CS 161 Fall 2005 — Computer Security — Joseph/Tygar/Vazirani/Wagner — HW 2

Due Friday, October 14 at 11am

Please include the following at the top of the first page of your homework solution:

Your full name
Your login name
The name of the homework assignment (e.g. hw3)
Your TA's name

Staple all pages together, and drop them off in drop box #2 (labeled CS161/Fall 2005) in 283 Soda by 11am on the due date.

Homework exercises:

1. **(4 pts.) Any questions?**
   What's the one thing you'd most like to see explained better in lecture or discussion sections? A one-line answer would be appreciated.

2. **(20 pts.) PGP**
   In this question, you will learn how to use PGP, a popular format for email encryption. Of course, plain email is totally insecure: it is very easy to send forged email with a spoofed From: address, and it is often easy to intercept (eavesdrop) on email. PGP was first built by Phil Zimmerman as a way for activists to communicate securely over the internet. Over time, it has evolved into an open standard, called OpenPGP. There are several programs that support this message format: PGP is made by PGP Corporation; `gpg` is an open-source Gnu implementation. They can all interoperate.

   The instructional machines already have `gpg` installed, so the following instructions assume that you will use `gpg` on the instructional machines. Note that you can get documentation on how to use `gpg` by simply typing `man gpg`. The instructional web pages also have a more detailed tutorial at `http://www-inst.eecs.berkeley.edu/cgi-bin/pub.cgi?file=gpg.help`.

   (a) Generate a new key pair. With `gpg`, you can use `gpg --gen-key`. This will construct a public key and a private key. Make sure your keysize is at least 1024 bits.

   Then, export your public key into ASCII-armored format, and save the result to the file `pubkey.asc` in your home directory. With `gpg`, you can use `gpg --export --armor > ~/pubkey.asc`. Make sure that the resulting public key is world-readable: `chmod a+r ~/pubkey.asc`.

   (b) Extract the fingerprint of your public key. With `gpg`, the fingerprint is printed out when you generate the key, or you can use `gpg --fingerprint`. The finger print is a long string of about 40 hex digits that can be used to uniquely identify your public key. It is generated by applying a collision-resistant hash function to your public key, so no two public keys will have

the same fingerprint. This is very useful: if I want to send you my public key, I can email you my public key (or send it to you over any insecure channel), and then we can compare fingerprints over a secure channel. For instance, I might print my fingerprint on my business card, or you can telephone me and I can read off my fingerprint out loud.

Write down your fingerprint as the answer to this question.

(c) Get your TA's public key, and import it into your "keyfile". Your TA's public key will be stored in `~cs161/pubkeys/ta-name.asc` on the instructional machines. (Replace `ta-name` with the last name of your TA.) With `gpg`, you can import it into your keyfile using `gpg --import ~cs161/pubkeys/whoever.asc`.

IMPORTANT: Check the fingerprint of the key you just imported. You can view the fingerprint of the key you just imported using `gpg --fingerprint`. Here are the fingerprints for the TA's public keys:

```
F40B 92ED F74C 98A6 427E  CEFD 6157 099E 7FCF 6767    Paul Huang
04B1 4409 6161 7094 77CD  9AC2 52BF 4AA6 EF37 EB53    Jeff Kalvass
8C67 1E41 CB1E 9D41 B008  E256 EF63 3EBB 7160 8841    Russell Sears
9512 62C0 789F EC33 26DB  CA4E 53C6 949D 4FF9 E26E    Ivan Tam
```

Verifying fingerprints is important; otherwise, Mallet (the malicious attacker) might be able to trick you into accepting a key pair she generated as your TA's key, and then Mallet would be able to read all the encrypted mail you send to your TA.

(d) Once you have verified your TA's public key, sign it. With `gpg`, you can use `gpg --sign-key name`, where `name` is a string used to identify your TA's public key (you can use your TA's fingerprint, typed as a hex string with no spaces, or part of your TA's real name).

Then, export the signed version of your TA's public key, and save the result into the file `~/ta.asc` in your home directory. With `gpg`, you can use `gpg --armor --export name > ~/ta.asc`, where again `name` is a string that identifies your TA's public key.

