Homework 2

CS161 Computer Security, Spring 2008
Assigned 2/13/08*
Due 2/25/08

1. **Signatures and Attacks**

   Recall that to use the ElGamal signature scheme, Alice randomly selects her private signing key $x \in \mathbb{Z}_p$ and computes her public verification key as $y = g^x \mod p$, where p is a large prime and g is a publicly known generator of $\mathbb{Z}_p^*$. To sign a message $m \in \mathbb{Z}_p^*$, Alice first picks a random integer $1 \leq k < p-1$ such that $\gcd(k, p-1) = 1$. Next she computes $r = g^k \mod p$ and $s = k^{-1}(m - xr) \mod p-1$. The signature on m is $\sigma = (r, s)$. If Bob wishes to verify whether a purported signature (r, s) corresponds to a message m and Alice's public key y, he checks that $y^r r^s \equiv g^m \mod p$, which will be true if it was computed as described.

   (a) (3 points) Mallory has discovered a bug in a system operated by Alice. Under certain circumstances, when signing a message within some protocol, Alice's system will not pick k randomly as intended, but instead use some specific value. Mallory doesn't know which value this is, but she has managed to intercept two different messages which have been signed using that same value for k.

   Assume $\sigma_1 = (r, s_1)$ and $\sigma_2 = (r, s_2)$ are valid signatures on messages m_1 and m_2 respectively, and that they were generated using Alice's private key and the same value for k (thus also causing the "r" values to be equal). Show how Mallory can compute k given m_1, m_2, σ_1, and σ_2.

*Consider this this assignment a Valentine's Day present ... because we love you and want you to learn more cryptography.

(b) (3 points) Once Mallory has done that, she can do something much worse. Show how Mallory can use m_1, σ_1, and k to achieve a *total break* of the signature scheme, that is, compute Alice's private key x.

(c) (6 points) As Mallory was busy using Alice's private key to rob, defame, and mock her, Bob was implementing his own system which employs ElGamal signatures. Fortunately, his implementation did not have the same bug. However, Mallory next noticed a more subtle problem with the ElGamal signature scheme as described above, even when it is implemented correctly. She found a flaw in the scheme allowing an *existential forgery*, that is, the production of a signature on some message (but not necessarily any message you want).

Show how, using only Bob's public key y', Mallory can compute a valid signature on some message m. The message need not be anything meaningful; in particular, it is fine if it is a random element in $\mathbb{Z}_p^*$.

(d) (5 points) What is a simple way to modify the (naive) version of ElGamal signatures given in this problem to prevent this existential forgery attack?

2. **Secret Sharing**

In one of lectures covered by this homework, we discuss schemes for sharing a secret among a group of people so that various subsets of the group will be able to reconstruct the secret by combining their shares. It turns out you don't need computers or sophisticated mathematics to accomplish this – you can implement even some of the more complex secret sharing scenarios using nothing but sheets that can be overlaid on one another.

If you are reading a hardcopy of this homework assignment that was handed out in class, you have probably already noticed that the last page is a transparency with a pattern on it. You have received one of five different such transparencies ("Share A" - "Share E"); each one is a share of a secret image.

To reconstruct the secret image, briefly get together with some of your classmates to collect one copy of each of the five shares. Try overlaying various combinations of two or more shares to see if you can reveal the image.

Some combinations of shares (in particular, any share by itself) will yield zero information about the secret. The other combinations all completely reveal the image (although some make it easier to see than others).

Here are a couple practical tips. If the sheets are misaligned by even a few pixels, the image will not reconstruct properly. Since the pixels are so small, you need to align the sheets very precisely for this to work. Also, it will be easier to see the image if you lay the sheets on a white background for better contrast.

