Procedural Seed: A Procedural Cryptocurrency Seed Manager

duudl3
duudl3@protonmail.ch
duudl3.xyz

The main problem with cryptocurrency wallet security is the recommended measure of writing down wallet recovery seeds on paper for maximum security against hacking or theft. The more wallets you download for various cryptocurrencies, the more paper you need. This can be quite difficult to manage with the various wallets a person may have. Procedural Seed proposes a system of recovery seed management that is paperless, and secure. The primary goal of this management system is to not store the wallet recovery seed at all.

Procedural Generation

Procedural generation, in essence, is the method of using the basic laws and accepted rules of math to retain data without ever needing to store it. A good example of procedural generation in action is the video game *No Man's Sky*. *No Man's Sky* uses procedural generation to generate a fully functional universe without ever needing to store the data in the game files. This allows for expansive gameplay and exploration while keeping the game file size low, as the data for the planets, stars, flora, and fauna is never stored on the disk of the computer, and is generated on-the-fly.

The implementation in Procedural Seed works the same. The only data stored on the disk is the name of the wallet, and the cryptocurrency the wallet is for. This is not actually needed, but for a clean user experience it is stored so the user does not need to manually generate the entry each time when the user needs to recover a wallet. The only other piece of data to generate a completely unique recovery seed is a master password.

Due to the nature of procedural generation, the system never needs to check if the master password entered is correct. The user will find out if the master password is correct if the wallet is recovered correctly. If it's not, then the user can simply relaunch the program and input the correct master password for the correct recovery seed. However, for a clean user experience I may look into storing a salted hash of the master password on the disk of the computer so the user can have confirmation that the master password is correct. This is not needed, but may be implemented for a fluid and stress-free user experience.

The System

On the launch of the program, a prompt will appear asking for a master password. This is data-point one of three for the recovery seed. The user is responsible for memorizing this, and it should be something they can remember on their own. Data-points two and three are the wallet name, and cryptocurrency name respectively. When entering the master password and entering the program, the master password will be salted and hashed multiple times using SHA256 and the original data of the master password will be thrown out for additional security measures, so the original master password is not stored in memory.

When generating a recovery seed using the implemented algorithm, the system will hash all three data-points and dump the original to avoid keeping the data in memory, this is to keep the original data out of the computer's memory as much as possible.

masterPassword
|
V
"...caefbbacf348acbefaba7cacba7cba..." (Salted hashed master password, "SHMP")

walletName
|
V
"...ca7c1209392cacb139..." (Salted hashed wallet name, "SHWN")

cryptocurrencyName
|
V
"...defcacb183acb193acbeff..." (Salted hashed cryptocurrency name, "SHCN")

SHMP + SHWN + SHCN = recoverySeed

Due to the nature of hashing and the basic laws of math, if one piece of data changes within the equation then the recovery seed that is generated will be completely different. For example:

$$1 + 1 + 1 = 3$$
and
$$1 + 2 + 1 = 4$$

Since the original data is hashed and the original master password is thrown out, the only data that is stored in memory is the hashed equivalent of the database entry. The only data stored on the computer's disk is the list of wallet names and respective cryptocurrency. This is all basic and public knowledge on the system of normal wallets, as the wallet name of various wallets like Electrum is the name of the file on the operating system, thus the data being stored is not an issue. The only attack vector an intruder may have is the fraction of a second when the program launches and the master password is salted and hashed before opening up the database.

If generating a new wallet for a coin, the user will need to generate the recovery seed in Procedural Seed first, then select "recover wallet from seed" in their chosen wallet. This process also applies to users recovering old wallets as well.

In conclusion, introducing a system like this into cryptocurrency could be beneficial to new users, and even more experienced users, as the process for generating wallets and securely storing recovery seeds would be much more efficient and user-friendly, with the same level of security as storing it on paper, without wasting paper.