Homework 4: Applications

Due: Monday, April 5 @ 11:59 pm EDT

Do not include any identifying information (name, login, Banner ID, etc.) on your handin. When you're done, please submit a PDF file to Gradescope where your answer to each problem (not each individual question) is on a separate page. Make sure to assign each problem on Gradescope to the correct pages in your writeup (or you may receive a deduction).

Unless otherwise stated, you should justify and explain all of your proposed attacks, schemes, protocols, defenses, etc., referencing specific properties when necessary. Precise, effective communication and presentation of your ideas and solutions in your writeups is something we highly value when grading. Similarly, be concise, but make sure that you don't elide intermediate details, take subtleties for granted, or make any assumptions without justification.

While we encourage everyone to discuss the problems on the homeworks, please remember that your homework solutions must be written up *entirely independently*, in your own words. You may not share your solutions with anyone (or read solutions written by others). You should not write your solutions while working with other students, and when you're writing your solutions, you should ensure that you independently understand and can reproduce your answers without referring to notes from collaboration sessions and consulting with other students—this applies even if you're submitting as a team.

Reminder: At the top of every problem (not question), you should write a *collaboration statement* that states who you discussed the problem with, who contributed major intellectual insights to your understanding of the problem, and any external resources you referenced while solving the problem. See the *Collaboration Policy* for examples. If you did not collaborate with anyone and did not refer to any sources, your collaboration statement should state as much.

Homework 4 Instructions

Answer Problems 1–5. If you're a CS162 student, you must also answer Problem 6. Please note that a portion of **part (c) of Problem 5 is due on April 3 at 11:59 pm ET**—no late days will be accepted for this part.

On this homework, you have the *option* of submitting a joint writeup with one other student for Problems 1–3. If you want to work *individually*, you should write up your answers independently, just as on previous homeworks. If you want to work with a partner, you should do the following:

- Find a partner—then, **have one partner fill out `https://forms.gle/jDk5kNUQe5ju1TbDA` to register your team.** There is no deadline to fill out this form, but, for the purposes of the *Collaboration Policy*, you need to do this before you start working together on a joint writeup.
- **You must individually write up your answers to Problems 4 and 5.** *If you're a CS162 student*, you must also individually write up your answers for Problem 6.
- You and your partner should work together to submit a **joint writeup for Problems 1–3 only**. Only *one* partner should hand in your final PDF, and you *must* use the team selection dropdown in Gradescope to select your partner's name when you hand in (otherwise, you'll receive a deduction).

If you're looking for a partner, you can use the "Search for Teammates" post on the course Piazza board.

When you're ready to submit, hand in your answers to Problems 1–3 to the "CS166 (Team-Optional): Problems 1-3" drop on Gradescope. You should then submit your *individual* writeups for Problems 4 and 5 to the "CS166 (Individual): Problem 4" and "Problem 5" drops. Finally, if you're a CS162 student, hand in your *individual* writeup for Problem 6 to the "CS162 (Individual): Problem 6" drop.

Questions start on Page 2.

Problem 1: Handshakes for Data Transfer

Suppose that a client with port P_C connects to a server with port P_S, and then performs the following TCP handshake and initial data transfer:

(1) Client sends SYN with sequence number `SEQ` $= A$ (where A is random).

(2) Server sends SYN-ACK with `SEQ` $= B$ (where B is random) and `ACK` $= A + 1$.

(3) Client sends ACK with `SEQ` $= A + 1$ and `ACK` $= B + 1$.

(4) Client sends DATA of length L_1 with `SEQ` $= A + 1$ and `ACK` $= B + 1$.

(5) Server sends DATA of length L_2 with `SEQ` $= B + 1$ and `ACK` $= A + 1 + L_1$.

(6) Client sends DATA of length L_3 with `SEQ` $= A + 1 + L_1$ and `ACK` $= B + 1 + L_2$.

Steps (4), (5), and (6) repeat until both sides are done sending data, at which point the connection is cleanly terminated using a FIN / ACK handshake.

Question a) Assume that the next transmission in this connection will be DATA of length L_4 from the server to the client. Fill in the table below with the headers of this TCP packet:

SEQ	ACK	Source Port	Destination Port

Question b) Consider each of the following adversaries who each have knowledge of the client and server's IP address. For each adversary model, answer the following prompt:

Show how the adversary can *hijack* the communication between the client and the server. By "hijack", we mean that the adversary should obtain unilateral control over the TCP connection in both directions, and thus be able to send any arbitrary data in the TCP connection. Then, calculate the *probability* that the adversary's attack works. You should propose an attack that gives the adversary the maximum possible probability of success.

