Distributed Trust and Blockchains Date: *September 18th, 2018*

Instructor: *Sujit Prakash Gujar* Scribes: Kritika Prakash, Arjun Nemani, Kartik Garg

Lecture 12: Bitcoin Network, Nodes and Storage

1 Recap

In the following subsections, we provide a brief recap of the topics discussed in previous class:

1.1 Proof of Knowledge

In cryptography, a proof of knowledge is an interactive proof in which the prover succeeds in 'convincing' a verifier that the prover knows something. It is the proof of knowing some data x at time t, and if desired, it is established that x is known at time t, without revealing the value of x at time t. This proof acts as a commitment to x without knowing the value of x.

1.2 Proof of Clairvoyance

Proof of Clairvoyance requires that in the process of predicting an outcome of an event, one doesn't maliciously try to predict multiple or all possible outcomes of that event. Since such a proving mechanism requires linking the prediction with a time-stamp (for example, a newspaper time-stamp), we need to make sure that the malicious user cannot get away with time-stamping multiple predictions (outcomes) of an event.

1.3 Non-Fungibility

Bitcoins are non-fungible. In economics, a fungible good is one where all individual units are equivalent and can be substituted for one another. Fiat currencies are fungible as the currency does not maintain the history of transactions it has gone through. Every bitcoin has a unique history of transactions to it, starting from the coin-base transaction (creation of the bitcoin). So despite two bitcoins having the same value, they are differentiable due to their respective transaction histories - some bitcoins originate from more trust-worthy sources as compared to other bitcoins. Some sources are considered more trustworthy than other sources as the bitcoin network is pseudo anonymous - identity of nodes is compromised by observing behaviour patterns.

1.4 Meta-Data

Crypto-currency has the expressive power to represent anything, as it is non-fungible. We can now hope to build useful applications using authenticated meta-data attached to a currency such that it is difficult to duplicate tokens with the same meta-data. This way, all the anti-counterfeiting properties are inherited. At the same time, the underlying value of the currency is also maintained. But this also implies that all of the useful meaning of this meta-data is only as good as our trust in the issuer who signed it. The key question now, is whether this can this be applied to the bitcoin network or any crypto-currency.

1.5 Applications of Non-Fungibility and Meta-Data

- Vintage Currency - Eg. The first bitcoin
- Stadium Tickets
- Transfer of Property

1.6 Pay To Script Hash (P2SH)

Instead of asking the sender to pay bitcoins to the hash of the receiver's public key, the receiver can ask the sender to pay bitcoins to the hash of a script, given the condition that to redeem those coins, it is necessary for the receiver to reveal the script that has the given hash, and to provide data that will make the script evaluate to true.

This simplifies and standardizes the process of payment for the sender, by making it independent of the receiver's script. The sender achieves this by using the Pay-to-script-hash (P2SH) transaction type, which has the required semantics.

P2SH also has a nice efficiency gain. Miners have to track the set of output scripts that havent been redeemed yet, and with P2SH outputs, the output scripts are now much smaller as they only specify a hash. All of the complexity is pushed to the input scripts.

1.7 Coloured Coins

The main idea is to stamp some bitcoins with a color, and track that color stamp even as the coin changes hands, just like we are able to stamp metadata onto a physical currency. A bitcoin stamped with a color still functions as a valid bitcoin, but additionally carries the specified meta-data.

Coloured coins are coins with special properties. The term 'Coloured Coins' loosely describes a class of methods for representing and managing real world assets on top of the bitcoin blockchain. The most popular proposal for implementing coloured coins in bitcoin is called *OpenAssets*. To achieve this, in one transaction, called the issuing transaction, we insert some extra meta-data that declares some of the outputs to have a specific color. Assets are issued using a special PaytoScriptHash address. If you want to issue colored coins, you first choose a P2SH address to use. Any coin that transfers through that address and comes in without a color will leave with the color designated by that address.

Now we recap the topics discussed in Lecture 10 as well as Lecture 12:

1.8 Merkle Trees

Each bitcoin block stores a merkle tree of it's transactions such that verification of a transaction is easier and faster, even though the merkle tree takes more space. The bitcoin block-structure consists of two main parts: block header and transaction data. The transaction data is stored as a merkle tree. In the process of solving the puzzle, the miners hash only the block header, which is a few bytes long.