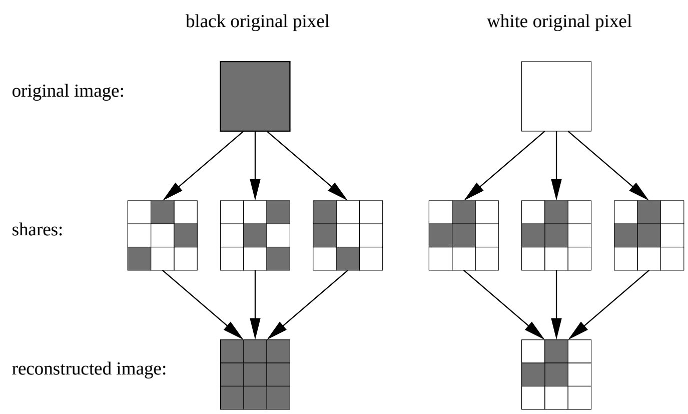
Collaboration is acceptable on parts (a) and (b) of this problem (and necessary to obtain the secret!), but not on part (c). Also, we have posted a pdf of the transparencies and png's of the raw patterns at `http://inst.eecs.berkeley.edu/~cs161/sp08/hw02images.tar.gz`. If you prefer, feel free to download these and print out your own sheets or assemble the images in an image editor. If you are curious about how all this is accomplished, `http://tinyurl.com/3c8o6m` is a good starting point.

(a) (1 point) Find a combination of shares that reconstructs the image and list the letters of those shares here. What does the image look like?

(b) (3 points) Experiment with other combinations of shares to determine which sets of shares reconstruct the image and which reveal no information. State the access policy these shares implement. For example, your answer might be 2 out of 5, 5 out of 5, or some more complicated access policy like $(A \wedge B) \vee C \vee (B \wedge D \wedge E)$.

(c) (5 points) Having seen a demonstration of these techniques, Alice decides to try to devise her own scheme, specifically, a 3 out of 3 scheme. Her idea works as follows.

Each pixel of the source image will be represented by a 3 by 3 grid of subpixels in the shares. To generate the shares for a white pixel in the original image, Alice randomly selects three subpixels, and darkens those same subpixels in the each of the shares. To generate the shares for a black pixel in the original image, Alice randomly selects three subpixels to darken in the first share. Next, she randomly selects three subpixels other than those to darken in the second share. Finally, the remaining three subpixels which were darkened in neither of the first two shares are darkened in the third share.



An example of this process is shown above. As you can see, when all three shares are combined, a region corresponding to a black pixel in the original image will be completely dark, while a region corresponding to a white pixel will be two thirds white.

Does Alice's scheme offer perfect (i.e., information theoretic security)? If your answer is yes, show that any set of shares other than all three reveals nothing about the original image. If your answer is no, explain why this is not the case.

3. **Zero-Knowledge**

> "There are known knowns. These are things we know that we know. There are known unknowns. That is to say, there are things that we know we don't know. But there are also unknown unknowns. There are things we don't know we don't know," a U.S. Secretary of Defense is quoted as saying. One might add, "And there is zero-knowledge. These are things we know that somebody else knows, and we provably cannot know what they are are."
>
> — *Complexity Theory and Cryptology, Jörg Rothe*

Suppose we have an undirected graph $G = (N, E)$, where N is a set of nodes and we represent the edges as a subset E of $N \times N$. Since G is undirected, E is a symmetric relation on N.

Now for any edge $(n_1, n_2) \in E$, we will say that the edge is *flagged* by the node n_1 and the node n_2. Also, we say that a subset of the nodes $N' \subseteq N$ flags an edge $e \in E$ if some $n \in N'$ flags e.

It happens that Merlin and Arthur are interested in flagging all edges in the graph with small subsets of the nodes. Merlin claims to Arthur, who also has G, that all its edges can be flagged with a set of k nodes (where $k < |N|$, otherwise it is obviously true).

Merlin is convinced of this fact because he actually has such a set N'. If Merlin wants to convince Arthur that such a set (i.e., of size k and flagging all edges) exists without revealing the members of N', how can he do so?

By devising a zero-knowledge proof system, of course. Merlin has seen some zero-knowledge proof systems for other graph properties and decides to design one in the same vein. It will have the following basic structure:

Phase 1 Merlin commits to some information about G and his set N'.

Phase 2 Arthur sends him a random challenge, which could be as simple as just telling him to do one of two things.

Phase 3 Merlin responds, and Arthur checks some property of the response and that it matches the previous commitments, then either accepts or rejects.

The protocol must have the following properties:

Completeness If Merlin is being honest (i.e., he does have such a set N') and both he and Arthur follow the protocol, Arthur will always accept.