(i) Abby, a network attacker who can observe traffic between the client and the server and can inject packets into the network, but cannot modify existing network traffic.

(ii) Charlotte, a network attacker who cannot observe or modify traffic between the client and the server, but can inject packets into the network.

Question c) *(Part (c) is unrelated to the previous parts.)*

Recall *TLS*, a protocol providing confidentiality of messages between web servers and clients. Explain why *basic key exchange* (RSA key exchange) in TLS fails to provide *forward security*.

Problem 2: Query IDs and DNSSEC

Recall that when one queries records from a DNS server, their query is associated with a *query ID*.

Question a) Explain why having DNS query IDs is more secure than *no* query IDs *(maximum 25 words)*.

Question b) Explain why having *randomized* DNS query IDs is more secure than having *sequential* query IDs *(maximum 25 words)*.

Question c) At the end of the *Networks III* lecture, we briefly discussed DNSSEC, a form of DNS that adds a layer of trust on top of DNS by providing authentication via *digital signature chains*. At a high-level, DNSSEC provides message integrity by providing digital signatures on all DNS records between nameservers—specifically, it provides signatures on the *message body* (containing the contents of the actual DNS records it is sending back to the querying server), but it does not provide signatures on the headers of the DNS record (such as the query ID, etc.). A nameserver recieving a DNS response can use the answering nameserver's public key to verify that the response was actually sent by that name server (and not another server).

What attack does DNSSEC prevent against that exists in the regular DNS protocol?

Question d) If we ask DNS for the IP address of a domain that does not exist, the DNS protocol simply returns an empty message body marked with the `NXDOMAIN` header, which signifies a *negative result* (that the domain did not have any associated records).[1] However, the empty message body presents a problem for DNSSEC, since then DNSSEC has nothing to sign!

One fix for the "nothing to sign" issue is to instead return a signature on the `NXDOMAIN` header. Then, the recieving nameserver could then use the responding server's public key to verify that `"NXDOMAIN"` was actually sent by the nameserver.

However, this is a bad idea in terms of security. Why?

Question e) DNSSEC resolves the "empty message body" issue described in part (d) by adding the `NSEC RR` record type, which is used to represent negative results.

The way the protocol works is as follows: on the case of a negative result, the nameserver returns a signed pair of domains that are alphabetically before and after the requested name. For example, suppose the following domains exist on records in the nameserver for `brown.edu`, in alphabetical order:

$$\{\ldots,\ \texttt{covid.brown.edu},\ \texttt{cs.brown.edu},\ \texttt{dam.brown.edu},\ \ldots\}$$

Suppose one makes a DNSSEC query for `crowd.brown.edu` (which is alphabetically between `covid.brown.edu` and `cs.brown.edu`). In response, the name server would return two records:

(1) An `NSEC RR` record that, in informal terms, states "the name in alphabetical order that comes after `covid.brown.edu` is `cs.brown.edu`"

(2) A `RRSIG` record that is a signature over the entire message in the `NSEC RR` record

Does this fully fix the issue identified in part (d)? Explain.

Question f) The protocol described in part (e) seemed unnecessarily complicated to Zachary, who proposed the following modification to DNSSEC: simply return a signature of the requested domain on a negative result. That is, return a record that, in informal terms, states "the domain `[insert-requested-domain]` does not exist", as well as a signature on that record.

Zachary says that this has several benefits: it avoids the need to maintain a sorted list of domain records; it also can substantially reduce the packet size of a negative result, which would increase performance.

As to be expected with these kinds of hypotheticals—this is a bad idea. Why?

[1] You can see this for yourself by using the `host(1)` command on an invalid domain, which will print out the message "`Host <domain> not found: 3(NXDOMAIN)`".

Problem 3: Onion-Flavored Handins

Recall Tor, a system for anonymous web browsing. Some of the CS166 TAs have established their own Tor network (depicted in Figure 1) to allow students to submit their handins "super-anonymously" on Gradescope. Students submitting a super-anonymous handin do not need to log in and no identifying information is associated with the application-level data sent to the Gradescope server. (This means TAs don't know who handed in a given super-anonymous handin.)