1.9 Coin-Base Transactions

With the creation of each new block in the blockchain, the miner of that block obtains a reward. The way the miner gets this reward for block creation is that the miner adds a reward addressed to himself (which is a standardized amount) and adds a record of this reward in the block he mines. This "transaction" is known as a Coin-Base Transaction.

A Coin-Base transaction creates new coins. It does not redeem a previous output, and it has a null hash pointer indicating this. It has a coin-base parameter which can contain arbitrary data. The value of the coin-base transaction is the block reward plus all of the transaction fees included in this block.

Note that this transaction has no input transactions or sources from where it gets the reward. This transaction is a special kind of transaction where bitcoins are created. One such Coin-Base transaction exists per block. This transaction is not signed, like other transactions. Instead, a field called *coinbase* is selected for this transaction.

The output of a Coin-Base Transaction is customizable by the miner. The miner could use scripts to redeem the bitcoins later. Each block has a special transaction in the merkle tree called the coin-base transaction.

1.10 Bitcoin Network

The bitcoin network is a peer-to-peer network in which all nodes are equal. There is no hierarchy, and there are no special nodes or master nodes. It runs over TCP and has a random topology, where each node peers with other random nodes. New nodes can join at any time.The goal of the bitcoin network is to propagate all new transactions and new blocks to all the bitcoin peer nodes. But the network is highly imperfect, and does a besteffort attempt to relay this information. The security of the system doesn't come from the perfection of the peertopeer network. Instead, the security comes from the block chain and the consensus protocol.

One can join a bitcoin network as a miner (or full-node) or as a thin client (SPV client) to receive payments in bitcoin. About 90 percent of the nodes run *Bitcoin Core* (C++) - a program to decide which block contains valid transactions. It maintains decentralization in the network.

One protocol to join the bitcoin network is called the *Gossip* protocol, which is a flooding algorithm. In this protocol, any person knowing an already existing node can join the network. Nodes connect to random peers and there is no geographic topology of any sort. Now say you launch a new node and want to join the network. You start with a simple message to one node that you know about. This is usually called your *seed node*, and there are a few different ways you can look up lists of seed nodes to try connecting to. You send a special message, saying, Tell me the addresses of all the other nodes in the network that you know about. You can repeat the process with the new nodes you learn about as many times as you want. Then you can choose which ones to peer with, and youll be a fully functioning member of the bitcoin network.

Similarly, other nodes in the network can also send special messages to you. If the network finds that you are inactive for more than three hours, then the network (your neighbours) forgets you.

Figure 1 shows the size of the blockchain over time.

1.11 Lightweight Nodes (Thin or SPV Clients)

A thin client or a Simplified Payment Verification client (SPV client) stores only the block headers of the most recent blocks, unlike the full node, which stores the entire blockchain. Such a client requests for transactions as required to verify incoming payment. The vast majority of nodes on the bitcoin network are lightweight nodes. Wallet programs typically use an SPV node. Lightweight nodes store only a relevant fraction of the block chain, specific to their use. They cannot act as miners.

The cost savings of being an SPV node are huge. The block headers are only about $1/1,000$ the size of the block chain. So instead of storing a few tens of gigabytes, its only a few tens of megabytes. Even a smart-phone can easily act as an SPV node in the bitcoin network.

An SPV node doesnt have the security level of a fully validating node. It can only validate the transactions that actually affect it. So they essentially trust the fully validating nodes to have validated all the other transactions that are out there.

1.12 Role of a Bitcoin Full Node

- Check if transactions inputs are in UTXO.

