Massachusetts Institute of Technology — Handout 4
6.857: Computer and Network Security — April 21, 2021
Professors Ronald L. Rivest and Yael Tauman Kalai — **Due:** May 10, 2021

Problem Set 4

This problem set is due on *Monday, May 10, 2021* 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. *Your solutions to all problems should be written up in a single pdf.* Have **one and only one group member** submit the finished pdf containing the problem writeups. Please title the PDF with the Kerberos of your group members as well as the problem set number. (i.e. *kerberos1_kerberos2_kerberos3_pset2.pdf*).

You are to work on this problem set in groups. For problem sets 1, 2, and 3, we will randomly assign the groups for the problem set. After problem set 3, you are to work on the following problem sets 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-staff@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! 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.

Problem 3-1. ZK problem

In this problem, we will consider two relaxations to the notion of zero-knowledge (ZK): witness-indistinguishability (WI) and witness-hiding (WH).

Recall that in a ZK proof, the prover tries to convince the verifier that $x \in L$ for some NP language L and element x. Namely, the prover is equipped with a witness w such that $(x, w) \in R_L$ where R_L is the NP relation corresponding to L, and uses w to prove that $x \in L$. For example, you can think of the language L as the language of all 3-colorable graphs, $x \in L$ as specific 3-colorable graph, and w as a legal 3-coloring of the graph x.

As we saw in lecture, in a ZK (interactive) proof the verifier learns nothing besides the fact that $x \in L$. A WI (interactive) proof only guarantees that for every $x \in L$ and for every two witnesses w_1, w_2 such that $(x, w_1), (x, w_2) \in R_L$ the verifier cannot distinguish between a proof for $x \in L$ that is generated using a witness w_1 and one that is generated using w_2. For example, if a graph has two valid 3-colorings, a verifier will not be able to distinguish between a prover who uses the first coloring as a witness and one who uses the second coloring as a witness. In a WH (interactive) proof, the only guarantee is that verifier doesn't learn a witness from the protocol.

For more information of WI and WH proofs, check out the original paper where these notions were defined: `https://www.isical.ac.in/~rcbose/internship/lectures2016/rt02feigeshamir.pdf`

Also, the Wikipedia entry for WI proofs may be helpful: `https://en.wikipedia.org/wiki/Witness-indistinguishable_proof`

(a) Argue that the hiding guarantee of ZK is stronger than that of WI and WH. Namely, argue that for every NP language L, a ZK proof for L is also WI and WH.

(b) Does WI imply WH, or does WH imply WI? Namely, is it the case that for every NP language L, a WI proof for L is also a WH proof for L, and vice versa? Explain why or why not.

Hint: consider the case where the elements in L have a single witness vs the case that they have multiple witnesses.

(c) Suppose a user wishes to prove knowledge of her secret key SK, associated with a public key PK. Moreover, suppose that the user uses this secret key to decrypt messages sent to her using the public key PK. Which proof should the user use so that the security of her ciphertexts would not be compromised? Is WI enough? WH? ZK? Explain your answer.

Hint: think carefully about the precise guarantees of each security notion, and think of the hiding guarantee needed for the encryption scheme to remain semantically secure.

Problem 3-2. Auditing an Election

In an election, voters cast ballots that indicate their single choice among the available options. In general, we can consider n ballots cast, each ballot choosing one of t options. Assume that the outcome of the election is determined according to a plurality vote. Then the outcome should be the option that received the most votes. However, it is possible that the reported outcome is "incorrect" somehow. We will consider false outcomes due either to error in tallying the cast ballots or election interference in reporting individual ballots.[1]

Our goal is to give a way to audit elections to provide assurance in the reported outcome. In this problem, you will play the role of an honest, one-person election authority.

Suppose you have just run an election with $n = 100,001$ voters on a proposition. The proposition had $t = 2$ options. Reportedly, $60,001$ voters cast ballots for "yes" (type 1) and $40,000$ of them have voted "no" (type 0). You have $100,001$ *paper ballots* numbered 1 to $100,001$. The paper ballots determine the true type of a cast vote. In particular, the i^{th} ballot has a true type $a_i \in \{0, 1\}$ that you can determine with a hand examination of the ballot.

However, paper ballots were run through a scanner and stored on a computer to report their type. The i^{th} ballot has a reported type $r_i \in \{0, 1\}$, which may or may not be equal to a_i; if the scanner is operating correctly, they are equal, but if not they may differ. That is, the scanner may be faulty or compromised.

Because votes in this case are either 0 or 1, we can determine which option won the election by taking the sum of all ballot types and determining if the sum is above or below $50,000.5$. The sum of the reported values is $60,001$, indicating "yes" is the outcome of the proposition election. You wish to audit this election by using a *Bayesian audit.* See

`http://people.csail.mit.edu/rivest/pubs.html#MR19`, and/or
`http://people.csail.mit.edu/rivest/pubs/MR19.pdf`

to read about this type of audit.[2] You can find a description of such a method on page 8 of the following article:

`https://www.usenix.org/system/files/conference/evtwote12/rivest_bayes_rev_073112.pdf`

At a high level, we can set up the problem as follows. You, as the election authority, have a collection C of n paper ballots. The reported outcome is R. To audit the outcome, you want to estimate the probability that *similar* collections C', also of size n, have outcomes *different* to R. That probability is what we need to test if it is below some threshold probability of failure ϵ, set exogenously (for example, by election audit laws). To generate similar collections C', this method involves initializing a sample of size s from the given collection C and "restoring" the rest of the simulated collection C'. A method called "Pólya's Urn" is used to restore the size of the simulated collection, which you can read about in the resources listed above. In particular, Appendix A.2 of the conference paper is useful here.

