Week of April 23, 2018

Question 1 ***Mining Pools*** **(20 min)**

On April 20th, 2018, the estimated hash rate of the bitcoin network was $\approx 3\times10^{19}$ hashes/sec. The DragonMint 16T, one of the most powerful ASIC miners in the world, can do $\approx 1.5 \times 10^{13}$ hashes/sec.

(a) Alice has just purchased a DragonMint 16T and wants to start mining. Given that a block is mined by the whole network every 10 minutes on average, how long does Alice expect to wait until she mines a block?

(b) Why might Alice want to form a pool with other individuals like her, even though her expected income from bitcoin mining will go DOWN due to the cost of pool overhead?

(c) Alice wants her pool to mine at least a block per week on average. How many DragonMint 16T's would need to join her pool to accomplish this?

(d) Some members of Alice's pool are using slower hardware, and some only wish to mine at night when electricity is cheap. How should Alice fairly distribute the pool's income amongst the members, when some are contributing more than others? How can she efficiently estimate how much each member contributed?

(e) Alice's pool pays proportionally, but Bob tells Alice that he thinks this is a bad idea. Bob says that mining for her pool becomes less profitable when they haven't found a block in a while. Why might this be, and how could Eve take advantage of pools like Alice's to make more profit than someone who simply mines for one pool forever. Also, what are some ideas for how Alice could resolve this issue?

Solution:

(a) Both Alice and the whole network should expect to perform the same number of HASHES on average before mining a block. In 10 minutes the network performs $600 \times 3 \times 10^{19}$ hashes, which would take Alice $600\frac{3\times10^{19}}{1.5\times10^{13}} = 600 \times 2 \times 10^6 = 1.2 \times 10^9$ *seconds* $= 38$ *years*.

(b) Because the variance is too high, and she might have to wait decades before making any income at all!

(c) Changing the hash rate by a factor of X changes the time per block by $\frac{1}{X}$. There are $52 \times 38 \approx 200$ weeks in 38 years, so we would need 200 DragonMint 16T's

(d) Alice can distribute the income from each block proportionally to the number of hashes each member in the pool performed since the last block. She can estimate how much each member contributed through the concept of shares. Shares are essentially attempts at mining blocks where the hashes were very low (below some threshold), but not low enough. By counting how many shares each member found, Alice can estimate how many hashes they contributed to the pool without seeing each hash. This allows us the repurpose the bitcoin proof of work to also be a proof of work for our pool!

(e) The issue is that when a pool has a lot of shares built up, the next block they mine will get diluted much more than for other pools. The result of this is that the expected profit from mining for a pool with many shares is lower than from mining for a pool with fewer shares. Eve can exploit this by pool hopping, where she mines for a pool until the number of shares in that pool gets too high, and then hops to a different pool. Eve will now have a higher expected profit than miners that stay in one pool.

There are many other pool payout schemes, such as paying a flat amount per share, only considering the last N shares contributed to the pool, or valuing recent shares more, that attempt to combat pool hopping. They try to ensure that mining for the pool is always equally profitable.

Question 2 *Consensus* **(10 min)**

(a) Eve is buying a penguin from Alice. Eve generates and then sends Alice a valid transaction message, which transfers 100BTC to Alice's wallet. The signature is correct, and Eve has enough funds to make this transaction. Upon receiving and verifying the transaction, Alice gives Eve the penguin. What attack could Eve do to avoid paying Alice the 100BTC?

(b) What can Alice do to make sure she's actually received the money?

(c) Alice tells Eve she will wait until the next block is mined to determine if the transaction went through. Given that the last block was mined 9 minutes ago, and a block is mined every 10 minutes on average, how long does Eve expect to wait?

(d) Alice is unsure if waiting for the next block is secure enough. Let's say Eve controls a mining pool with a large fraction of the total network hash rate, and is trying to perform an attack similar to the one from part 1, even though Alice is now waiting for the next block to be mined with her transaction in it before releasing the penguin. What does Eve need to do to pull off the attack?

Solution:

(a) Eve can do a double spend attack, where she generates another transaction sending the 100BTC to herself, and races Alice to get hers confirmed first.