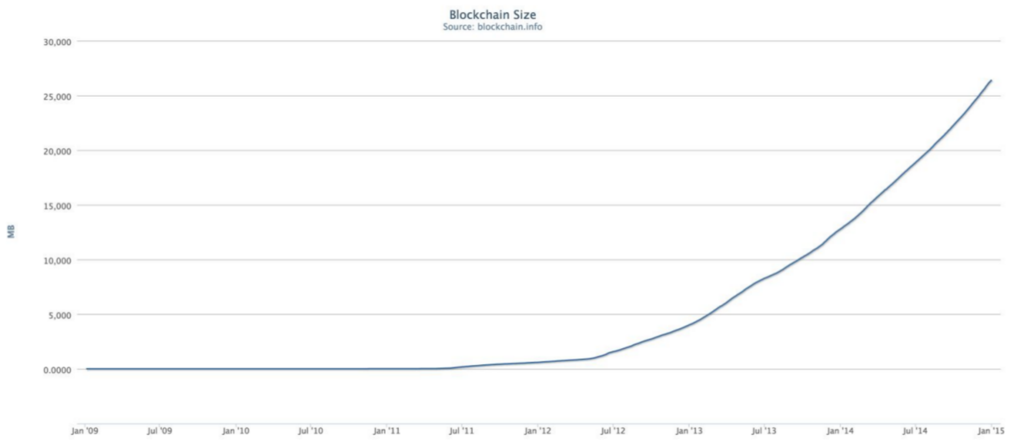

Figure 1: Blockchain Size [4]

- Ensure it is not a double spend attack.
- Execute output script of input transaction along with script signature.
  - If true add it to transaction protocol.
  - If true in previous step , broadcast transaction.
  - If a transaction is repeated, do not broadcast it again (to avoid flooding).

1.13 UTXO: Unspent Transaction Output

An Unspent Transaction Output (UTXO) is an output of a blockchain transaction that has not been spent, and can be used as an input in a new transaction. In the case of a valid blockchain transaction, unspent outputs (and only unspent outputs) may be used to effect further transactions. The requirement that only unspent outputs may be used in further transactions is necessary to prevent double spending and fraud. Figure 2 shows the variation of the UTXO set size with time since the origin of the bitcoin network.

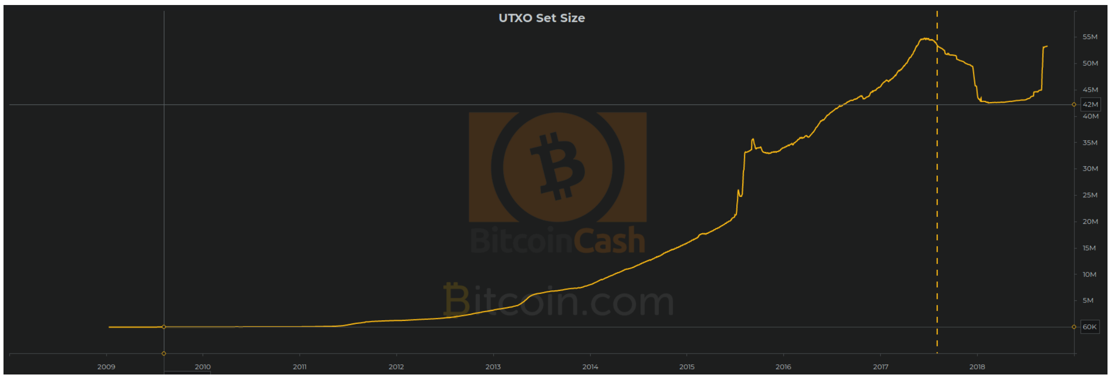

Figure 2: UTXO Set Size Vs. Year[2].

2 Overview of Lecture

The following topics were covered in this lecture,

1. Incentives to join as a full node
2. Things that cannot be undone without consensus
3. Effects of block size on block propagation time
4. Changing the protocol
5. Bitcoin Storage
6. Hot Storage vs Cold Storage

3 Incentives to Join as a Full Node

So far we have assumed that we would be able to pick an honest node from a bitcoin network randomly. But if there are financial incentives for the participants we cannot assume that a node will be honest. As we know that an honest node creates block which ultimately ends up on the long-term consensus chain , so now we are going to use bitcoins to incentivize the nodes by paying certain fees to miners that maintain bitcoin network, i.e., participants will be paid for creating blocks. Following are the two incentives mechanisms in bitcoin:

3.1 Block Reward

- The node that creates a block includes a coin-creation transaction in that block.
- This transaction is basically a payment that node receives in exchange for its service.
- Block adds recipient address in the transaction which is generally its own address
- Block reward started at 50 bitcoins and at present it is 25 bitcoins which halves every $210,000$ blocks.
- Block reward is halved every four years hence it limits total bitcoins to 21 million and by 2040 block reward will run out.
- Only those nodes whose blocks get added in long-term consensus chain (honest nodes) receive block reward.

