**Question 1**

**General solutions** (regardless of the approach)

For the most of the mentioned user requirements (add, edit), the server can simply send back the hash of the inbound file along with an ACK of the operation success; and for file retrieval, the server can compute and send the file hash along with the timestamp to the user. The hash function algorithm must be strong (most importantly, collision-proof), such as SHA-3. Also when the transmission occurs one way or another, the server must authenticate itself via methods like SSL (essentially, to sign transmissions with its private key so that the user can verify the other side using the public key).

To prove that the file was not tampered with since the last session, user (either with or without letting SecureBox server know about such actions) can apply the digital watermark onto all files user sends to SecureBox. That way when the files are retrieved back to the user, the user can verify whether the files were tampered with or not. This can, again, be done similarly to SSL (the user signs file with their secret key, and then once the file is retrieved back from the server, they verify the file with the secret key they have)

As to prove that some file does not exist, there are more drastic measures required. I can think of the three possible approaches, where the first is block-chain-like, the second is rather extravagant, and the third approach combines these two. Bear in mind that all approaches also add extra layers to meet the other requirements (“to add, edit, delete, and retrieve files, and to list the folder contents”), however, because they are being put in the first place just to solve the “prove that file does not exist if such” problem, and add a burden of the additional and vast computational and storage requirements, perhaps in reality (that is, outside of scope of the question) this requirement may need to be lifted in favour of the operational cost savings. However, assuming this requirement is not being lifted (i.e. sticking to the question), assuming how much is user’s anonymity and file security are important in contrast to the computational and storage requirements required, any approach can be found appropriate for the SecureBox’s needs and resources.

As a side note, such system is likely to store the illegal materials and eventually, the policing authorities are to be interested in interfering the illegal material sharing. If the users are to see that the website does not cooperate with the police, especially for the long periods of time (for example, several decades), there is going to be some end-user trust in the system

**Approach #1: Blockchain-like.**

***Description****:* SecureBox launches a complete blockchain log of operations being done to the files and shares it to everyone interested can download the full blockchain and launch a competitor server. Along with the full log of file operations, the core server may want to provide with the parallel ‘reduced’ blockchain download option, which contains hashes and timestamps of operations but does not contain the exact changes being made to the files, to solve the majority of other problems with the less computational and storage cost.

Note that the service costs are not necessarily being included into the blockchain, contrary to the Bitcoin approach (essentially, because we design a file storage system, not a cryptocurrency)

***How such approach adds extra layers towards the other requirements***: The full the blockchain adds integrity towards all operations done to the files. The reduced blockchain still solves the “file non-existence” and “tamperproof” problems via lookup (via checking the timestamps of all operations done to file), but is not suitable to provide the services of file uploading, editing and downloading to the user.

***How user can see their folder contents****:* To cut costs, the users are generally financially encouraged to store the pointers to their files locally, however, there is a paid service of user files lookup run by the concise-blockchain holder online services.

***How much is such system secure:*** Very similarly to Bitcoin and other cryptocurrencies, such system is as secure as user habits are. Essentially, the users are assumed to be anonymous and as all files are accessible via the full blockchain, the security comes through the users’ anonymity. The users are generally advised to have a new account per every file upload and download, and pay for the services through every new cryptocurrency wallet.

***How system is kept being financially stable:*** First of all, as mentioned above, the user-file lookup comes as a paid service to encourage users to keep their file ‘pointers’ locally and reduce unnecessary computations. But just like user-file lookup, depending on the service popularity and statistics, the other file upload, storage, edit and/or download services would come at a cost (which should be relatively low to the user). Since anyone is able to download both the full and reduced blockchain and launch the same service, the rates are capped at whatever globally storage and computational costs are, similarly to how the real-world commodities prices are capped at the market price through the capitalistic economical system. Perhaps at the very beginning, SecureBox must have at least another, artificial (meaning ‘forced’) but real competitor server bearing the same blockchains (similarly to the two-party system in the US), to keep end-user prices naturally reasonable by giving the end-user a choice. Once (and if) SecureBox becomes popular, the full, and perhaps reduced, blockchain downloading service might become a paid service as well.

***Possible vulnerabilities:*** Generic blockchain vulnerabilities. But most prominently the 51%-attack vulnerability, especially at the early stages.

***Service operational costs*** *(abstracted):* Relatively high storage costs, relatively low computational (CPU) costs.

**Approach #2: Integrity-based:**

***Description***: There are archival standards that, for the lack of the better term, require the output archives to be completely integral and ‘irreparable’ via the known means. That is, even if a single byte is altered or even a bit is being flipped, all the files are no longer extractable. Additionally, if even a minor change occurs to any file, the whole archive file’s data is being completely changed (similarly to how a good hashing algorithm’s result changes completely even by a minor change in the input). The best example I know of, is the 7z (pronounced ‘seven-zip’) archive type. If a 7z archive’s any bit is altered, while the file structure is still accessible, the files themselves within the archive are no longer recoverable, and empirically, if a minor change occurs to the 7z archive contents, the whole byte sequence changes completely Internally, SecureBox is to store the user files as the single 7z (or any other archive type having the same properties) archive file, or multiple archive files of about the same size. If a new file is being added, or an existing file is being edited, the server extracts the most suitable archive file’s contents (or the only archive file’s contents, in case of a singular archive file), adds changes and archives it back, (assumingly) changing the whole archive’s body byte archive’s body byte sequence. The user can also download the whole archive file itself, for verification (explained below) or other purposes.

