COS 433 — Cryptography — Homework 4.

Boaz Barak

Total of 110 points. Due October 18th, 2007.

Exercise 1 (30 points)**.** Prove that the following encryption scheme is CCA secure. Let $\{p_k\}$ be a collection of pseudorandom permutations mapping $\{0,1\}^{3n}$ to $\{0,1\}^{3n}$.

- To encrypt $x \in \{0,1\}^n$ with key k do the following: choose $r \leftarrow_{\mathrm{R}} \{0,1\}^n$, and send $p_k(x\|r\|0^n)$ (were $\|$ denotes concatenation).
- To decrypt $y \in \{0,1\}^{3n}$, compute $x\|r\|w = p_k^{-1}(y)$. if $w \neq 0^n$ then output $\perp$. Otherwise, output x.

Exercise 2 (40 points)**.** For each of the following statements either prove that it is true, or give a counterexample showing that it is false:[1]

1. A MAC tag always maintains secrecy of the message. That is, if $(\mathsf{Sign}, \mathsf{Ver})$ is a CMA-secure MAC with m-bit long messages and n-bit long keys, then for every two strings x and x' in $\{0,1\}^m$, the random variable $\mathsf{Sign}_{U_n}(x)$ is computationally indistinguishable from the random variable $\mathsf{Sign}_{U_n}(x')$.
2. A MAC tag always has to be longer than the message. That is, for every MAC scheme $(\mathsf{Sign}, \mathsf{Ver})$, $|\mathsf{Sign}_k(x)| \geq |x|$.
3. Reusing a key for authentication and encryption does not harm secrecy: Suppose that $(\mathsf{Sign}, \mathsf{Ver})$ is a secure MAC with n bit key and (E, D) is a CPA-secure encryption scheme with n bit key. Suppose that a sender chooses $k \leftarrow_{\mathrm{R}} bits^n$ and a random number $x \leftarrow_{\mathrm{R}} 1, \ldots, 100$, computes $y = \mathsf{E}_k(x)$ and sends $y, \mathsf{Sign}_k(y)$ (note that the same key k is used for both authentication and encryption). Then, *secrecy* is preserved: an eavesdropper can not guess x with probability higher than, say $1/99$.
4. Reusing a key for authentication and encryption does not harm secrecy if we use a pseudorandom generator. Suppose that G is a PRG mapping $\{0,1\}^n$ to $\{0,1\}^{2n}$ and in the scenario above the sender after choosing the key k first computes $k_1k_2 = \mathsf{G}(k)$ (i.e., k_1 denotes the first n bits of the PRG's output and k_2 denotes the second n bits) and then uses k_1 for the encryption and k_2 for the MAC. Then, the eavesdropper can not guess x with probability higher than, say $1/99$.

Exercise 3 (40+10 points)**.** An *encrypted file system* is used to ensure that theft or unauthorized access to a laptop or desktop computer will not cause any compromise of sensitive data. The idea is that there is a secret key k on a smartcard, and this key is required to read and write to the hard disk. Formally, the interface to such a system is the operations:

[1] Counterexamples can be contrived as long as they are valid. That is, if a statement says that every MAC scheme satisfies a certain property then to show this statement false you can present *any* chosen-message attack secure MAC scheme that does not satisfy this property. The MAC scheme can be constructed just for the sake of a counterexample, and does not have to be "natural looking", as long as it is chosen-message attack secure.

`writeBlock`$_k$`(`i`,`x`)` Write $x \in \{0,1\}^m$ to the i^{th} block of the hard disk using the secret key k. We let M denote the total number of blocks.

`readBlock`$_k$`(`i`)` Returns a string in $\{0,1\}^m$, which is the (decrypted) contents of the i^{th} block of the hard disk. We assume that if the system detects that this block was tampered with then it shuts down the computer.

Intuitively, the security of the system should be as follows: suppose that each night, after using the computer normally (word processing, internet, email etc..) for the day, the user of the computer leaves home with her smartcard, and then an attacker has complete access to the computer (i.e., able to read and write directly to the hard disk). Then, the attacker should not be able to learn anything about the contents. Of course the attacker can "wipe out" the hard disk, in which case the system will detect this and shut the computer down, but we do not consider this a break of the system.

1. Suppose that we are only interested in preserving the *secrecy* of the data on the hard disk. is it still important to prevent an attacker from *modifying* the contents of a block on the hard disk without being detected?

2. Write a formal definition for security of an encrypted file system scheme.

3. Give a construction for an encrypted file system. That is, give algorithms for `writeBlock` and `readBlock`. You can assume that you have access to the functions `directWrite`(i, y) and `directRead`(i) that allow you to directly read and write blocks of the underlying hard disk. We denote the block size of the underlying disk by m' and the total number of blocks by M'. The numbers m and M (defining the block size and number of blocks you present to the user) are given to you, but you can choose m' and M' to be any values of your choice. Try to minimize the *overhead* $M'm' - Mm$ (that is, the difference between the number of bits you allow the user to use and the number of bits you actually need in the hard disk).

4. Prove that your construction remains secure in the following two attack scenarios (for 10 points bonus - do this by first proving that your construction satisfies your definition of Item 2, and then proving that any construction satisfying that definition remains secure under these attacks).

   (a) User chooses x to be a random number between 1 and 1000 and writes it in the first block (i.e., runs `writeBlock`$_k$(1,x)). The attacker then gets access to the hard disk and outputs a guess x'. System is secure under this attack if the probability that $x' = x$ is less than 1/999 (for large enough n).

   (b) User chooses x to be a random number between 1 and 1000 and writes it in the first block (i.e., runs `writeBlock`$_k$(1,x)) and writes the number 1 to the second block (i.e., runs `writeBlock`$_k$(2,1)). The attacker then gets access to the hard disk. User then reads the value y of the second block and publishes it on the web (where everyone including the attacker can see it). Attacker then gets again access to the hard disk and outputs a guess x'. System is secure under this attack if the probability that $x' = x$ is less than 1/999 (for large enough n).