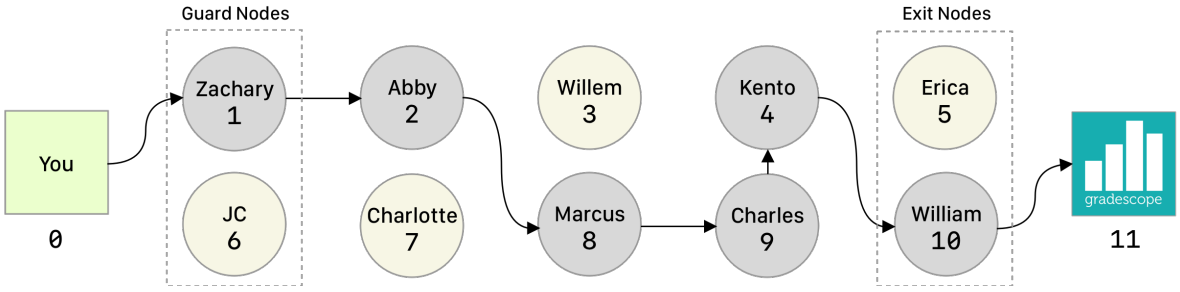

Figure 1: Tor network set up by the CS166 TAs comprising 10 nodes depicted by circles with IDs 1 through 10. Depicted with squares are the client machine of a student, with ID 0, and the Gradescope server, with ID 11. The Tor node with ID i ($i = 1, \cdots, 10$) has public key PK_i, which is known to all other Tor nodes and the client. For simplicity, we assume messages sent to Tor nodes are encrypted with public-key encryption while in reality, a combination of public-key and symmetric encryption is used.

Question a) The arrows in Figure 1 depict a Tor circuit between you and the Gradescope server. Assume that the format of each network packet is `[next-hop-id, data]`; for example, to send message `"Hello"` to node 1, one would transmit network packet `[1, "Hello"]`. Also, assume that the encryption of a message M with public key PK_i is denoted as $E(PK_i, M)$. To send a message M to the Gradescope server, what is the packet your Tor browser sends out in this circuit?

Question b) Bernardo uses HTTP (not HTTPS) when using the CS166 Tor network. Bernardo says TLS is not necessary to preserve *confidentiality* (that is, no one can read M other than Bernardo and the Gradescope server) because the onion packets are already encrypted. Given this setup, is the *confidentiality* of M maintained in the presence of an adversary who eavesdrops the communication between any two machines on the network? Explain.

Question c) Sierra *actively* controls the last link between node 10 and the Gradescope website *and* a router in the CIT through which many students simultaneously connect to the Gradescope site via Tor and HTTPS (see Figure 2). Is it possible for Sierra to determine which student submitted which super-anonymous handin? Explain. (*Hint:* Recall that the Tor circuit used by each student is randomly chosen, and thus the "wire distance" that messages travel along the Tor network is different for each student.)

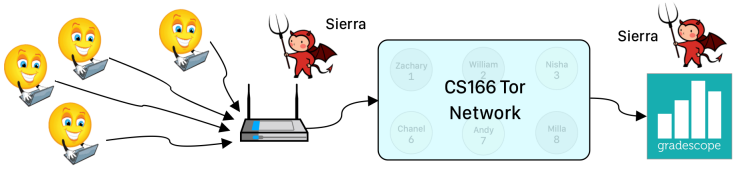

Figure 2: Multiple students using the CS166 Tor network through the CIT router.

Question d) Consider a modified version of the scenario from part (c) where you are the only user of Tor and Sierra is now a *passive* attacker (specifically, she cannot modify or add traffic sent from the Gradescope server). Does using the Tor network ensure that Sierra cannot identify your computer as the one who hands in a given super-anonymous handin? Explain.

Problem 4: Compression and Encryption

> *Zachary says: "As a reminder, each student must independently writeup their answers for Problem 4, even if you're working with a partner to submit a joint writeup for Problems 1–3. Also, while you may not have have previously seen this exact technical construction or application, you should be able to use a combination of the security mindset intuition you've built throughout the semester to gain enough information from the provided binary to solve this problem."*

One practical consideration of networking that we've abstracted away in the course so far is *data compression*. To improve performance, clients will often use a *compression algorithm* to reduce the size of data prior to sending it over the network.

