Distributed Trust and Blockchains	Date: *14 October 2019*
Instructor: *Sujit Prakash Gujar*	Scribes: *Shafiya Naaz, Ekta Gavas*

Lecture 10 : Bitcoin Storage

1 Recap

The following subsections briefly cover the contents covered in the previous class.

1.1 Mining pool rewards

Two most commonly used reward schemes :
1) Pay per share
2) Proportional
Pay-per-share model is the best for miners and the pool managers essentially absorb all the risk. In the proportional model, the miners still bear some risk proportional to the risk of the pool in general. Proportional payouts provide lower risk for pool managers.
There is also a concept of pool hopping. A clever miner might try mining in a proportional pool early in the cycle (just after the previous block was found), while the rewards per share are relatively high, only to switch ("hop") to a pay-per-share pool later in the cycle, when the expected rewards from mining in the proportional pool are relatively low.

1.2 Energy Requirements

Lower Bound : 6500 MW
Upper Bound : 6600 MW
Expected : 8333 MW

1.3 Other strategies

1) Forking the chain :
51% attack refers to an attack on a blockchain by a group of miners controlling more than 50% of the network's computing power. The attackers would be able to prevent new transactions from gaining confirmations, allowing them to halt payments between some or all users. They would also be able to reverse transactions that were completed while they were in control of the network, meaning they could double-spend coins.

2) Selfish Mining :
Do not reveal the block immediately.A selfish miner will maintain their own private chain, and publicly reveal it opportunistically in order to obtain greater rewards that would normally be granted based on their computing power to the mining pool.

3) Undercutting :
It is a mining strategy which involves producing new blocks not on the longest known chain, but instead 'undercutting' the longest chain by building a block, which collects less of the available transactions fees, on a shorter chain. This behavior creates a direct incentive for another miner to now build new blocks on the 'undercutting' chain as there exist a higher sum of transactions fees for them to collect.

2 Introduction

Storing bitcoins is all about storing and managing Bitcoin secret keys. Different approaches to key management offer different trade-offs between availability, security and convenience. The simplest key management method is to store them in a file on your own local device: your computer, phone, or some other kind of gadget that you carry, own, or control. But this option is not great for availability or security if you lose the device, if the device crashes and you have to wipe the disk, or if your file gets corrupted, your keys are lost, and so are your coins. Similarly for security: if someone steals or breaks into your device, or infects it with malware, she can copy your keys and then send all your coins to herself. So what we do is store a little bit of information - a little bit of our money - in our wallet and keep most of our money somewhere else.

3 How to use bitcoins

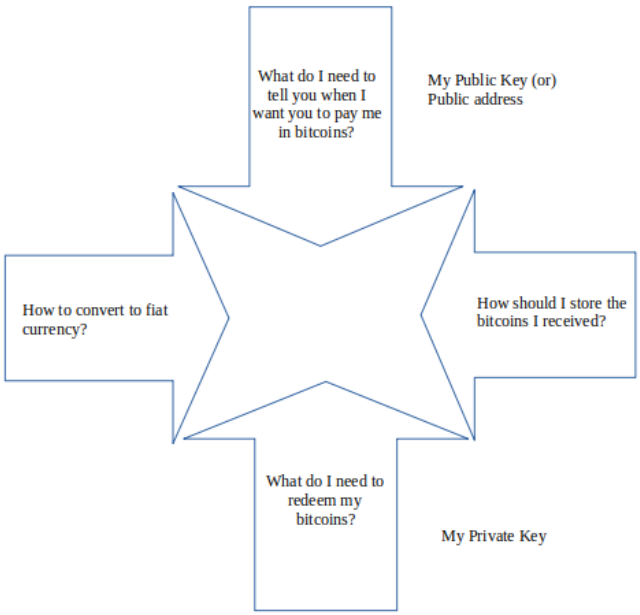

Figure 1

4 Encoding Keys : Base 58 encoding

We get 168 bit address after applying SHA256 to public key and then RIPEMD to output of that. To encode this address as a text string, we take the bits of the key and convert them from a binary number to a base-58 number. Then we use a set of 58 characters to encode each digit as a character; this is called "base-58 notation." Why 58? Because that's the total number of available uppercase letters, lowercase letters, and digits that can be used as characters (minus a few that might be confusing or look like another character). For example, capital letter "O" and zero are both taken out, because they look too much alike. Similarly small letter "l" and capital letter "I" are also taken out.