3.2 Transaction Fees

- Every bitcoin transaction spends zero or more bitcoins to zero or more recipients.
- The difference between the amount being spent and the amount being received is the transaction fee (which must be zero or more).
- A transaction must follow three conditions.
  1. Size of transaction < 1000 bytes.
  2. All the output values must be 0.01 bitcoins and higher.
  3. Priority must be high.
- Priority = (Σ input age * input value)/(transaction size)

4 Things that Cannot be Undone Without Consensus

- Average duration between two consecutive blocks is 10 minutes.
- Block size is ≤ 1 Mb.
- A Satoshi is 10^{-8} bitcoins (a Satoshi is smallest bitcoin denomination)
- Number of bitcoins that would ever be minted are 21 million
- Bitcoin network can handle only 7 transactions per second.
- Bitcoin reward structure.
- Only couple of Hash functions($SHA256, RIPEMD160$) and only one signature algorithm $ECDSA$ is available.

Among all the above the most worrying factor is that only 7 transactions can be handled per second by bitcoin network whereas visa transactions can handle 2000 transactions per second on average.

5 Effects of Block Size on Block Propagation Time

- Block propagation time is the average time new block takes to propagate to every node of the network.
- It is directly proportional to the block size because of the network bottleneck

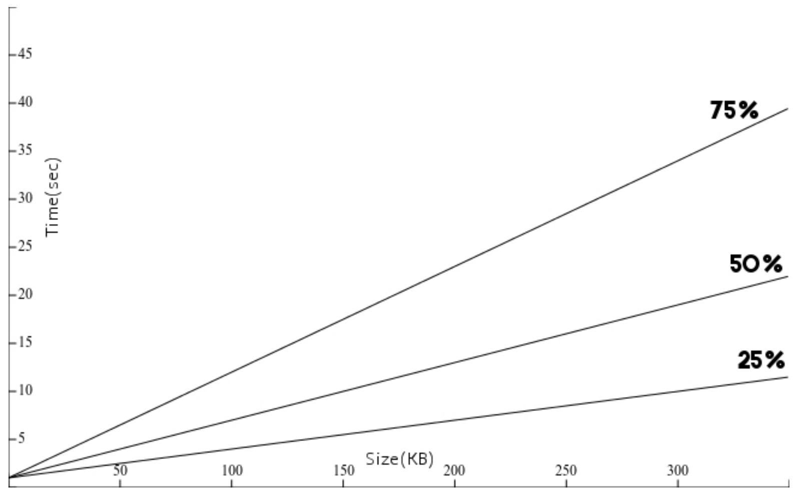

Figure 3: Block Reach[4].

Figure 3 shows the average time that it takes a block to reach various percentages of the nodes in the network.

6 Changing the Protocol

Adding new features to the existing bitcoin protocol is very complicated task. Suppose a new version of software is released, and some nodes have still not upgraded. So in this scenario in which most of the nodes have upgraded while few nodes are running the old software the consequences depend on the type of changes. There are two types of changes to the rules of bitcoin, known respectively as hard forks and soft forks.

6.1 Fork

- Bitcoin forks are defined as changes in the protocol of the bitcoin network.
- Fork between majority and remaining minors when they don't agree on something.
- Forks are typically conducted in order to add new features to a blockchain.

6.1.1 Hard Fork

- Hard fork can be thought as an update in software which is not backward compatible, and is described schematically in Figure 4.
- New rules support more flexibility.
- All nodesof miners, merchants and users will need to upgrade to the new nodes to be able to validate the new blocks.
- If you do not join the upgraded version of the blockchain then you do not get access to any of the new updates or interact with users of the new system whatsoever,
- There will be two versions of blockchain which will never merge.

