Weaver
Spring 2022

CS 161 Computer Security

Discussion 12

Question 1 ***A Tour of Tor*** ()

As a reminder, when connecting to a normal website through Tor, your computer first queries the Tor "consensus" to get a list of all Tor nodes, and using this information it connects to the first Tor node and, from there, creates a circuit through the Tor network, eventually ending at an exit node.

Q1.1 (4 min) Consider the scenario where you are in a censored country and the censor choses not to block Tor, the censor is the adversary, and no Tor relays exist within this country. How many Tor relays must your traffic pass through, including the exit node, to prevent the censor from blocking your traffic.

[✓ One] ○ Four
○ Two ○ Tor doesn't stop this adversary
○ Three

Solution: The censor doesn't block Tor and the relay is outside of the country, so one hop will get you safely past the censor. The censor will see you sending packets to an encrypted Tor relay but will not be able to determine who you're actually communicating with.

This is equivalent to using a VPN where the VPN server is in a different country.

Q1.2 (4 min) Consider the scenario where you are the only user of Tor on a network that keeps detailed logs of all IPs contacted. You use Tor to email a threat. The network operator is made aware of this threat and that it was sent through Tor and probably originated on the operator's network. How many Tor relays must your traffic pass through, including the exit node, to guarantee the network operator can't identify you as the one who sent the threat?

○ One ○ Four
○ Two [✓ Tor doesn't stop this adversary]
○ Three

Solution: Since you are the only user of Tor, the network operator just needs to look at the IP of the only person trying to connect to a Tor relay. The network operator can look through the list of IPs and see that you contacted a Tor relay regardless of how many relays you use.

Q1.3 (4 min) Consider the scenario where there is a single hostile Tor node but you don't know that node's identitity, and that node can be an exit node. You want to keep confidential from this node what HTTP sites you are visiting through Tor. How many Tor relays must your traffic pass through, including the exit node, to guarantee this adversary can't know what sites you visit?

- ○ One
- [✓ Two]
- ○ Three
- ○ Four
- ○ Tor doesn't stop this adversary

Solution: If you only use a single relay, then if that relay is hostile they will be able to see your request and the site you're visiting. If you use two relays, the first relay cannot see your request, and the second can see your request but doesn't know who it's from. So in either case, you are protected.

In other words, if the second relay is positioned between you and the hostile node, the hostile node will not know the request originated from you since it only sees the incoming request coming from "that other node." If the second relay is positioned between the hostile node and your destination, then while the hostile node knows the request comes from you, it doesn't know the destination since it forwards the request to "that other node."

Q1.4 (4 min) Consider the scenario where there are mulitple independent hostile Tor nodes but you don't know their identities, and these nodes can be exit nodes. You want to keep confidential from all these nodes what HTTP sites you are visiting through Tor. How many Tor relays must your traffic pass through, including the exit node, to guarantee that every independent hostile node can't know what sites you visit?

- ○ One
- [✓ Two]
- ○ Three
- ○ Four
- ○ Tor doesn't stop this adversary

Solution: The solution is the same as the previous question. Since the hostile nodes are independent (non-colluding), it doesn't matter that there are multiple. No individual node can ever know both your identity and the request as long as you use at least two relays.

Q1.5 (4 min) Consider the scenario where there are multiple colluding hostile Tor nodes but you don't know those nodes identities, and these nodes can be exit nodes. You want to keep confidential from all these nodes what HTTP sites you are visiting through Tor. How many Tor relays must your traffic pass through, including the exit node, to guarantee that the colluding system of hostile nodes can't know what sites you visit?

- ○ One
- ○ Two
- ○ Three
- ○ Four
- [✓ Tor doesn't stop this adversary]

Solution: Now, since the hostile nodes are colluding, you cannot ever be sure you are anonymous since you could get "unlucky" and have every node in your path be colluding hostile nodes.

