. The adversary sees and can also modify any message sent by the client or the server to a different message of its choice, including dropping the message, in which case we assume both client and server abort this particular interaction.

2. Whenever the server accepts a key k, he outputs $\mathsf{E}_k^{\mathsf{priv,cca}}(0)$, whenever a client accepts a key k, it outputs $\mathsf{E}_k^{\mathsf{priv,cca}}(b)$.

3. The adversary outputs $b' \in \{0,1\}$. We say the adversary is *successful* if $b' = b$.

We say the protocol is *secure* if the probability the adversary succeeds in this attack is at most $\frac{1}{2} + n^{-\omega(1)}$.

Notation: We denote by $(\mathsf{Sign}, \mathsf{Ver})$ a secure signature scheme. We denote by $\mathsf{E}^{\mathsf{pub,cca}}$ a CCA secure public key encryption scheme, by $\mathsf{E}^{\mathsf{pub,cpa}}$ a CPA secure public key encryption scheme, and by $\mathsf{E}^{\mathsf{priv,cca}}$ a CCA secure private key encryption scheme. The protocol is secure if it is secure for *every* suitable choice of the underlying schemes. In all cases we denote by e and by v the public encryption key and verification key of the server, and assume that the client knows them.

For each of the following protocols, either prove that it is secure (for *every* suitable choice of the schemes) or give an example showing it is insecure (for *some* choice of the schemes).

Protocol 1:

- Client chooses $k \leftarrow_{\mathrm{R}} \{0,1\}^n$ and sends $y = \mathsf{E}_e^{\mathsf{pub,cca}}(k)$. Client immediately accepts k.
- Server receives message y, and decrypts it to obtain k. If decryption was valid then he accepts k.

Protocol 2: Same as Protocol 1 replacing $\mathsf{E}^{\mathsf{pub,cca}}$ with $\mathsf{E}^{\mathsf{pub,cpa}}$.

Protocol 3:

- Client chooses $k \leftarrow_{\mathrm{R}} \{0,1\}^n$ and sends $y = \mathsf{E}_e^{\mathsf{pub,cpa}}(k)$.
- Server receives message y, and decrypts it to obtain k. If decryption was valid then he accepts k. Server sends $t = \mathsf{Sign}_s(y)$ to Client.
- Client receives message t from server, and accepts k if and only if $\mathsf{Ver}_v(y,t) = 1$.