Massachusetts Institute of Technology
6.857: Network and Computer Security
Professor Ronald L. Rivest

Handout 11
April 6, 2009
Due: April 13, 2009

Problem Set 4

This problem set is due *via email,* to `6857-hw@mit.edu` on *Monday, April 13* by the beginning of class.

You are to work on this problem set in groups of three or four people. For this problem set, you are free to choose your own groups. If you do not have a group, please email `6.857-tas@mit.edu`, and we will help you find a group. Be sure that all group members can explain the solutions. See Handout 1 (*Course Information*) for our policy on collaboration.

Homework must be submitted electronically! Each problem answer must appear on a separate page. Mark the top of each page with your group member names, the course number (6.857), the problem set number and question, and the date. We have provided templates for LaTeX and Microsoft Word on the course website (see the *Resources* page).

Grading and Late Policy: Each problem is worth 3 points. Late homework will not be accepted without prior approval.

With the authors' permission, we will distribute our favorite solution to each problem as the "official" solution—this is your chance to become famous! If you do not wish for your homework to be used as an official solution, or if you wish that it only be used anonymously, please note this on your homework.

Problem 4-1. More Kleptography

On the quiz we saw a malicious implementation of RSA that allowed an adversary to hide the factorization of the public key modulus n in the public key (n, e). In this problem you will implement a similar malicious RSA key-generation routine.

In this example the adversary has an RSA public key (N, E) which is used to encrypt the value of one of the prime factors. Pseudocode for the keygen routine follows:

1. Choose two random primes p, q. Compute $n = pq$.
2. Compute $e = p^E \bmod N$ and check that e is relatively prime to $\phi(n)$. If not, start over from step 1.
3. Compute $d = e^{-1} \pmod{\phi(n)}$.
4. Return the public key (n, e) and the private key d.

(a) We need a way to find large primes. Implement a suitable prime-finding routine (e.g. Miller-Rabin). Submit your code.

(b) Implement the kleptographic RSA key-generation routine above. Submit your code, a 1024-bit malicious public key and the adversary's RSA key so we can verify your results. Please submit your results in a text file using the following format:

```
N = < adversary's public modulus in base 10 >
E = < adversary's public exponent in base 10 >
D = < adversary's secret exponent in base 10 >
n = < malicious public modulus in base 10 >
e = < malicious public exponent in base 10 >
```

Problem 4-2. Blind Signatures

A blind signature scheme allows an authority to sign a message without learning the contents of the message. In a blind signature scheme there is a message author and a message signer. The author creates the message m, performs some operation to hide its contents (this step is called "blinding"), then passes it to the signer. The signer signs the blinded message m' with his public key and sends the signature back to the author.

The author then uses the signature of the blinded message m' to produce a signature for m (that is verifiable with the signer's public key).

RSA is easy to adapt into a blind signature scheme. We assume that the author knows the public key (n, e) of the signer, and only the signer knows the secret key d.

To sign a message m:

1. The author chooses a random $r \in \mathbb{Z}_n^*$ and computes $m' \leftarrow mr^e \bmod n$.
2. The author sends m' to the signer.
3. The signer computes $s' \leftarrow (m')^d$ and sends s' to the author.
4. (See part (a))

(a) Show how the author, given s' and r, can compute a valid signature s for the message m.

(b) Argue that even a computationally unbounded signer learns nothing about m from seeing the message m'.

One problem with this scheme is the author can change the contents of the signed message after it has been signed – the author simply claims to have used a different r. This could be a problem in some applications; perhaps m is a record of the author's votes in an election. We would like to modify the scheme so that once the message m is signed, the author can only construct a signature for m.

(c) Suggest a modification to the above scheme that accomplishes this goal. Does your modification change the security guarantee from part (b)?