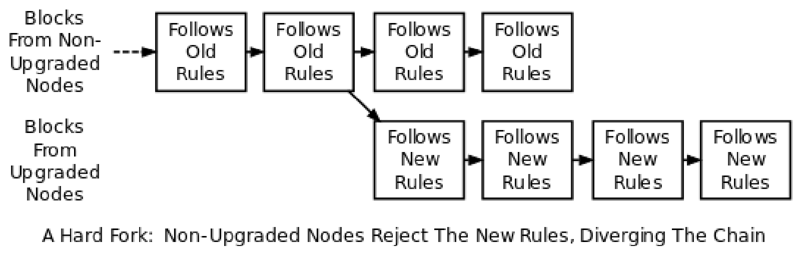

Figure 4: Hard Fork[1]

6.1.2 Soft Fork

- Soft fork can be thought as an update in software which is backward compatible
- New rules are more strict and tighter.
- Older versions of the bitcoin software will recognize new blocks.
- When a soft fork change is made, all nodes (whether upgraded or not) will continue to recognize new blocks and maintain consensus on the blockchain.

7 Bitcoin Storage

To spend a bitcoin you need to know

- Public info saved on the blockchain (Worth of coin, identity, etc)
- Private info that is the secret key that is needed to verify the owner

[1]https://bitcoin.org/img/dev/en-hard-fork.svg

We don't need to store the public info, as we can get it back anytime from the blockchain, but we do need to store the private key.

So, effectively storing a bitcoin boils down to storing and managing your secret bitcoin keys.

There are 3 aspects of storage that we need to keep in mind

- Security - Making sure no one else spend your coins
- Availability - Making sure you can spend your coins when you want
- Convenience - Making sure key management is continence and retentively easy to do

All different approaches to key management offer different trade offs between these three properties.

On one end of the spectrum is a simple mobile app that lets you transfer funds with a few clicks (high convenience) but if you lose the device, you lose your bitcoins (low availability) also if your mobile is stolen then the thief can simply send all the coins to himself. (low security)
On the other end of the spectrum is a procedure to encrypt and write the key to a flash drive and keep it locked in your Bank's safety deposit box. While this has very low convenience, the chances of losing the key or it getting stolen are relatively extremely low.

7.1 Bitcoin Address

A bitcoin address is an identifier that represents a possible destination for a bitcoin payment.

The address is a hash of a public key, and can be shared without any security risk.

We use addresses instead of public keys so that even if a vulnerability is found in the encryption your coins can still be safe since the public key isn't even known to anyone until you spend the money, only the hash (address) is known.

So, when you have a private key, you can use that to derive the public key, which you can use to derive the address/hash. But not the other way round. That's why you only need the private key backed up.

There are two major ways to encode the address

- Base58 String

  The address is read as a Base58 number and characters are assigned to each digit 0-57, then the string can be shown as a sequence of characters.

  We use Base 58 because that is the number of alphanumeric characters after removing similar looking characters (Zero "0", Uppercase o "O", Uppercase i "I" and Lowercase L "l"). So the 58 characters are *123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghjklmnopqrstuvwxyz*

  Bitcoin Addresses start with "1" or "3" and are usually 32-34 characters long[1].

- QR Code

  Another way to encode addresses is to make a 2D bar-code which can be scanned.
  This is to provide convenience, for example in a store, where you can click a photo of the bar-code and wallet software will automatically convert it into the actual address, to which you can send the money to.

  We have seen similar methods in Indian payment applications like *PayTM* and protocols like UPI.

7.2 Vanity Address

Some individuals or merchants like to have an address that starts with some human-meaningful text. For example, the gambling website *SatoshiBones* has users send money to addresses containing the string bones in positions 2-6, such as *"1bonesEeTcABPjLzAb1VkFgySY6Zqu3sX"*

So how did this website get this address, as explained above you cannot even get the public key from an address, how can we get the private key of it?
We cant. The procedure to create vanity addresses is a brute-force approach. We repeatedly generate private keys and until we get its public key hashes to the address we want.
There are many tools which can help you generate vanity addresses including Bitcoin Vanity Gen which will even send you the keys via snail mail to ensure security. You can also generate the address yourself offline using tools like vanitygen.
So, how much would it take these tools to generate such an address?
Suppose I want an address with my name *Arjun* which is 5 chars long and case sensitive.
Since each character in the address can have 58 possible values and my name is of length 5, I will need to generate, on average, 58^5 addresses to find an address containing my name. That is over 600 Million addresses! This can be done on a normal computer these days without much time required, but as we increase the length of the string, it gets exponentially tougher to find a matching address.