Soundness If the claim Merlin is making is false (i.e., there is no set of size k that flags all edges), then no matter what Merlin does, if Arthur follows the protocol he will reject with probability at least p_s, where p_s is a constant greater than zero.

Zero-knowledge If Merlin follows the protocol, then no matter what Arthur does, Arthur will not learn anything about Merlin's solution (the members of set N').

Efficiency All computations required of Arthur are polynomial time.

Merlin begins Phase 1 of the protocol as follows:

Proceeding as in many other zero-knowledge proof systems relating to graph properties, Merlin first picks a random renaming of the nodes, that is, a permutation $\pi : N \to N$. Merlin computes a commitment to the permutation,[1] and sends this commitment to Arthur. Note that he does not reveal the permutation itself.

Merlin then constructs the adjacency matrix[2] corresponding to E

	n_1	n_2	$\dots$	$n_{\lvert N\rvert}$
n_1	0	1	$\dots$	0
n_2	1	0	$\dots$	1
$\vdots$	$\vdots$	$\vdots$		$\vdots$
$n_{\lvert N\rvert}$	0	1	$\dots$	0

and applies the permutation to rearrange the adjacency matrix (that is, regenerates the adjacency matrix, but with the columns and rows in the permuted order):

	$\pi(n_1)$	$\pi(n_2)$	$\dots$	$\pi(n_{\lvert N\rvert})$
$\pi(n_1)$	0	0	$\dots$	1
$\pi(n_2)$	0	0	$\dots$	0
$\vdots$	$\vdots$	$\vdots$		$\vdots$
$\pi(n_{\lvert N\rvert})$	1	0	$\dots$	0

He then computes commitments to each individual entry in this adjacency matrix and sends all the commitments to Arthur. Again, he does not send the values within the adjacency matrix, only the commitments.

At this point, Merlin has committed to a random relabeling of G in a flexible way that allows him to later reveal any individual parts of the relabeled graph that he wants.

[1]The following is one way he might do this. If s_p is a string defining the permutation (for example, by listing the nodes in the permuted order), then Merlin picks a random string of bits s_r and computes the commitment as $c = h(s_p||s_r)$, where h is a preimage resistant hash function. To open this commitment if he later needed to, Merlin would reveal both the permutation and s_r.

[2]The actual 1's and 0's in these two tables are of course only given as examples.

(a) (13 points)

Design the rest of the protocol. Your solution should include the following parts.

- The rest of Phase 1, which should consist of one or more additional commitments that Merlin computes and sends to Arthur.
- Phase 2, which should consist of Arthur posing a challenge to Merlin.
- Phase 3, which should specify how Merlin responds to the challenge and how Arthur checks the response and decides whether to accept or reject.

Possibly helpful hint: Note that a set N' flags all the edges in G if and only if for all $n_1, n_2 \notin N'$, $(n_1, n_2) \notin E$.

(b) (8 points)

Prove that your protocol is sound. That is, assume no $N' \subset N$ of size k flags all the edges in G. Then show that no matter what Merlin does, if Arthur follows the protocol, he will accept with probability at most p_s, where p_s is some positive constant.

(c) (4 points)

Prove that your protocol is zero-knowledge. This need not be formal, but it should be a convincing explanation of why the messages Arthur receives from Merlin give him no information about the members of N', provided Merlin follows the protocol (whether or not Arthur does). Note that revealing the size of N' is acceptable (and required, since that is what is being proven).

(d) (6 points)

Suppose we replace the graph problem we have discussed so far with the problem of proving that a graph is two-colorable, that is, proving that with two available colors, we may assign a color to each node so that no nodes connected by an edge have the same color.

As in the previous situation, Merlin has a witness to this fact (in this case a valid two-coloring) and wants to prove its existence to Arthur without revealing anything more about it.

Is there a protocol for this problem that satisfies the requirements given for completeness, soundness, zero-knowledge, and efficiency?

If your answer is yes, give one; if your answer is no, explain why none exist.

Hint: Note that it is easy (polynomial time) to determine whether a graph is two-colorable and compute a two-coloring if one exists. Your answer may assume this fact. Using this fact, answering correctly should only take two or three sentences.