(e) Finally, compose and send an encrypted, signed email to your TA. Create a text file containing a very short message to your TA, and include your name. This is the cleartext. Now encrypt it with your TA's public key, and sign it with your private key, and save it in ASCII-armored format, and email the resulting ciphertext to your TA. With `gpg`, you can use `gpg --encrypt --sign --armor --recipient name < msg > msg.asc`, where `name` is a string identifying your TA, to generate the ciphertext. Once you've generated the ciphertext, email it to your TA.

3. (10 pts.) One-time pad

(a) Suppose you want to encrypt a message $M \in \{0,1,2\}$ using a shared random key $K \in \{0,1,2\}$. Suppose you do this by representing K and M using two bits (00, 01, or 10), and then XORing the two representations. Does this scheme have the security guarantees of the one-time pad? Explain.

(b) Give an alternate encryption algorithm for carrying out the above task that does provide strong security guarantees.

Note: You must not change the message space $\{0,1,2\}$ or the key space $\{0,1,2\}$. Instead, we want you to design an encryption algorithm $E(\cdot,\cdot)$ so that $E(K,M)$ is a secure encryption of M, when K and M are distributed as above.

4. (10 pts.) An RSA reduction

For this question, assume we use RSA with $e = 3$. Suppose the adversary with knowledge of the RSA

public key (N,e) can determine the private key d. Show that she can use this information to quickly factor N. How fast is your algorithm?

5. (21 pts.) The definition of a secure block cipher

The goal of this problem is to help you work through the definition of what it means to be a secure block cipher.

Suppose we have a block cipher with key length k and block length n. Recall the definition of a secure block cipher: it is that $\mathrm{Adv}A \leq \varepsilon$, for all algorithms A running in time at most T. In other words, we perform the following experiment: the adversary is given a box which contains either (I) the block cipher with a random key or (II) a uniformly random permutation on n bits. Eve is allowed T time in which to play with the box to guess whether it type I or type II. The advantage of the adversary, A, is the absolute value of the difference between the to be $\mathrm{Adv}A = |p-q|$, where p is the probability that the adversary guesses box I when the box she is given is actually of type I, and q is the probability that the adversary guesses box I when the box she is given is actually of type II. Formally, $p = \Pr_K[A^{E_K(\cdot)} = \text{type I}]$, and $q = \Pr_P[A^{P(\cdot)} = \text{type I}]$. Here P denotes a uniformly random permutation on n bits, and $\Pr_X[\mathrm{Ev}]$ denotes the probability of event Ev with respect to the random choice of X. Also, A^f is a special kind of notation that represents executing algorithm A with access to a box that compute the function f; at any point, A may supply a value x to the box, and the box responds with the value $f(x)$. Thus, the box is like a special kind of subroutine that A can invoke at any time.

For this question, assume that AES is a secure block cipher, for some reasonable values of T and ε (say, $T = 2^{80}$ and $\varepsilon = 1/2^{48}$, maybe).

Each part below defines a candidate block cipher E. Which ones are secure? For each part, say either "secure" or "insecure"; you should have a justification in mind, but don't bother writing it down.

(Suggestion: If you conjecture it is probably secure, you might want to look for a *reduction*: a proof that it is secure if AES is. If you conjecture that it is probably insecure, you might want to work out how to attack it. This will help you be confident in your answer. But again, you don't need to write down whatever proof or attack you come up with.)

(a) $E_K(M) = K \oplus M$, where M and K are 128-bit values.

(b) $E_K(M) = \mathrm{AES}_K(M) \oplus \mathbf{1}$, where $\mathbf{1}$ represents the all-ones 128-bit value and $\mathrm{AES}_K(M)$ represents the AES encryption of message block M under key K.

(c) $E_K(M) = \mathrm{AES}_{\mathbf{0}}(M)$, where $\mathbf{0}$ represents the all-zeros AES key and M is a 128-bit value.

(d) $E_K(M) = \mathrm{AES}_{\mathbf{0}}(M \oplus K)$.

(e) $E_K(M) = \mathrm{AES}_{K_1}(\mathrm{AES}_{K_0}(M))$, where $K = (K_0, K_1)$ is a 256-bit value (the concatenation of two 128-bit AES keys K_0 and K_1).

(f) $E_K(M) = (\mathrm{AES}_{K_L}(M_L), \mathrm{AES}_{K_R}(M_R))$ where $M = (M_L, M_R)$ is a 256-bit message (the concatenation of 128-bit values M_L and M_R) and $K = (K_L, K_R)$ is a 256-bit key (the concatenation of 128-bit values K_L and K_R).

