Massachusetts Institute of Technology — Handout 6
6.857: Network and Computer Security — April 24, 2017
Professors Ronald L. Rivest and Yael Tauman Kalai — **Due:** May 3, 2017

Problem Set 5

This problem set is due on *Wednesday, May 3rd, 2017* 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 Gradescope. *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_pset4_problem1.pdf*).

You are to work on this problem set with groups of your choosing of size three or four. If you need help finding a group, try posting on Piazza or email `6.857-tas@mit.edu`. You don't have to tell us your group members, just make sure you indicate them on Gradescope. 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 be provided as a separate pdf. 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 5-1. Zero knowledge

Here we ask you to describe a zero-knowledge proof for a graph property, and argue that the protocol is complete, sound, and zero knowledge.

Our proposed graph property is having feedback arc set of size k.
See `https://en.wikipedia.org/wiki/Feedback_arc_set`

For convenience, for any positive integer n, let $[n]$ denote the set $\{1, 2, \ldots, n\}$.

Definition. A directed graph G with n vertices and m edges has a feedback arc set of size k if

1. The n vertices can be given distinct labels from $[n]$.
2. The m edges can be divided into two disjoint sets A and B of sizes $m - k$ and k, respectively.
3. The edges in A go from lower-labelled vertices to higher-labelled vertices, while the edges in B go from higher-labelled vertices to lower-labelled vertices.

It is interesting to note that every directed cycle in G contains at least one edge from the feedback arc set B.

This problem asks you to design and evaluate a ZK protocol for the property of having a feedback arc set of size k. The Prover and the Verifier both know the directed graph G and the integer k, and the Prover also knows a "witness" (i.e., a feedback arc set B of size k).

Hint: For a problem like this, let the prover create and commit to a fresh random data structure for each round. This data structure may have several parts. If the data structure is correctly formed, then the graph has the desired property. Then the verifier can challenge the prover to open a small randomly-chosen portion of the committed data structure, to see that it looks OK. The piece is small enough that, by itself, it doesn't reveal anything about the witness, but if the prover can't answer the challenge, then the proof fails. This basic round is repeated until the desired confidence level is achieved.

More Hint: Number the vertices $\{1, 2, \ldots, n\}$. Number the edges $\{1, 2, \ldots, m\}$. The integers n, m, and k are known to the Prover and to the Verifier.

The set E of directed edges is known to Prover and Verifier. Assume that E is represented by two functions (arrays) t and h, such that the e-th directed edge is the ordered pair $(t(e), h(e))$, for $e \in [m]$. The edge $(t(e), h(e))$ goes from the vertex $t(e)$ at its tail to the vertex $h(e)$ at its head.

The Prover (but not the verifier) knows a set B of k edges that forms a feedback arc set.

In each round, the prover will work with a fresh random renumbering of the graph G. This renumbering may be represented by two functions π (mapping the new vertex numbers to their old (standard) ones), and δ (mapping the new edge numbers to their old (standard) ones).

$$\pi(v') = v \ , \tag{1}$$

$$\delta(e') = e \ . \tag{2}$$

These mappings are one-to-end on $[n]$ and $[m]$, respectively. We can imagine that π is a random permutation of $[n]$, and that δ is a random permutation of $[m]$. These random renumberings will keep the Verifier from learning the Prover's secret feedback arc set B.

The set E' of edges in the renumbered graph may be represented by mappings t' and h' from $[m]$ to $[n]$, giving the tail and head of each edge in the renumbered graph. The mappings t' and h' must be consistent with their unnumbered versions t and h. That is, they must satisfy for all edges e':

$$\pi(t'(e')) = t(\delta(e')) \ , \tag{3}$$

$$\pi(h'(e')) = h(\delta(e')) \ . \tag{4}$$

The Prover also creates an array β mapping $[m]$ to $\{0, 1\}$ such that

$$\beta(e') = 1 \longleftrightarrow e' \in B' \tag{5}$$

where B' is the set of edges e' such that $\delta(e')$ is in B. That is, B' is the feedback arc set in the renumbered graph. The number of ones in the range of β is equal to k.

