How a Seed Phrase is Created

Jeff Hong ⋮ 8-10 minutes

A ***seed phrase*** is an ordered set of 12-24 words that is generated by your wallet and stores all the information that can be used to recover your wallet.

For the most part, seed phrases follow Bitcoin Improvement Proposal: 39 (BIP 39). BIP 39 standard describes the process in how a mnemonic phrase (a group of easy to remember words) is generated and can be used to recover your wallet in the event your wallet fails.

The order of the words in your seed phrase matters and the words themselves come from a **specific wordlist** known as the BIP39 wordlist consisting of 2048 commonly used words.

Not just any 12-24 words from the BIP39 list will work. There is a specific structure required based on the BIP39 standard for a valid seed phrase.

We're going to take you through the steps of generating a 12 word seed phrase.

Let's Start with Entropy

A seed phrase begins with something called *entropy*.

What is *entropy*?

In the case of data or information, entropy is a measure of "randomness". To put it simply, you can think of entropy as a very large random number that is **extremely** unlikely to be randomly generated ever again.

How is this random number generated? It can be generated on a secure device (like a computer without external connections like internet, WiFi, bluetooth, etc.) or flipping a coin (H=1, T=0). But regardless of the method, it should be **truly random**.

In the case of a seed phrase this starts out as a random binary number (meaning only 1's and 0's) such as:

`11011010010000101100101111001001110010000111111000010010100000001100001101001100101001011110011111001000010`

We're going to call each 1 and 0 a **bit** and this **sequence of bits** is what we'll call our **entropy**. You might notice that it is 128 bits long which is required for 12 word seed (256 bits for a 24 word seed). This is important in being able to complete the next steps.

Now that we have our entropy, we're going to turn it into the words we all know and love.

Now Let's Make Some Numbers

Remember when we said the BIP39 wordlist consists of 2048 words. Each word is represented by the number in which the words are alphabetically ordered starting with 1 (abandon = 1, ability = 2 . . . zoo = 2048)

In binary, a set of 11 (1's and 0's) bits can represent a number between 0 and 2047.

If you're interested in the math behind binary conversion, see below:

A binary conversion to a decimal number (i.e. a "normal" number that we are familiar with), is done by taking the sum of a sequence numbers dictated by whether there is a 0 or 1 in a certain position of the binary.

The order of the binary dictates whether you multiply a 0 or 1 times 2^n where n is based on the descending positional place of the 0 or 1 (starting at the total number of characters minus 1).

For example, 11100101001 can be converted to decimal by the following:
$(2^{10} \times 1) + (2^9 \times 1) + (2^8 \times 1) + (2^7 \times 0) + (2^6 \times 0) + (2^5 \times 1) + (2^4 \times 0) + (2^3 \times 1) + (2^2 \times 0) + (2^1 \times 0) + (2^0 \times 1) = 1833$

Thus with our entropy, each set of 11 bits can represent a number (the first set of 11, 1-11, representing the first number, the second set of 11, 12-23, representing the second number, etc.)

```
11011010010 = 1746
00010110010 = 178
11110010011 = 1939
10010000111 = 1159
11100001001 = 1801
01000000011 = 515
00001101001 = 105
10010100101 = 1189
11100111110 = 1854
01000010001 = 529
01011000010 = 706
1111111 = ?
```

You might notice there are enough numbers for 11 complete sets of 11 bits (which is enough for 11 words).

But what about the *last* (in this case 12th word) *set*? There are 7 bits leftover but you need 11 bits for a number in this case. Where do you get the last 4 bits?

(A 24 word seed phrase would only have enough for 23 complete sets of 11 bits, enough for 23 words. You would have 3 bits leftover and require an additional 8 bits)

The Checksum

The last 4 bits are known as the **checksum**. A checksum is a number (in this case 4 bits) with the purpose of *detecting errors* in a set of data (in this case, your seed phrase).

The checksum is calculated by first by hashing your entropy (that long, random sequence of 1's and 0's).