(g) $E_K(M) = \mathrm{AES}_K^{-1}(\mathrm{AES}_K(M) \oplus \mathbf{1})$, where $\mathrm{AES}_k^{-1}(c)$ represents the AES decryption of ciphertext block c under key k and $\mathbf{1}$ represents the all-ones 128-bit value.

6. (35 pts.) Security of CBC Encryption

This goal of this problem is to prove that a symmetric encryption scheme using the CBC mode is secure under chosen plaintext attack.

Assume that the symmetric encryption scheme is built out of a secure block cipher E with key length k and block length n and with security guarantee $\mathrm{Adv}A \leq T/2^l$, where l is a security parameter and the adversary is allowed T time.

Security under chosen plaintext attack involves the following game. The adversary has access to a box which contains a hidden key K. The box takes as input a pair of same-length plaintext messages (M, M') and the box outputs either (I) the CBC encryption of M (under key K) or (II) the CBC encryption of M' (under key K). The adversary is allowed to play with the box for the available time and must guess whether it is a box of type I or II. To establish security of the CBC mode under plaintext attack we must show that the adversary has negligible advantage in distinguishing between the two cases. As before, the advantage of the adversary is $\mathrm{Adv}A = |\Pr[A^{\text{box of type I}} = \text{type I}] - \Pr[A^{\text{box of type II}} = \text{type I}]|$.

In this problem we will establish this in the special case where the adversary submits only one input of her choice (M, M') to the box. Let us assume that the messages M, M' each consist of j blocks of n bits each. In parts (a)–(d), we will show that if the block cipher is a truly random permutation P, then the advantage of the adversary is at most $2\binom{j+1}{2}/(2^n - j)$. In part (e), we will show that CBC mode is secure when used with a secure block cipher E (instead of a random permutation).

(a) Show that if CBC encryption of messages M and M', with truly random permutation P, each invoke the cipher P on distinct inputs, then the adversary has no advantage in distinguishing between the encryption of M and the encryption of M'.

HINT: What does the distribution of the ciphertext C look like if the box is of type I? if it is of type II?

(b) Recall when using CBC mode with random permutation P, the encryption algorithm picks a random n bit string as the initial vector IV, and outputs the ciphertext $C_0, \ldots, C_j$, where $C_i = P(D_i)$, where $D_0 = IV$ and $D_i = C_{i-1} \oplus B_i$ for $i \geq 1$, and where B_i is the ith block of the message to be encrypted.

Suppose that $D_0, D_1, \ldots, D_{m-1}$ are given and are all distinct, and that $i < m$ is given. Prove that, in this case, the probability that $D_m = D_i$ is at most $1/(2^n - m)$.

HINT: What is the probability that $P(D_{m-1}) = D_i \oplus B_m$? Why?

(c) Show that the probability that the cipher P is invoked twice on the same input while encoding M (or, for that matter, M') is at most $\binom{j+1}{2}/(2^n - j)$.

HINT: Use the fact that $\Pr[A_1 \cup A_2 \cup \cdots \cup A_n] \leq \Pr[A_1] + \Pr[A_2 \mid \text{not}(A_1)] + \cdots + \Pr[A_n \mid \text{not}(A_1 \cup \cdots \cup A_{n-1})]$. (Do you see why this is true?)

(d) Show that, when using a truly random block cipher P, the advantage of the adversary at breaking CBC is at most $2\binom{j+1}{2}/(2^n - j)$.

HINT: There is some event E so that $\Pr[A^{\text{type I box}} = \text{type I} \mid \mathsf{E}] = \Pr[A^{\text{type II box}} = \text{type I} \mid \mathsf{E}]$. What is the event E? What is $\Pr[\mathsf{E}]$? What does this imply about $\mathrm{Adv}A$?

(e) Now show that if we replace the truly random cipher P in the CBC protocol with a block cipher E with security $T/2^l$, then the resulting CBC protocol has security $T/2^l + 2\binom{j+1}{2}/(2^n - j)$.

HINT: Show how to modify any adversary A_{CBC} that can break the CBC protocol with advantage ε, to obtain an adversary A_{cipher} that can break the block cipher E with advantage at least $\varepsilon - 2\binom{j+1}{2}/(2^n - j)$. It is possible to define A_{cipher} by making some simulating the behavior of A_{CBC}—do you see how?