Handout 4: Message Authentication Codes and Chosen-Ciphertext Security

Boaz Barak

Total of 109 points.
Exercises due October 18th, 2005 1:30pm.

Exercise 1 (20 points)**.** Consider the following variant of CMA-security for MACs: instead of giving the adversary black boxes for both the signing and verification algorithms, give it only a black box for the signing algorithm. Let's call this definition CMA'-security. That is,

> **Definition 1.** A pair of algorithms $(\mathsf{Sign}, \mathsf{Ver})$ (with $\mathsf{Sign} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^t$, $\mathsf{Ver} : \{0,1\}^n \times \{0,1\}^m \times \{0,1\}^t \to \{0,1\}$) is a (T, ϵ)-CMA'-secure MAC if for every x, k, $\mathsf{Ver}_k(x, \mathsf{Sign}_k(x)) = 1$ andf or every T-time Adv, if we run the following experiment:
>
> - Choose $k \leftarrow_{\mathrm{R}} \{0,1\}^n$
> - Give adversary access to black box for $\mathsf{Sign}_k(\cdot)$
> - Adversary *wins* if it comes up with a pair $\langle x', s' \rangle$ such that **(a)** x' is *not* one of the messages that the adversary gave to the black box $\mathsf{Sign}_k(\cdot)$ and **(b)** $\mathsf{Ver}_k(x', s') = 1$.
>
> Then the probability Adv wins is at most ϵ.
>
> $(\mathsf{Sign}, \mathsf{Ver})$ is CMA'-secure if there are super-polynomial functions T, ϵ such that for every n, $(\mathsf{Sign}, \mathsf{Ver})$ is $(T(n), \epsilon(n))$-CMA'-secure. In other words, there is no polynomial-time Adv that succeeds with polynomial probability to break it.

A MAC scheme has *unique tags* if for every message there is only one tag that passes verification. An equivalent way of stating this property is that the verification algorithm on input x and t outputs 1 if and only if $S_k(x) = 1$. Note that the MAC scheme we saw in class has this property. Prove that for MACs with unique tags, CMA security and CMA' security are *equivalent* (e.g., such a scheme is (T, ϵ)-CMA secure if and only if it is (T', ϵ')-CMA' secure for some T',ϵ' polynomially related to T, ϵ. (The condition of unique tags is important — if a MAC scheme does *not* have unique tags then these notions may not be equivalent.)

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 $\mathsf{Sign}_{U_n}(x')$.

[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.

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. A CMA-secure MAC scheme has to be *probabilistic* that is, $\mathsf{Sign}_k(x)$ can't be a deterministic function of k and x and has to toss additional coins.

4. 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 $1/99$.

5. Reusing a key for authentication and encryption does not harm integrity: In the same setting as the previous item, *integrity* is preserved. That is, if the receiver obtains (y, t), where $\mathsf{Ver}_k(y, t) = 1$ and computes $x' = \mathsf{D}_k(y)$ then $x = x'$.

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:

`writeBlock`$_k$`(`i`,`x`)` Write $x \in \{0,1\}^m$ to the i^{th} block of the hard disk. i is a number between 1 and M. (It is assumed that k will be used to protect the contents of the hard disk)

`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 (T, ϵ)-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). If you can, try to explicitly write the overhead as a function of the other parameters (T, ϵ, m, M). You can assume that $T = 1/\epsilon > M > m$. Also, if it makes your life simpler, you can assume that any primitives you use (pseudorandom functions, permutations, generators) are $(2^{n/10}, 2^{-n/10})$-secure.

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 $\texttt{writeBlock}_k(1,x)$). 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 $\texttt{writeBlock}_k(1,x)$) and writes the number 1 to the second block (i.e., runs $\texttt{writeBlock}_k(2,1)$). 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).

I know this exercise may be too short for those already addicted to infinitely long crypto homework :) If you find yourself in need of more questions, for additional 10 points you can submit with this handout the answer to Exercise 9 (the back-door cipher question) (or a revised answer if you already submitted this).