Finally, the prover also creates an array λ mapping $[n]$ to itself, such that $\lambda(v')$ is the label assigned to the vertex v' in the renumbered graph. The mapping λ is one-to-one. It should satisfy the condition that for any edge e'

$$e' \notin B' \longleftrightarrow \lambda(t'(e')) < \lambda(h'(e')) \ . \tag{6}$$

In each round, the Prover generates a new randomly renumbered graph. The Prover sends the verifier commitments to π, δ, t', h', β, and λ.

(a) Argue that if the revealed *all* components of π, δ, t', h', β, and λ, the Verifier could confirm that the Prover actually knows a feedback arc set B of size k for the graph G.

(b) Argue that it suffices for the Verifier to know the following, in order for the Verifier to be convinced that the Prover does indeed know a feedback arc set B of size k:

- π is one-to-one on $[n]$,
- δ is one-to-one on $[m]$,
- t' satisfies equation (3) for all $e' \in [m]$.
- h' satisfies equation (4) for all $e' \in [m]$.
- β has a range with k ones and $m - k$ zeros.
- β satisfies equation (5)
- λ is one-to-one on $[n]$

- λ satisfies equation (6) for all $e' \in [m]$.

(c) Devise a ZK protocol for the property of having a feedback arc set B of size k in a graph G. Assume that in the first step of a round the Prover provides commitments to π, δ, t', h', β, and λ. In the second step of a round, the Verifier asks for evidence in support of a randomly chosen property from the above list. The Prover reveals just enough to allow the verifier to check the property (on a statistical basis), but not enough to let the Verifier figure out what B is. For example, to prove that a function f is one-to-one, the Verifier could ask the Prover to decommit the array representing f.

(d) Argue that your protocol is complete: if the Prover does in fact know a feedback arc set of size k, then the Prover will always succeed.

(e) Give a lower bound (in terms of m and n) on the probability that one round of your protocol will catch the Prover cheating, if in fact the Prover does not know a feedback arc set B of size k.

(f) Argue that your protocol is *zero-knowledge*: the Verifier can *on his own* produce a transcript of his interaction with the Prover with exactly the same statistics as transcripts produced when the Verifier is running the protocol with a Prover who knows the feedback arc set B. (To make this problem simpler, you only need argue this for *one* of the properties above being tested.)

Problem 5-2. FHE

In class (lecture 17) we presented a homomorphic encryption scheme (Gen, Enc, Dec), where for (pk, sk) generated according to $Gen(1^k)$ and for any two messages $m, m' \in \{0, 1\}$, given pk, $Enc(pk, m)$ and $Enc(pk, m')$, one can compute both C^+ and $C^\times$, such that $Dec(sk, C^+) = m + m'$ and $Dec(sk, C^\times) = m_1 \cdot m_2$.

Recall the scheme presented in class, where $Enc(b) = BR + bG$, all the operations are mod q for some large prime q, and R is a random m-by-m matrix with 0/1 coordinates (B is the public key and G is a fixed "gadget" matrix). To decrypt a ciphertext $C = BR + bG$ compute tC (where t is the secret key) and output 0 if and only if tC is small (say smaller than $q^{1/2}$). In class, we showed how to do homomorphic computations on ciphertexts (i.e., how to compute C^+ and $C^\times$).

(a) Can the scheme as presented support an unbounded number of homomorphic operations? What happens to the ciphertexts when more and more homomorphic operations are performed. Explain your answer.

(b) Say we know (a priori) that we want to perform at most 5000 homomorphic operations on ciphertexts, where each operation is addition or multiplication. How would you set the prime q so that the scheme would support such homomorphic computations?

(c) Suppose two honest parties, one with input x and the other with input y, wish to compute $f(x, y)$, for some given function f, without revealing any information to each other beyond the output $f(x, y)$. Use the homomorphic encryption scheme, to construct a protocol for achieving this task.

Problem 5-3. Voting

This question asks you to vote using the "end-to-end verifiable" online election system "Helios" (`https://vote.heliosvoting.org/`).

Visit the Helios site. Browse the documentation. Then:

1. Create an election using Helios.
2. Vote in this election with your friends. (Challenge at least once of your cast votes.)
3. Close and tally the election.
4. Verify that your vote was properly tallied.

Then:

- Describe what you did in each of the above steps.
- Discuss the usability of the Helios system. What worked well, and what was confusing?
- Choose one aspect of the security architecture in either steps 2, 3, or 4 and describe it. What could be problematic about this feature (from a security point of view)?