5 Vanity Addresses

Some individuals or merchants like to have an address that starts with some humanly meaningful text. For example, the gambling website Satoshi Bones has users send money to addresses containing the string "bones" in positions 2–6, such as *1bonesEeT-cABPjLzAb1VkFgySY6Zqu3sX*. Addresses are outputs of a hash function, which produces random-looking data, so how did the string "bones" get in there? If Satoshi Bones were simply making up these addresses, lacking the ability to invert hash functions, they wouldn't know the corresponding private keys and hence wouldn't actually control those addresses. Instead, they repeatedly generated private keys until they got lucky and found one that hashed to this pattern. Such addresses are called vanity addresses, and there are tools to generate them.

5.1 How much work does it take?

Since there are 58 possibilities for every character, if you want to find an address that starts with a specific k-character string, you'll need to generate 58^k addresses on average until you get lucky. So finding an address starting with "bones" would have required generating more than 600 million addresses! Such a search can be done on an ordinary laptop today. But the search becomes exponentially harder with each extra character in the desired name.

5.2 Speeding up Vanity Address generation

In Bitcoin, if we call the private key x, the public key is g^x. The address is $H(g^x)$, the hash of the public key. But exponentiation is a time-consuming step in address generation. The naive way to generate vanity addresses would be to pick a pseudorandom x , compute H(g^x), and repeat if the resulting address does not work. A much faster approach is to try x + 1 if the first x fails and continue incrementing instead of picking a fresh x each time. That's because g^{x+1} = g g^x , and we have already computed g^x , so we only need a multiplication operation for each address instead of exponentiation, and that's much faster. It speeds up vanity address generation by more than two orders of magnitude.

6 Bitcoin Key Storage (Wallets)

6.1 Storing Private Keys

1. Should be easily available and convenient to use.
2. Should be able to spend.
3. Should not be stolen.
4. Cannot store on laptop or mobile which we use to connect to the internet as they may get lost or may be hacked.

If we have only one public key private key pair, then all the money is accessible every time and thus is prone to be stolen.

6.2 Proposals

6.2.1 Paper Wallet

We can print the key material to paper and then put that paper in a safe or other secure place. Obviously, the security of this method is just as good or bad as the physical security of the paper that we're using. Both the public and private keys can be encoded in base-58 notation. Storing a small amount of key material is sufficient to re-create a wallet.

6.2.2 Brain Wallet

The second method we can use is called a brain wallet. This method controls access to bitcoins using nothing but a secret passphrase.The key trick behind a brain wallet is to have a predictable algorithm for turning a passphrase into a public and a private key.We can then generate an entire sequence of addresses and private keys from a passphrase, thus enabling a complete wallet.
However, an adversary can also obtain all private keys in a brain wallet if he can guess the passphrase. So the adversary can try various passphrases and generate addresses using them, This is called the Dictionary attack. Now if he finds any unspent transactions on the block chain at any of those addresses, he can immediately transfer them to himself.
Furthermore, unlike the task of guessing your email password, which can be rate-limited by your email server (called online guessing), with brain wallets, the attacker can download the list of addresses with unredeemed coins and try as many potential passphrases as he has the computational capacity to check. Note that the attacker doesn't need to know which addresses correspond to brain wallets. This is called offline guessing or password cracking.
Generating Memorable Passphrases
One passphrase-generation procedure that gives about 80 bits of entropy is to pick a random sequence of six words from among the 10,000 most common English words ($6 \cdot \log 2\ (10000)$ is roughly 80). Many people find these easier to memorize than a random string of characters.

