Problem Set 6 Solutions

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

Given the following values:

$s = 30771931851803123741886562372298615155696330435975237661714002840641542197296$
$n = 85212746447079824936395777044274071120738223794208795362205208542665542508313$
$e = 67$

where $n = pq$ for two secret primes p, q and $x^d \bmod n = s$ where $ed = 1 \bmod (p-1)(q-1)$, a solution to x is equivalent to 47236967882377955842774881613496506684928002649044 mod n.

(b) Lots of code

We have posted on the 6.857 Web page our code for dealing secret shares and for combining signature shares. We have also posted code from Patrick Cody, Ed Cotler, Joseph Hastings, and Pavel Langer. They implemented both the code to combine signature shares and the code to deal secret shares.

The trickiest and most perilous part involved where to reduce a number by the modulus. Because you do not know the factors of n, you do not know m. If you reduced an exponent by n instead of leaving it in its unreduced form, your code would not work properly. Also, if you used the Kaffe 1.0.5 Java compiler, the compiler would produce incorrect results for negative exponents in BigInteger.modPow. All the submitted solutions were written in Java.

(c) Obtain your share

Each group received one secret share. In this $(3, 21)$-threshold for Shoup's shared RSA signature scheme, each group needed signature shares from two other groups. We created the following global values 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

It took on average 200 iterations of Java's BigInteger prime number generator for the TA's machine to generate a 512-bit safe prime. Here are the parameters we used:

$p = 2 * p1 + 1 =$
227831032330598799519387521067306149501158245904765157855933942032760183351633
38316734849912471563878932639430986252641659246356056131375148087798804411983

p1 =
113915516165299399759693760533653074750579122952382578927966971016380091675816
69158367424956235781939466319715493126320829623178028065687574043899402205991

q = 2*q1 +1 =
155767675613212975391327614596113067126436269452096723906471714441997543443330
00516592379011440314849908509372400819653142656103544453478591720460054254183

$q1 =$
778838378066064876956638072980565335632181347260483619532358572209987717216650
0258296189505720157424954254686200409826571328051772226739295860230027127091

$n = p * q =$
354887103386961515808503593782831322689175884250283740820474607418089097856391
732009947276145394697842449630057681710242263832046684525002892603425586158946
626633968932284768151659462840248268968605949354140661451495035929748503597397
933099756217341471075443735683962358557376320185217486990146673928933074889

$m = p1 * q1 =$
887217758467403789521258984457078306722939710625709352051186518545222744640979
330024868190363486744606124075144204275605659580116711312507231508563965397270
666907936377768192700364741245816515522886034169881213027323471134939560252647
74943131823357398086650646720143821565643604431404225534101716417518602181

$e = 37$
secret $d =$
815281183456533211992508255987585471042701355710111296479468692717231711291710
735698527526279960792340762663646025550556552046593734719601239764626346581275
747969455049841041940875708171831392642652031399350303862945892394268785097027
66163958972814906349895188877969998195456285153182261301606982653936012815

These were the three coefficients for the polynomial:
$a[0] =$
815281183456533211992508255987585471042701355710111296479468692717231711291710
735698527526279960792340762663646025550556552046593734719601239764626346581275
747969455049841041940875708171831392642652031399350303862945892394268785097027
66163958972814906349895188877969998195456285153182261301606982653936012815

$a[1] =$
341459893691398401174614571473269605606020732050863175826103390386248820004988
435433820350993835194693797951805440698839883702878758335766883487295985440816
995651058055891764526220142201391864815716794858962980767613696060445247818713
88223405571870964902162411408963452506618882933207375824568521944393441358

$a[2] =$
517762635241974822188300006649741994547454260497664919345208803622604054455734

5258959245809242206685071506745512288957776689490541327335660248618372310328522010427586722459498596456302828985186540842150677810364431771907251276727510003799173844891760761618555327709883147658876843255224120375850272622881818 6

The message to be signed was "quiessettantusfructus" from Cicero's *De Amicitia* VI (22) in Latin. The full sentence would roughly translate to "Is not prosperity robbed of half its value if you have no one to share your joy?" This text seemed highly relevant to this problem set. We didn't actually hash the message; we just took it base 36.

This message base 36 is

$x = 35888433672751881497862094167965 2$

and the expected signature after combining signature shares is
$y =$
13068467459287386325552213242746665418241177111920537660522003241037451070987714591519311424925687924130842726518370892282813674676208534531454884959080464535063253981125524748061103784432323818953131363645581851900711588896069258291982176756169728730362716261594760690972744992544693404290466553386610524183 0

If you did not reduce the signature mod$n$, then your solution was likely

$y =$
1203783266572671260173101071022458900615429451953900125663391147620292461435114364153126093402041818776088397293967354605800131905318797672424286738357526138429369566383266846898300833932556673929207660382759740729446564066562395642383195953638341476172979508095617459423817205561282570010466290181713595371388224137428741361495902960131514709043841606506714094836268188062221844461652557035226189004095005329312843037783862538989491212101627947135596003087371026236834352822937539210229347909453238307214673372572269139663219565680159980592649713428896367327708001164611966008915918184217460504626662000700652498754 4

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

Everyone who submitted problem 6-1 was able to successfully generate signature shares. For your enjoyment, Figure 1 shows the popularity of each signature share. The first column is a group number. The second column is the number of times someone (including that group) used the signature share. The third column is the timestamp of the email of the combined signature from the group.