7.3 Bitcoin Wallets

- A wallet software keeps track of all your coins and manages all the details of your keys.
- It makes things simple and convenient with a nice user interface.
- If you want to send, for example, $10.50 worth of bitcoins to your friend the wallet software would give you some easy way to do that.
- Creating a new public-private key pair is easy, and you can utilize this to improve your anonymity or privacy by using new pairs for different things.
- Wallet software also gives you a simple interface that tells you how much is in your wallet and when you want to spend bitcoins, it handles the details of which keys to use and how to generate new addresses and so on.

8 Hot Storage Vs Cold Storage

In the following section we are going to describe two major types of bitcoin storage, *Hot* storage and *Cold* storage. We will also see the various different types of cold storage available and their special properties.

- Storing bitcoins on your computer is like carrying money around in your wallet or your purse. It is convenient but relatively risky. This is called Hot Storage.
- Cold Storage in contrast is "offline", it is not connected to the Internet and it is archival. It is safer but not very convenient.
- It is similar to how we carry cash in our wallets or purses but we have our life savings stored somewhere else.
- It is thus best to use a combination of both hot and cold storage, when saving your bitcoins.
- Obviously both of these should have different addresses so that even if your hot storage is compromised, your life savings would still be secure.
- Both of them should know the public key / addresses of the other to facilitate transfer of coins.

- Since the cold storage is offline, there cannot be any connection between the two stores. Thankfully, cold storage does not need to be online to receive coins, since the coins are not actually stored there but only the private keys.
- So if at anytime you feel the amount in your hot storage is uncomfortably large, you can send the coins to your cold storage.
- The next time the cold storage comes online, it will fetch information from the blockchain and update itself with information about all received coins.
- For privacy and other reasons we want to be able to receive each transaction at a different address with different private key. But since the cold storage is not online, how do we communicate to the cold storage what addresses we will be sending the coins to?
- One naive approach is to get a batch of usable addresses from the cold storage when it comes online, but what happens when we run out of these addresses? Towards this, we use *Hierarchical Address Generation* described next.

8.1 Hierarchical Address Generation

- A more effective solution is to use a hierarchical addresses. It allows the cold side to use an essentially unbounded number of addresses and the hot side to know about these addresses, but with a short, one-time communication between the two sides.
- In hierarchical address generation, instead of generating a single address we generate what is called "address generation info" and rather than a private key we generate what is called "private key generation info".
- Given the address generation info, we can generate a sequence of addresses: we apply an address generation function that takes as input the address generation info and any integer i and generates the i^{th} address in the sequence.
- Similarly we can generate a sequence of private keys using the private key generation info.
- The important cryptographic property here is that the i^{th} private key generated is a valid key for the i^{th} public key generated, as if the pair was generated using a normal Key Generator.
- Figure 5 is a visualization of this process.
- Not all digital signature schemes have this property but ECDSA, the scheme on which bitcoin works, does have this property.

Normally an ECDSA private key is a random number: x
The corresponding public key: g^x.
For hierarchical key generation, we need two random values: k and y
Private key generation info: k, x, y
i^{th} private key: $x_i = y + H(k||y)$
Address generation info: k, g^y
i^{th} public key: $g^{x_i} = g^{H(k||i)}.g^y$
i^{th} address: $H(g^{x_i})$

- One another important property is that keys generated by this method are not linked to each other in any way. There is no way to determine if the final key was generated using a hierarchical generator or a simple one.
- The only way you can infer that a set of public keys were generated using a hierarchical generator is if you have the address generator info.
- So this property does not hold true if the hot storage is compromised.
- Since this scheme supports arbitrarily many security levels it is called hierarchical address generator.