[1] In this problem, we do not consider other ways for a reported outcome to differ from what it "should" be. One example factor that could lead to such a difference is targeted voter suppression, which cannot be addressed by auditing paper ballots.

[2] Additional information can be obtained by Googling "Rivest bayesian audit."

(a) Write a program that audits an election using the method described above, printing either "OK" or "Not OK" at a 95% confidence in the election outcome, which is $\epsilon = 0.05$. In addition, it should give the size s of the last sample that your program took.

Sadly, the more realistic numbers given above take far too long to audit. Instead, we will audit a small election, with only $n = 301$ total votes cast, 90 of which are type 0 and the rest type 1. The reported outcome is $R = 1$. Using your program, determine if the election is OK or not OK. At what value of s did your program stop?

Please submit a copy of your code in the 3-2 Code submission on Gradescope.

(b) Give a brief, somewhat intuitive explanation of the profile simulation method. Although it is important, don't worry so much about getting bogged down in the details of Pólya's Urn. Instead, consider: what was the purpose of sampling s paper ballot types? What effect does treating that sample as the first s draws of the urn have on the rest of the simulated ballot profile?

Problem 3-3. Proof of Work vs. Proof of Stake

Proof of Stake (PoS) has been proposed as an alternative to Proof of Work (PoW) for deciding the next block in cryptocurrency blockchains. Unlike PoW, PoS does not require work in the form of hash computations to "win" the right to add blocks to the chain, making it substantially more energy efficient. Instead, users "win" the right to add new blocks to the chain based on how many coins they own in the currency, or their "stake." PoS protocols are typically more complex than PoW protocols, and they have quite different trust assumptions.[3] Despite this complexity, the low energy consumption and faster block times make PoS protocols very attractive, and several mainstream currencies use it today. In fact, Etherium, a very popular PoW cryptocurrency, is planning to switch to PoS!

In this problem, you'll look at some of the main differences between PoW and PoS from a performance and security perspective. To simplify PoS slightly, we will consider the case where each coin in the currency has a corresponding public and private key that can be used to prove ownership of the coin.

(a) Eve's get rich quick scheme
Eve is convinced that the best way to make money on cryptocurrencies is by earning the block reward. A block reward is a small amount of coins given to whichever user adds the next block to the chain. In order to do this she wants to increase her chances of being selected to propose the next block as much as possible. Her options are:

1. Invest in some ASICs for fast computation of hashes.
2. Invest in the currency itself by purchasing coins.
3. Lie shamelessly and claim to have been chosen to select the next block.

Which of these options would be most useful to Eve in a PoW protocol? Which would be most useful in a PoS protocol? Explain your answer.

(b) Liveness
In PoW cryptocurrencies like Bitcoin, miners are working hard computing hashes to discover the next block. Once found, they quickly announce it to let everyone know that they won the competition and the miners can move on to the subsequent block. Imaging a PoS scheme where the winner, who is allowed to choose the next block, is selected by picking a single coin from the currency uniformly at random.[4] Whichever user can demonstrate ownership of the selected coin is the winner and chooses the next block, announcing their choice to all other users. This scheme correctly satisfies the condition that users are selected with probability directly proportional to the number of coins they own, but it still isn't perfect. An obvious security concern is the case where a malicious user has been chosen to

[3]See this article for a high level description of PoW and PoS: `https://www.kraken.com/en-us/learn/proof-of-work-vs-proof-of-stake` Note that the hardware requirements they list for both PoW and PoS are more like *suggestions* for good performance and are in no way requirements for either PoW or PoS.

[4]This can be implemented by hashing the last block which outputs a pointer to one of the coins.

select the next block, but the scheme also has an inconvenient performance issue that isn't present in PoW.

How might "liveness," or availability of users, impact the block selection time in this scheme? Why is this not an issue in PoW schemes?

(c) Forking

In the last part we saw that choosing only a single user to propose the next block might have liveness concerns for a PoS cryptocurrency. In PoW, whoever **first** manages to compute a hash of the specified format gets to mine the next block and announce it to the rest of the users. All the users will then gossip about the new block until everyone is aware of it.

Suppose we tried to incorporate this method into PoS. That is, instead of choosing only one coin uniformly at random, choose 100 coins and any of the users who own the selected coins will be able to propose the next block. Like in PoW, honest users will accept the **first** valid block they hear about as the next block on the chain. In PoW, validity is decided based on the hash of the block, while in this PoS proposal a block is valid if and only if it was proposed by a user who owns one of the 100 selected coins.

What could go horribly wrong if this idea was used? (Hint: Note that the title of this question is "forking.")