6.3 Hot and Cold Storage

Storing bitcoins on a computer is like carrying money around in a wallet to be able to spend when needed. This is called 'hot storage'. Whereas, a bank resembles a 'cold storage' which is offline. Hot storage is risky but convenient, whereas cold storage is less convenient but more secure. To transfer coins from hot storage to cold storage and vice versa, respective public keys or addresses would be required. We don't use same address for hot storage and cold storage because it involves a lot of risk. This is because if the hot storage is compromised, cold storage also becomes vulnerable if they have same keys. So, for this reason, hot and cold storage have different set of keys so that if one is stolen, the other is safe. Would you like to send coins from hot storage to cold with the same address? The answer is no due to privacy and security reasons. For this, both should know each others' addresses or keys. To achieve this, the cold storage generates a batch of addresses when it comes online and shares it with hot storage. The drawback is that we have to periodically reconnect the cold side in order to transfer more addresses.

Hierarchical wallets Here, we generate address generation info, and rather than a private key, we generate private key generation info.

Cold side:

$$Private\ key\ generation\ info : seed, x, y, i^{th} \tag{1}$$

$$i^{th}\ private\ key : x_i = x + H(seed||i) \tag{2}$$

$$i^{th}\ public\ key : g^{x_i} = g^x.g^{H(seed||i)} = y.g^{H(seed||i)} \tag{3}$$

Hot side:

$$Address\ key\ generation\ info : seed, g^y \tag{4}$$

$$i^{th}\ public\ key : g^{x_i} = y.g^{H(seed||i)} \tag{5}$$

$$i^{th}\ address : H(g^{x_i}) \tag{6}$$

This satisfies the unlinkability property i.e it is not possible to infer that the keys come from same wallet. Also, even if the hot side is compromised, the private keys are still safe. Thus, this scheme supports many levels of security and so 'hierarchical'.

HD wallets An HD Wallet, or Hierarchical Deterministic wallet, is a new-age digital wallet that automatically generates a hierarchical tree-like structure of private/public addresses (or keys), thereby addressing the problem of the user having to generate them on their own. Popular HD key

generation includes Hardware wallets. Modern Hardware wallet providers are Ledger Nano S, Trezor, KeepKey, Mycelium (an Android app) etc. Still, it is required to generate address. Any compromise on access to private key may lead to stealing of all your bitcoins! This gives the notion of splitting the keys.

6.4 Splitting and Sharing Keys

Till now, we were storing the secret keys all at one place, which makes them more vulnerable.

The idea is to split the secret into N pieces, such that given any K pieces, we can reconstruct the secret given fewer than K pieces, but if we are given fewer than K pieces, we won't be able to learn anything about the secret.

The motivation behind is that to know the polynomial of degree K-1, we need K points. So, we generate N points on a curve of polynomial of degree K-1. This idea was presented by Shamir.

Let's say N=2 and K=2. P is a large prime and S is our secret (128-bit) in [0, P) and R is a 128-bit large random number. Then,

split: $X_1 = (S + R) mod P, X_2 = (S + 2R) mod P$

reconstruct: $(2X_1 - X_2) mod P = S$

Equation	Random parameter	Points needed to recover S
$(S + RX) mod P$	R	2
$(S + R_1X + R_2X^2) mod P$	R_1, R_2	3
$(S + R_1X + R_2X^2 + R_3X^3) mod P$	R_1, R_2, R_3	4

Good: Store shares separately, adversary must compromise several shares to get the key

Bad: To sign, need to bring shares together and reconstruct the key which may become vulnerable at that point. For this, we have an option for avoiding this single point failure.

Multi-signatures

This lets you keep shares apart and approve transactions without reconstructing the key at any point.Here, we collect money to script(Pay2Script). N keys are generated and coins can be redeemed only if K out of N sign.