Problem Set 6

This problem set is due on *Thursday, October 25, 2001* at the beginning of class. Late homeworks will *not* be accepted. You are to work on this problem set in groups of three or four people. You may offer your non-secret *signature* shares and proofs of correctness to anyone. You may not disclose your secret share.

Mark the top of each sheet with your names, 6.857, the problem set number and question, and the date. **Type up your solutions** and be clear. Each problem should begin on a new sheet of paper. That is, you will turn in each problem on a separate pile of paper. **Cite** your sources of information.

Problem 6-1. Practical Threshold Signatures

This problem set asks you to implement part of the threshold RSA signature scheme (protocol I) from handout 21. This homework involves partial collaboration with other groups. Get started early! Your success depends on the success of the groups you choose to partner with. Remember to cite those whom you collaborate with.

(a) In the beginning

Before we play with threshold RSA signatures, let's make sure we can implement a simple RSA signature verification routine. Write a function that verifies an RSA signature given the public key (e, n), a hashed message x, and a signature s. You do not need to submit this code, but in your printed solutions you must submit a decimal x that has the following signature for the given public key:

$s = 3077193185180312374188656237229861515569633043597523766171400284064154219729 6$
$n = 85212746447079824936395777044274071120738223794208795362205208542665542508313$
$e = 67$

where $n = pq$ for two secret primes p, q and $x^d \bmod n = s$ where $ed = 1 \bmod (p-1)(q-1)$. We will provide a copy of these numbers to `6.857-students-public@mit.edu` as well.

(b) Lots of code

Now implement the code that a share holder would need for Shoup's threshold RSA signature scheme (protocol I). You may use code from previous problem sets. You may also use any built-in functions (modPow, modInverse, etc). We have provided a framework in Java on the 6.857 Web page. You are welcome to use this as a starting point.

The following functions are necessary for a share holder:

Function: genRsaSignatureShare
Given: A 128-bit hash x of a message, a secret key share s_i, a modulus n,
and the number of groups l
Return: A valid signature share x_i

Function: verifySignatureShare
Given: A share x_i, modulus n, global quadratic residue v, challenge c, sharer's quadratic residue v_i, response z, $\tilde{\mathrm{x}}$, v', and x'
Return: True iff the signature share is valid

Function: combineRsaSignatureShares
Given: Array of signature shares, array of the group numbers of the signature shares, the number of signature shares you have, the hashed message x, the public key (e, n), and the number of groups l
Return: The RSA signature of x.

Function: verifyRsa
Given: message x, signature y, and public key (e, n)
Return: True iff the message is properly signed with RSA

Function: lambdaProduct
Given: An array of signature shares, their group numbers (whose share it is), the number of signature shares, the number of groups, and the modulus
Return: The value w as defined by Shoup

Assume that the hash function $H'(m_1, m_2, m_3, \ldots, m_k) = 2^{m_1} 3^{m_2} 5^{m_3} \ldots p_k^{m_k} \pmod{n} \pmod{2^{128}}$ where p_k is the kth prime number and n is the public modulus.

Submit a printout of your code.

(c) Obtain your share

Each group will receieve one secret share. Kevin (the 6.857 dealer) will create secret shares with a $(3, 21)$-threshold for Shoup's shared RSA signature scheme. We will send by email to each group the following parameters, all according to the specifications of Shoup's protocol I:

Global, public information:

- A modulus n
- A special (already hashed) message, x, in decimal form
- The number of groups, $l = 21$.
- The threshold required to sign a message, $k = 3$.
- A key index i (that is, which group number you are)
- A global quadratic residue $\pmod{n}$, v
- Your group's personal quadratic residue $\pmod{n}$, v_i

We will also give you a secret share which you CANNOT disclose to other team members: a key share s_i.

In your printed solution, show the output of generating a signature share, x_i, and a proof of correctness (z, c), as specified in handout 21.

(d) *Nullius boni sine socio iucunda possessio est*[1]

Knowing that Kevin dealt one share to each group in a $(3, 21)$-threshold scheme, obtain the appropriate number of signature shares from other groups to produce a valid signature on the decimal which you will receive by email. This decimal represents a hash of a message.

You may give your signature share x_i (not your secret share!), your proof of correctness (z, c), your personal quadratic residue v_i, and your group number i to anyone in the class. If you do not trust in the validity of a signature share from another group, verify that its proof of correctness is good.

Submit by email the details of generating a signature from signature shares. Here is an example filled with fake values of how your email to `6.857-staff@mit.edu` should read. Use the notation from handout 21:

```
To: 6.857-staff@mit.edu
Subject: Ssssh!
-------
Signature share x_3: 5294380932581242146509l324
From group 3: Alice, Bob, and Charlie
2\lambda_{0,3}^S = 945121409214
v_3 = 2149218521095215
v' = 1095235123515
c = 32152352185215359135
z = 320593210252135

Signature share x_5: 1240326445329985763943209
From group 5: Donna, Eve, and Fred
2\lambda_{0,5}^S = 3251294821412
v_5 = 85353221095215
v' = 351546263265
c = 3292185215359135
z = 2502109582150925
...

\~x = 1039535215215
w = 124092185092142l4214
e'=  32210948214021424
a = 2494212147787
b = 87213649832112
Combined signature y: 32501325512552365325
```

We must receive your email by the problem set deadline.

Hint: If you want to make sure your signature combination code works, write your own dealing code and verify that the signature using d is the same as the signature from combining signature shares. This will involve the generation of safe primes and quadratic residues mod a composite.

[1] Seneca. *Epistulae Morales* Liber I, §VI (4)

Remember that a number g is a QR mod pq where p, q are prime iff g is a QR mod p and g is a QR mod q.

Problem 6-2. The many ways to share

Compare and contrast Shamir's secret sharing scheme with Shoup's threshold signature scheme. In what practical situations may Shamir's scheme be most appropriate? In what practical situations may Shoup's scheme be most appropriate? Limit your discussion to one page and at most two main points.

Problem 6-3. Acknowledgements

List the names of the students from other groups with whom you collaborated. For each person in your group, explain in a few words what each person did (coding, writing up, designing, nothing, etc.) for this problem set.

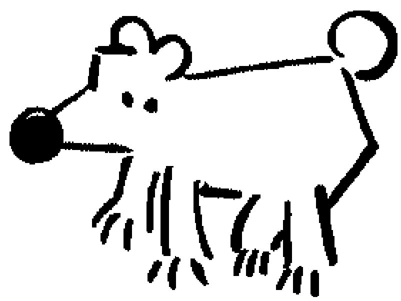

Figure 1: This bear allegedly produces prime numbers. ©Arktinen Krokotiili Projekti