***How such approach adds extra layers towards the other requirements***: At the very least, data tampering is not as easy as it would have been in case of the plain data storage. However, to make matters even worse for the attacker (and better for the end-user), we can send something extra to the user once user-data is changed (either by new files, editing or deleting the existing files), that is, we can send a random/user-defined but relatively big chunk of bytes within the archive file (for example, the 100 bytes beginning from 60% of the archive). We could as well send the hashsum of such chunk, but the chunk itself is very small (in the example, 100 bytes), SecureSafe can send the bytes themselves). Assuming there is no known effective way to produce a tampered archive file with these 100 bytes being the same, the change for Eve (the hypothetical hacker) to fabricate an archive with the archive body chunk’s same 100 bytes is,

Which, in practice, means ‘impossible’.

***How user can see their folder contents****:* To cut costs, the users are generally financially encouraged to store the directions of where to find which files locally. That is, in case of a singular archive, store the path within the archive, and in case of multiple volumes, store which volume the file is located in. However, unlike for the blockchain approach, because archives in general (and 7z in particular) designed in such a way that getting the contents listing without data extraction is very fast, perhaps SecureSafe can we designed to provide the user contents without necessarily financially forcing users to keep the listing information locally, allowing to get listings for free for the user (and covering the listing expenses as part of the payment for the other services provided)

***How much is such system secure:*** Compared to the pure approach #1 (blockchain), the system is much more protected and designed towards anonymity. However, the user is to trust the service alone instead of the blockchain structure, but to tackle this problem, we can allow, of even enforce the user to set up the password onto the archive, which will then be verifiable upon necessity by the user by downloading the whole archive and checking if password is indeed being set up

***How system is kept being financially stable:*** Simple and profitable. Because such service is one of a kind, and unlike for the blockchain-based approach, the service is unlike to face any competitors offering as secure service as this for this concept, hence our service will have its niche if successful

***Possible vulnerabilities:*** Generic traditional web-service vulnerabilities (DDOS, phishing, mainframe vulnerabilities etc). However, the biggest danger comes from the fact that the archives can be corrupted by flipping a single bit, hence we need to make sure the archives are not being corrupted by the natural data decay, forcing to use RAID (likely RAID 6) and hence forcing to deploy more storage equipment than otherwise would have been necessary

***Service operational costs*** *(abstracted):* Relatively high storage costs, relatively high computational (CPU) costs.

**Approach #3: Combined:**

***Sidenote*:** Because in vast majority of the aspects, ‘combined’ implies “the pros of the first approach plus the pros of the second approach, reducing the cons of them both”, to keep the statements concise (and reduce the amount of copy-pasting), the descriptions of this approach are going to be written in comparison to the previous two approaches

***Description****:* The blockchain benefits of the first approach, combined with the integrity of the second approach. Essentially, implies the previous two approaches applied together with a minor difference.

The difference is, because the archives’ body byte sequence is (assumed) to change completely upon any file change, the blockchain no longer has the “file edit” logs, which are then completely replaced with the ‘file replace’ logs (that is, the ‘archive replace’ logs).

***How such approach adds extra layers towards the other requirements***: The approach adds the blockchain benefits of the first approach, and coming one layer down, the approach adds the archive-properties benefits of the second approach.

***How user can see their folder contents****:* To cut costs, the users are generally financially encouraged to store the pointers to their files locally. There could be services to restore the user folder structure for the cost, however, because such services will be required not only to scatter across the blockchain, but also to unpack the archive(s), the computing time will be

And thus the price to be charge will be as high accordingly

***How much is such system secure:*** Essentially, the security of the first approach on the blockchain layer + the security of the second approach on the archive layer.

***How system is kept being financially stable:*** Because the blockchain is being applied, there is no way to keep the monopoly, especially at the latter stages of the product. The revenue is capped at the maximum revenue of the first approach (that is, capped by the free global market). In other words, the users can still be charged for using the service, however the prices we can charge are limited by what the competitors have to offer.

***Possible vulnerabilities:*** The blockchain vulnerabilities from the first approach. The second approach’s vulnerabilities are to be neglected as they can be easily corrected with the use of blockchain.

***Service operational costs*** *(abstracted):* Relatively high storage costs, relatively high computational (CPU) costs.

**Question 2**

1. Most obvious attack possible is that the user’s computer can be compromised by a keylogger (and a mouse-logger or a screen recorder, or perhaps both). Once the private key is generated, a mouse-logger or a screen-shooter can allow the hacker to get the user’s private key once they connect back to the internet. And with the hacker having the victim’s private key, once the user receives the cryptocurrency, the hacker can steal it, transferring it to their wallet.
2. As it was discussed previously, it is not impossible for the criminals to obtain the authentic online certificate by various means. It is possible that the hacker can claim and register a seemingly the same yet a phishery website with the same appearance, such as

Then get a certificate for it, and once that’s done with, gain money from the unfortunate visitors. If everything is done correctly, the user’s browser will still show the “https” tick, despite the website being fraudulent.

1. The so-called ‘trusted’ website can be compromised on itself, either with or without the owners’ knowledge. For example, it could be that each 10000th transaction fraudulently goes to the getaway wallet (either hacker’s or owner’s) instead of the end-user. Since one in 10000 transaction is statistically not going to undermine the trust toward the website, the money can being stolen without any further consequences towards the website reputation.

**Question 3**

1.

As a side note, the more data is to gather, the more likely the hacker is to guess n

2. Because for the p out of p+1