(b) Alice should wait for some blocks to be mined with her transaction in it. The more blocks she waits for, the more unlikely it is for her transaction to be orphaned.

(c) 10 minutes. The last 9 minutes of mining do not affect the probability of a block being found in the future.

(d) In order to perform the double spend, Eve needs to mine a chain where she sends the money to herself, and have it overtake the rest of the network. One option is to mine 2 consecutive blocks faster than the rest of the network does.

Question 3 ***Bitcoin Potpourri*** **(15 min)**

(a) Tired of generating new addresses for every transaction, Bob decides to use the same address over and over again for his transactions. Why might this be a bad idea?

(b) Alice wants to "anonymize" her bitcoin by moving them to new addresses without them being traced (for totally benign reasons). How can Alice, and many other users like her, accomplish this together?

(c) Alice currently uses a wallet application on her laptop to store her bitcoin. Alice is concerned about malware on her computer making her bitcoin vulnerable. She wants to use her laptop to send bitcoin, but doesn't want her bitcoin to be stolen even if her laptop is compromised. What are some ways she could accomplish this?

(d) Alice decides to use a hardware wallet to store her bitcoin. Her wallet looks like a usb drive with one button on the side. When she wants to send bitcoin, she inputs the transaction details into an application on her computer, and then presses a button on her wallet to confirm the transaction. When the button is pressed, it signs the transaction proposed to it by the computer. The button ensures transactions are only signed when Alice intended them to be, and the wallet's internals ensure no information about her private keys are ever leaked to the computer, only signatures. Is this scheme secure? How could attacker that compromised Alice's computer steal her bitcoin? What can Alice do to prevent this?

Solution:

(a) There is a security concern, and an anonymity concern here. For anonymity, reusing an address creates a link between different transactions, which compromises anonymity. An attacker could observe that different users both received bitcoin from the same person. For security, address could be concerning because of how bitcoin addresses work. Addresses are not public keys, but rather the HASHES of public keys. Therefore, even if there were a weakness in the asymmetric cryptography, attackers would never know what the public key for an address is until a transaction using that address is posted. With address reuse, people will know what the public key is for a bitcoin address once it has been used once, so if more bitcoin are sent to it later and the key is compromised, bitcoin can be stolen.

(b) A large group of users can "mix" their bitcoin together. Alice could pick a few random users in the group to send bitcoin to. Other users will randomly send bitcoin to Alice. They can organize offline such that each user ends up with the amount they started with.

(c) Alice needs to keep her private keys isolated from her computer. A hardware wallet can accomplish this. She would use her laptop to generate a transaction, and then send it to the hardware wallet. The wallet would then sign the trans-

action with the secret key it hold, and send it back. The keys are never exposed to the laptop, so a malicious computer cannot steal her bitcoin.

(d) The issue is that we don't know if the transaction being signed by the wallet is the same as the one Alice input into her computer. If Eve compromised the computer, she could make it send a different transaction to the wallet than the one Alice inputted, which transfers the money to Eve instead. To get around this, Alice needs the secure portion of wallet to display the transaction details before she confirms it. Remember, our TCB here is ONLY the wallet itself.

Question 4 ***Review: Crypto*** **(10 min)**

(a) Let E_0 be the AES block cipher with a key of all zeros. What property is E_0 missing to be a hash function?

(b) What is one advantage of CTR over CBC mode?

(c) Say RSA signatures are used with a secure hash function to create a signature S on the message M. What property of the hash function prevents an attacker from creating another message M' such that S is a valid signature on M'?

(d) Alice decides that she will generate her password by finding the SHA256 hash of an English word. True/False: this new password is more secure.

(e) Boogle decides to add client-side password hashing to all their login forms, using Javascript. This way, even if a TLS connection is compromised, the attacker will still not be able to login as that user. What is wrong with this scheme?

Solution:

(a) Cannot take arbitary length inputs

(b) CTR has parallelizable encryption.

(c) Hash functions are collision-resistant (and thus second-preimage resistant).

(d) False, same entropy.

(e) Attacker can pass the hash, does not stop them from logging in

A final note: do not hesitate to ask for help! Our office hours exist to help you. Please visit us if you have any questions or doubts about the material.