In the email we asked you to send, it turns out that you do not have to provide the v' value from each group. A few students noticed this error on our part.

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

[1] Seneca. *Epistulae Morales* Liber I, §VI (4) roughly means "nothing is good to have without friends to share it."

Group	# of References	Timestamp
17	7	Thu, 25 Oct 2001 02:51:09
9	6	Tue, 23 Oct 2001 23:42:52
15	5	Wed, 24 Oct 2001 16:52:11
2	4	Wed, 24 Oct 2001 22:06:13
5	4	Thu, 25 Oct 2001 08:05:06
20	4	Wed, 24 Oct 2001 20:30:45
7	4	Thu, 25 Oct 2001 01:48:25
1	3	Thu, 25 Oct 2001 18:15:59
11	3	Thu, 25 Oct 2001 13:37:34
13	3	Thu, 25 Oct 2001 02:49:53
3	2	Thu, 25 Oct 2001 00:03:41
14	2	Thu, 25 Oct 2001 01:21:35
8	2	Thu, 25 Oct 2001 14:00:25
19	2	Thu, 25 Oct 2001 13:59:32
12	2	Thu, 25 Oct 2001 07:53:56
10	2	Thu, 25 Oct 2001 16:42:38
4	2	Wed, 24 Oct 2001 18:02:59
6	1	Thu, 25 Oct 2001 23:53:15
18	1	Thu, 25 Oct 2001 17:53:37
21	1	Thu, 25 Oct 2001 21:34:13
16	0	N/A

Figure 1: A signature share popularity contest.

may Shoup's scheme be most appropriate? Limit your discussion to one page and at most two main points.

Jordan Gilliland, Steve Lustbader, Yao Li, and Ryan Wagner submitted the following essay:

Both Shoup's and Shamir's schemes are very similar, since they are both based on splitting information into pieces so that no one person contains all of the necessary information. They differ in their goals, however, as Shoup's scheme is used to generate a valid signature, while Shamir's is used to protect any piece of data, which may, for example, be the secret key of a signature scheme. There are two major properties for these schemes: reusability and verification. Shoup's scheme has both of those, in that each share can be verified to be valid and secret shares can be reused many times, while Shamir's has neither.

Shoup's scheme is geared towards digitally signing a message wherein no single person can be trusted to construct the signature by him or herself. Each participant is given a secret share, from which one can derive a signature share for a given message. When any one of them wants to sign a message, he or she collects enough signature shares for that message from other participants to reach the threshold k. Each share can be verified, and when enough valid shares have been collected, they can be combined to produce a valid signature for that message only. It is impossible to construct a valid signature without enough shares, and nor can any other message be signed with those same signature shares. It is also impossible for a participant to pass off an invalid signature share as real because each share can be verified. No secret is ever revealed during the process. This allows for everyones secret shares to be reused for different messages. This scheme is useful when each person is individually distrusted to sign a message but k participants together can be trusted. An excellent example is a company signing electronic payments. No single employee should be allowed to sign the e-check, but several of them together should be able to do it. At the same time, signing one e-check does not allow any future e-checks to be signed without again collecting group consent.

Shamir's scheme is a little more general in that it is used to protect any secret and not just for signing messages. Like Shoup's scheme, each participant is given a secret share (by picking points off a $k-1$ degree polynomial, where k is again the threshold), and having fewer than k shares reveals no information about the secret. In this system, however, the secret shares are directly combined to reveal the secret. This makes the system un-reusable, unless extra steps are taken (such as putting the secret shares on smart cards and having a smart card reader combine the shares and act on the secret, in which case only the smart card reader would know the secret). In addition, the secret shares cannot be verified. A single user could give bogus data, which would result in the secret that is revealed not actually being the secret. The other participants would not know that the secret data is incorrect until they actually tried to use it (e.g., if the secret is a combination to a safe, they would not know the revealed combination was wrong until they tried to unlock the safe). This means the process would have to be repeated, although no one knows which of the shares was false. This scheme requires two assumptions to be used in practice. One is that all or most of the participants involved can be trusted to not give false information, and two, that the secret will either not be used again or that it is protected by some other mechanism (such as in the smart card reader example above). One use of Shamir's system could be in protecting an electronic will. Each beneficiary could receive a share of it and when enough shares are combined, the will would be revealed. Since the will does not need to be hidden again, it does not need to be continually hidden after the first time it is revealed. Another use might be in electronic voting, wherein a voter somehow submits shares of his or her vote, rather than the complete vote. After the shares have been combined, the vote is revealed and no longer needs to be kept secret.

Shalini Agarwal, Steve Bull, Christine Karlovich, and Casey Muller offered this list of practical applications of secret sharing:

Practical situations for Shamir:

1. Pieces of the decryption key for a will are distributed among several attorneys and/or heirs.
2. A scavenger hunt where a certain number of clues must be found to decrypt the final clue.
3. Nuclear activation codes are distributed to some number of authorized personnel.
4. A simple secret ballot system where voter anonymity is preserved (more sophisticated systems are obviously available) and a majority can easily be detected.

Practical situations for Shoup:

1. An online publication that requires a certain percentage of editors to approve each submission.
2. A constitution document where amendments require ratification by a certain number of board members.
3. A marriage witnessing system that requires multiple priests to be present. Keys would come from the Pope.

Problem 6-3. Acknowledgements

Thanks for listing your acknowledgements. An important part of academic research involves giving proper credit to those who deserve it.