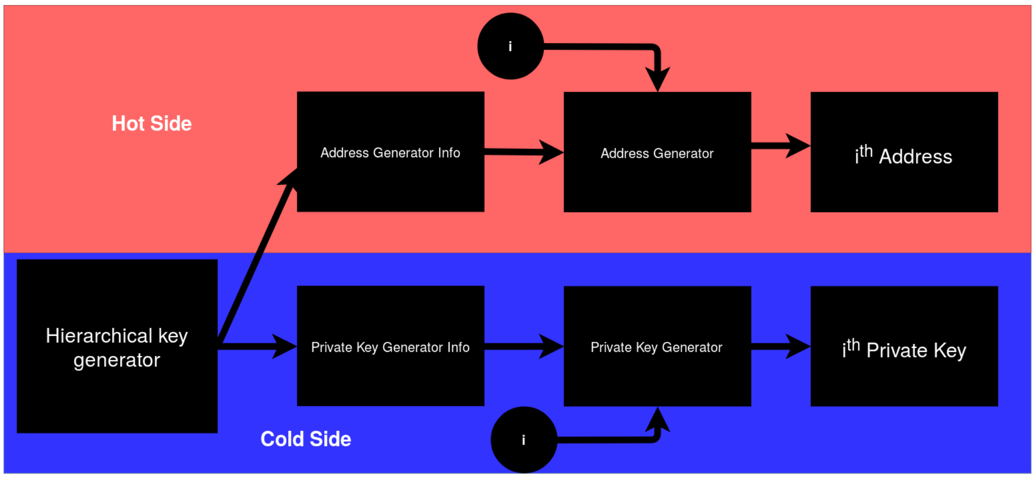

Figure 5: Hierarchical Address Generation

8.2 Types of Cold Storage

8.2.1 Naive Approach to Cold Storage

- The naive approach is to store it in some kind of device and put that device in a safe.
- It might be a laptop computer, a mobile phone or tablet, or a thumb drive.
- The important thing is to turn the device off and lock it up, so that if somebody wants to steal it they have to break into the locked storage.

8.2.2 Paper Wallets

- The second way is to use a paper wallet.
- We print both the keys onto a paper and then keep this paper into the safe.
- If anyone gets access to this paper, they can access any coin sent to this address or address if we are using a hierarchical generator.
- Thus it is very important to keep this paper safe.
- The paper has both the public and private key in both Base58 and 2D Bar-code encodings.

8.2.3 Brain Wallets

- The third way is to use a brain wallet.
- This approach does not need any long term storage device like laptop, flash drives or paper.
- The access control is done by using a secret pass-phrase and nothing else.
- This property is be particularly useful in situations where you have poor physical security, like when youre traveling internationally.
- Brain wallets work by applying a predictable algorithm on the pass-phrase to generate public and private key.
- Suppose we have a hash function that can convert input text string into a private key and then use the private key to get the public key and thus the addresses.
- Again combining this with a hierarchical generator, we can get a full wallet with many addresses based on just one pass-phrase.

8.2.4 Dictionary Attack in Brain Wallets

- Because of the way brain wallets work, if an adversary can guess the pass-phrase of the user, then then he/she would gain access to all the addresses and all unspent transactions stored in any of those addresses.
- As always in computer security, we must assume that the adversary knows the procedure you used to generate keys, and only your pass-phrase provides security.
- So an adversary may use a dictionary attack to try to find the a pass-phrase which generates a set of keys with unspent transactions.
- The adversary does not need to know who the pass-phrase and the addresses belongs to and since this is not a targeted toward a specific user, it leaves no trace at all.
- When an adversary tries to do such an attack to guess the password of your email account, the web-server will employ some kind of rate-limiting to stop the attack. (You can even try this on *research.iiit.ac.in*). After a few failed attempts you are blocked from trying to log into the website.
- But the attacker does not need to access to the server to try a pass-phrase and can brute force this offline. This is called *offline guessing* or *password cracking.*
- There are some precautions we can take such as using a slower hash function so that the adversary takes longer to go through the pass-phrases. This is called key stretching. We have to be careful not to make it too slow, otherwise honest users may get annoyed.
- We should also use long memorable pass-phrase with high entropy. A common misconception is that using a shorter pass-phrase with complex characters is more secure than using a longer password with simpler characters. Figure 6 is a cartoon from the famous XKCD [3] on this topic.
- If a brain wallet pass-phrase is inaccessible (like if forgotten, and cant be guessed) then the coins are lost forever.