Note that in real life, using three relays makes the probability of this happening negligible (assuming a certain amount of randomness in relay selection).

Q1.6 (4 min) Consider the scenario where there is a single hostile Tor node but you don't know that node's identity, and that node can be an exit node. You want to have data integrity for the HTTP sites you are visiting through Tor. How many Tor relays must your traffic pass through, including the exit node, to guarantee this adversary can't manipulate the data you receive from the sites you visit?

- ○ One
- ○ Two
- ○ Three
- ○ Four
- [✓ Tor doesn't stop this adversary]

Solution: The exit node could modify the HTTP response without detection before forwarding the HTTP response to you.

Question 2 ***Bitcoin*** ()

Assume a simplified Bitcoin model, where each block contains the following fields:

- `minerID`: The public key of the node who mined this block. Recall that the person who mined a block is given a mining reward in Bitcoin. Assume that a miner can redeem this award by simply referencing the block ie. the initial award is *not* stored as a transaction.
- `prevHash`: The hash of the previous block
- `transactions`: The list of transactions. Recall each transaction contains references to its origin transactions, a list of recipients, and is signed using the private key of the coins' owner.
- `nonce`: A value such that the hash of the current block contains the correct number of zeros

Assume that the hash of a block is computed as:

```
Hash(minerID || prevHash || transactions || nonce)
```

Bob wants to save on computing power by omitting certain fields in a block from being part of the hash. For each modified block hashing scheme below, select all the things an adversary with a single standard CPU can do.

Assume that if the adversary can come up with a modified blockchain of the same length, the rest of the network will accept it. Furthermore, assume the adversary has not made any transactions thus far. **Any option that could result in an invalid state should not be selected.**

Q2.1 (4 points) Each block hash is computed as `Hash(prevHash || transactions || nonce)`

[✓ (A) Modify a block to gain Bitcoin]

☐ (B) Given some amount of pre-computation, can consistently win proof of work

☐ (C) Modify some transaction amounts

☐ (D) Can remove any transaction in an arbitrary block by *only* modifying that block

☐ (E) None of the above

☐ (F) ——

Solution: An adversary can change the `minerID` of some past blocks to give themselves the mining reward. Note that this mining reward can't be used in a subsequent transaction or else we would reach an invalid state, but, at the very least, the most recently added block will always have a mining reward that hasn't been spent yet.

Q2.2 (4 points) Each block hash is computed as `Hash(minerID || transactions || nonce)`

■ (G) Modify a block to gain Bitcoin

■ (H) Given some amount of pre-computation, can consistently win proof of work

☐ (I) Modify some transaction amounts

☐ (J) Can remove any transaction in an arbitrary block by *only* modifying that block

☐ (K) None of the above

☐ (L) ——

Solution: Like before, an adversary can change any `minerID`s that haven't been spent yet since blocks no longer have a requirement on the past chain.

They can also precompute a valid nonce for a block they want to add, since the hash is independent of the chain.

Since the blocks aren't directly dependent on eachother anymore, the adversary can change any individual block. *However*, they can't remove a transaction if a future transactions makes use of it (this would be an invalid state).

They cannot modify a transaction amount because each transaction is signed.

Q2.3 (4 points) Each block hash is computed as `Hash(minerID || prevHash || nonce)`

☐ (A) Modify a block to gain Bitcoin

☐ (B) Given some amount of pre-computation, can consistently win proof of work

☐ (C) Modify some transaction amounts

☐ (D) Can remove any transaction in an arbitrary block by *only* modifying that block

■ (E) None of the above

☐ (F) ——

Solution: We can't modify `minerID`s anymore since the blockchain has dependence on them.

We can't consistently win PoW via pre-computation since the blocks form a blockchain.

We can't remove any transaction in an arbitrary block as this might cause an invalid state and we can't modify transaction amounts because of signatures

This is the end of Q2. Leave the remaining subparts of Q2 blank on Gradescope, if there are any. You have reached the end of the exam.