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

Handout 5
March 30, 2015
Due: April 13, 2015

Problem Set 4

This problem set is due on *Monday, April 13* at **11:59 PM**. Please note our late submission penalty policy in the course information handout. Please submit your problem set, in PDF format, **on Stellar**. *Each problem should be in a separate PDF.* Have **one and only one group member** submit the finished problem writeups. Please title each PDF with the Kerberos of your group members as well as the problem set number and problem number (i.e. *kerberos1_kerberos2_kerberos3_pset1_problem1.pdf*).

You are to work on this problem set with your assigned group of three or four people. Please see the course website for a listing of groups for this problem set. If you have not been assigned a group, please email `6.857-tas@mit.edu`. 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: All problems are worth 10 points.

With the authors' permission, we may 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 in your profile on your homework submission.

Our department is collecting statistics on how much time students are spending on psets, etc. For each problem, please give your estimate of the number of person-hours your team spent on that problem.

Problem 4-1. Group generators

Let p be an odd prime.

(a) Show that -1 is a quadratic residue modulo p if and only if $p = 1 \bmod 4$.

(b) It is also true that 2 is a quadratic residue modulo p if and only if $(-1)^{(p^2-1)/8} = 1$. Show that this implies that 2 is a quadratic residue if and only if $p = 1 \bmod 8$ or $p = 7 \bmod 8$.

Using the above facts, argue that:

(c) 2 is a generator of the multiplicative group modulo 5387

(d) (-2) is generator of the multiplicative group modulo 5399

(e) 2 is a generator of the multiplicative group modulo 5693

Problem 4-2. Elliptic Curves

(a) Let $p = 11$. Using the curve $y^2 = x^3 + x + 1$ over $\mathbb{F}_p$. let $P = (1, 6)$ and $Q = (6, 5)$. What is $P + Q$? For this part, do not use code, perform the group action by hand as described in lecture.

(b) Let $p = 1000000007$. Using the curve $y^2 = x^3+1234567x+555555555$ over $\mathbb{F}_p$. let $P = (185860167, 970276797)$ and $Q = (865214106, 357690920)$. What is $P + Q$? What is $1000000P$? (Hint: It is OK to use Sage here and later in this problem.)

(c) Using the same curve, and the point $G = (176935010, 637089476)$, Alice and Bob would like to share a key using ECDHE. Alice picks $d_A = 123456789$ and Bob picks $d_B = 987654321$. What is their shared secret? Show your work and be sure to verify that Alice computes the same secret as Bob.

(d) Using the same curve, and the point $G = (176935010, 637089476)$, Alice and Bob would like to share a key using ECDHE. Alice picks $d_A = 70297$ and Bob picks $d_B = 14226$. What is their shared secret? Explain what happened.

Problem 4-3. Unlinkable Serial Transactions

In this problem, we will explore *unlinkable serial transactions*, as described by Stubblebine, Syverson, and Goldschlag; the paper is available on the course website. The general model is that:

- There is a server providing some service to multiple clients.
- Clients must "subscribe" initially (in the real world, this may involve payment, age verification, or some other check). After subscribing, clients may use the service as often as they wish.
- The server wishes to verify that every time it receives a request, the client has previously subscribed.
- To protect the privacy of the clients, *the server should not be able to trivially correlate different requests as being from the same client.* Thus, simply making each client provide a username and password does not work, since the server could simply look at all requests coming from the same username. [1]

Many existing services require the first three points, and solve the problem by having users create accounts with usernames and passwords. However, this does not meet the final criterion, since the server could easily look for all transactions associated with one account. (Ideas along the lines of creating multiple "one-time accounts" encounter difficulties with ensuring that the owner of each account is properly subscribed, without tying them together.)

The proposal uses *blind signatures*, a technique for creating digital signatures where the signer does not know the message. The idea works as follows:

- The user signs up for a subscription at which time she submits payment and a message of the form

  $$\text{Blind}(\text{Hash}(N_1))$$

  (e.g. $(r^e \cdot \text{Hash}(N_1)) \pmod n$ where server's $PK = (n, e)$), where N_1 is her initial nonce.
- The server accepts payment and gives the user

  $$\text{Blind}(\text{Sign}(\text{Hash}(N_1))$$

  (e.g. $r \cdot (\text{Hash}(N_1))^d \pmod n$ where server's $SK = (n, d)$), from which the user can obtain

  $$\text{Sign}(\text{Hash}(N_1))$$

  (e.g. by dividing by r to obtain $(\text{Hash}(N_1))^d \pmod n$). This is the server's hash on her first nonce (which the server hasn't seen).
- Later on, the user can request service by sending a message of the form:

  $$\text{“}\mathsf{request}\text{”}, \text{Sign}(\text{Hash}(N_i)), N_i, \text{Blind}(\text{Hash}(N_{i+1}))$$

- Upon receiving such a request, the server checks that the nonce N_i is previously unused, but that the signature is valid. If so, it provides service, and also returns

  $$\text{Blind}(\text{Sign}(\text{Hash}(N_{i+1}))$$

  so the user now has a token for her next request.

For this problem, we will use RSA signatures for the signature algorithm, and SHA-256 for the hash function. The server has a public key (n, e) and a secret key d. It signs messages m by generating $\sigma_m = m^d \pmod n$.

We have set up a server that follows the above protocol. The service it provides is simply returning different plaintext ASCII strings. We will skip the "subscription" step, so all the requests will be service requests; thus, there's no need to include the string "request" as part of the request.

[1] Here we won't consider traffic/timing analysis, or looking at the IP address of the incoming request, or the like; we'll just consider the interaction at the cryptographic/protocol level.

On the course website you will find a file `USTclient.py` that includes skeleton code for communicating with the server. (It uses HTTP, so you can access it in other ways if you wish, but using this code is almost certainly easiest.) This code includes the server host:port and its public verification key.

You will also need an initial (nonce, signature) pair $(N_i, \mathrm{Sign}(\mathrm{Hash}(N_i)))$ to get started (the one that normally would be provided with subscription). Each student can get ONE such nonce/signature pair by going to `https://6857.scripts.mit.edu:444/2015ps4/index.py` with their MIT certificate. Remember that the server will only accept each nonce *once*, so be careful with your code! If you lose all of your nonces, you can email the TAs for one additional nonce.

Find eight of the strings returned by the server. (Once you find some of the strings, do not share them with other groups!) Submit the strings you found, and any code you used.

Remember that in order to be successfully anonymous, both the nonces and the blinding must be *random*!