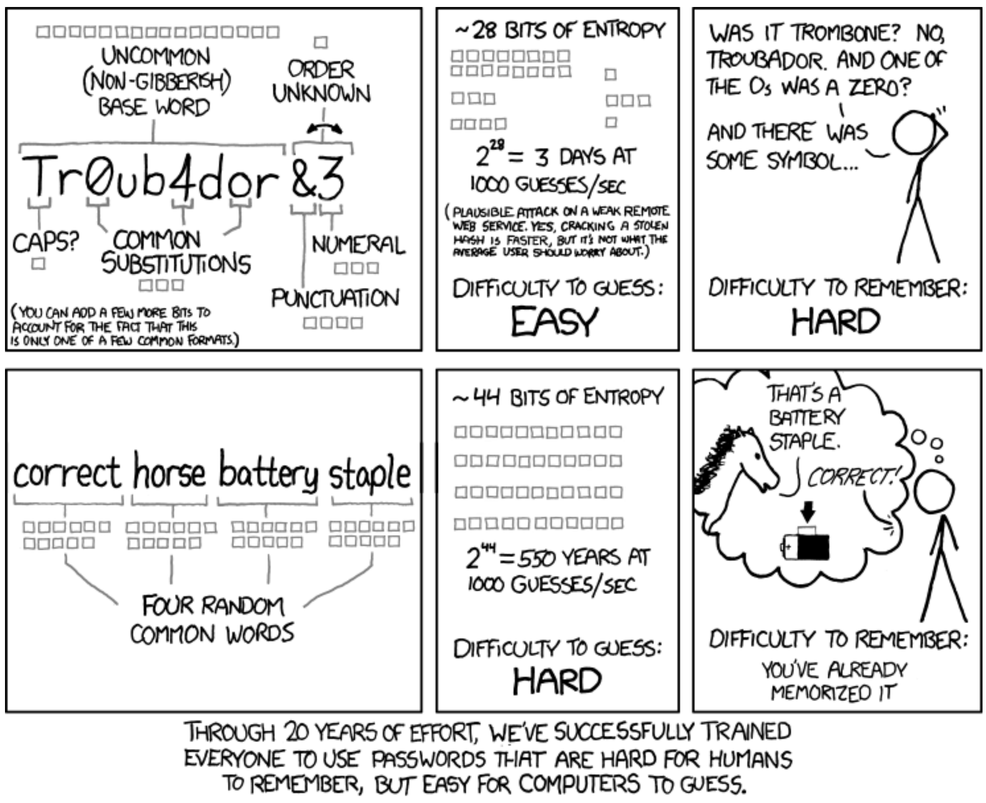
UNCOMMON (NON-GIBBERISH) BASE WORD
ORDER UNKNOWN
Tr0ub4dor&3
CAPS?
COMMON SUBSTITUTIONS
NUMERAL
PUNCTUATION
(YOU CAN ADD A FEW MORE BITS TO ACCOUNT FOR THE FACT THAT THIS IS ONLY ONE OF A FEW COMMON FORMATS.)
~28 BITS OF ENTROPY
2^28 = 3 DAYS AT 1000 GUESSES/SEC
(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)
DIFFICULTY TO GUESS: EASY
WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE Os WAS A ZERO?
AND THERE WAS SOME SYMBOL...
DIFFICULTY TO REMEMBER: HARD
correct horse battery staple
FOUR RANDOM COMMON WORDS
~44 BITS OF ENTROPY
2^44 = 550 YEARS AT 1000 GUESSES/SEC
DIFFICULTY TO GUESS: HARD
THAT'S A BATTERY STAPLE.
CORRECT!
DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT
THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

Figure 6: Cartoon [3]

References

[1] Wikipedia Contributors. *Bitcoin — Wikipedia.* `https://en.wikipedia.org/wiki/Bitcoin`. Accessed on 2018-09-24.

[2] Saint Bitts LLC. *Bitcoin.com Charts.* `http://charts.bitcoin.com`. Accessed on 2018-09-24. Sept. 2018.

[3] Randall Munroe. *Password Strength.* `https://xkcd.com/936/`. Accessed on 2018-09-24.

[4] Arvind Narayanan. *Bitcoin and Cryptocurrency Technologies. A Comprehensive Introduction.* Princeton Univeristy Press, 2017, pp. 93–94.