Wait, what is hashing? Hashing is the process in which you take **an input** (your entropy) and **transform it into a different output** (another number or set of characters) using a unique set of instructions or an algorithm. This set of instructions is known as a *hash function*.

Why would I want to do that?

Well, you can transform a long, cumbersome number into a small, more practical number that can check if that long number is correct.

For example, say I want to communicate the a sequence of numbers to someone else over the phone.

The numbers are 35199234551. Over the phone I repeat each number one by one. Now I want to check with the person if they wrote it down correctly.

I have a couple options, I can repeat each number again OR I can tell the person to add up the individual numbers and make sure they get a sum of 47. If they don't get 47, then we know immediately something is wrong. The instruction of adding up the individual numbers is the "hash function" in a sense.

This is an extremely simple example and hash functions are way more complex to ensure the data is indeed correct among other benefits.

The hash function used for BIP39 seed phrases is known as **SHA-256**, a cryptographic hash function developed by the NSA and the basis of the bitcoin protocol. It will transform your entropy into a hexadecimal (a sequence of characters consisting of 0 - 9 and A - F). This process is typically done in a secure setting (i.e. a device that cannot be compromised via an external connection)

In our on-going example, putting our entropy through a SHA-256 hash function will result in the following hexadecimal:

`**c**ed25fa131e86640ffc9517b590f84877e40ad20f7f1ae88707ec79945d0d454`

Note: In order to do a SHA-256 on your entropy, you must ensure it is being interpreted as binary data *or it will* **not** *be the correct result. Using an "online" SHA-256 function is* **not** *viable. Rather, this can be done in your terminal using the command:*

echo
1101101001000010110010111100100111001000011111100001001010000000110000110100110010100101111001111100100001000101011000
| shasum -a 256 -0

The "-0" specified the data should be interpreted as binary data.

For a 12 word seed, you take the first hexadecimal character and convert it into a 4 digit binary number (a 24 word seed you would take the first two hexadecimal characters for two 4 digit binary numbers).

In our example, it is the first hexadecimal character is "c" which is 1100 in binary.

The conversion table is below:

Hex	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

We can take those 4 bits (1100) and place them at the end of the leftover 7 bits (1111111) for a total of 11 bits (1111111**1100**). These 11 bits can now be used to determine a number for the last word: 2044.

(A 24 word seed would have 8 bits from two 4 digit binary numbers that would be added to the leftover 3 bits for a total of 11 bits)

Finally, Words!

Now we have:

```
11011010010 = 1746
00010110010 = 178
11110010011 = 1939
10010000111 = 1159
11100001001 = 1801
01000000011 = 515
00001101001 = 105
10010100101 = 1189
11100111110 = 1854
01000010001 = 529
01011000010 = 706
1111111100 = 2044
```

Now using the BIP39 wordlist, we can look up each number and the word in their corresponding position. But remember, since we're representing each 11 bits as a number between 0 and 2047 and the BIP39 wordlist starts at 1 (rather than 0), add +1 to each number to get its positional number.

```
1746+1 = 1747 = surge
178+1 = 179 = bind
1939+1 = 1940 = venue
1159+1 = 1160 = movie
1801+1 = 1802 = thrive
515+1 = 516 = document
105+1 = 106 = artwork
1189+1 = 1190 = net
1854+1 = 1855 = treat
529+1 = 530 = drama
706+1 = 707 = flame
2044+1 = 2045 = zebra
```

We now have our 12 word seed phrase:

```
surge bind venue movie thrive document artwork net treat drama flame zebra
```

Concluding Thoughts

That seems like a pain doesn’t it?

The purpose of this exercise was to show you the nuances of a seed phrase and how it's created. It's more than just choosing random words.

There are many steps required to create a seed phrase securely and requires the use of computer-aided computation.

This is why it's much **easier and safer** to generate your seed on a device that has access to the necessary functions and is already in a secure environment (like a hardware wallet).

So let your wallet generate a seed phrase for you and I hope this exercise helped you learn a little more about seed phrases.