At a high-level, compression algorithms work by identifying repeated substrings in a message and replacing equivalent substring instances with some short, unique sequence of bits. To uncompress the message, we can later replace the sequence of bits with the actual substring they originally replaced. In this technical question, we'll examine the implications of compression on confidentiality mechanisms like SSL and TLS.

The CS166 staff has developed a grading server on which they run autograding scripts against student submissions. Given the sensitive nature of student grades, we utilize a one-time-pad scheme to provide *perfect secrecy* on all communication with the server.

To interact with the server, TAs invoke a script that consumes (on `stdin`) a student name to grade. This sends an encrypted message containing the username as well as a fixed, *TA Secret* to the grading server. If you could extract the *TA Secret*, perhaps you could take over the grading server yourself! You've found a copy of the script (which you can run with the binary at `/course/cs1660/student/<your-login>/hw4/script`):

```
secret = "XXX-XX-XXXX" # Redacted; the X's are numbers from 0-9
while True:
    name       = raw_input("") # Read from standard input
    message    = "Grade %s; my TA identifier is %s. Repeat %s." % (name, secret, secret)
    compressed = zlib.compress(message)
    ciphertext = one_time_pad(compressed)
    send_to_grading_server(ciphertext)
    print(ciphertext.encode("hex")) # Prints the ciphertext as a hexadecimal string
```

Assume that the `one_time_pad` never runs out of pad material and is truly random.

Question. Is it possible to determine the value of the *TA Secret* in polynomial time in the length of the `secret`?

- If yes, submit a text file named `SECRET` that contains the *TA Secret* on the first line (and no other text). Also, submit the source code of any programs you used to derive it, and a text file named `README` that contains a justification of the polynomial runtime of your attack.
- If no, submit a text file named `README` that explains why such an attack is impossible. Your explanation should explain how the properties of the cryptographic construction used here guarantee the security of the *TA Secret*.

(*Hint:* Recall the *perfect secrecy* property of the *one-time-pad* cryptosystem covered in the *Cryptography I* lecture. Also, consider messages A = `"here is a long message"` and B = `"roberto!! or roberto!!"`. When uncompressed, both A and B are the same length. Will A take up more, less, or the same space as B when A and B are compressed?)

Problem 5: Cybercrime & State-Sponsored Cyberattacks

As a reminder, each student must independently writeup their answers for Problem 5, even if you're working with a partner to submit a joint writeup for Problems 1–3.

During lecture, we have been discussing cybercrime and and cyberattacks. Please read and refer to the following article in response to the following questions:

- `https://cs.brown.edu/courses/csci1660/files/docs/cyber_attacks.pdf`

Question a) What steps, if any, should the US take to deter state actors from attacking critical infrastructure such as electrical grids, transportation systems, and water supplies? What level of confidence in attribution would you require to consider this retaliation appropriate?

Question b) According to the author, why is international law unlikely to be successful at limiting state-sponsored cyberattacks? Do you agree?

Question c) *Instead of submitting an answer to part (d) in your writeup, please submit your answer as a new follow-up comment on the pinned "Discussion: Problem 5, Part (c)" post on the Piazza Discussion board.*

The article discusses the role that non-state actors, particularly large technology companies, have played in shaping norms in cyberspace given the lack of progress states have made in creating and enforcing norms. Should non-state actors take such an active role in shaping norms? Why or why not?

Piazza Instructions: Write a post of about 200-300 words (not including citations), and respond to at least one other post (about 150-200 words). For full credit, **posts are due by Saturday, April 3 at 11:59 pm ET, and response comments are due by Monday, April 5 at 11:59 pm ET.** Pursuant with the late policy in the *Syllabus*, *no late submissions for part (d) will be accepted*, regardless of whether or not you are using a late day for this homework. Feel free to cite outside sources to strengthen your response (though this is not necessary).

Problem 6: Mining Pools

Only CS162 students are required to complete this problem. As a reminder, each CS162 student must independently writeup their answers for Problem 6, even if you're working with a partner to submit a joint writeup for Problems 1–3.

In a *Bitcoin mining pool*, the pool administrator assigns to each worker in the pool a range of nonces to use in trying to solve the current puzzle. The worker who finds the solution reports the successful nonce to the pool administrator, who then shares the block reward with all the workers in the pool in proportion to the work they have performed.

What prevents a worker who has found the solution to the puzzle from taking the entire block reward themselves?