Duck Improvement Proposal and the Efficiency of Entropy for KDF

When using a “Seed Phrase” to create cryptographic keys, whether that be for digital signatures, encryping data, or sending cryptocurrency, it’s extremely important that this seed phrase has enough “entropy” or “randomness” so that nobody else is able to guess it, even with very fast computers, but you are also encouraged to remember this phrase, and for the paranoid, to never write it down. The most common standard in the cryptocurrency scene for the selection of seed phrases that are used for this purposes is BIP39. In BIP39, words are selected from a dictionary containing exactly 2048 words. But wouldn’t it be a lot more difficult to guess a random word out of Websters Unabridged Dictionary, which includes 470 thousand entries, than it would from a dictionary of only 2048 entries? Sure! It would be! However, some words are easier to remember than others, and there are likely to be a lot more mistakes recalling a word from memory when that word could be either “friend” or “friends” or maybe you think it’s “freind” or “fren”, and this is why the BIP39 standard limits the number of words. Additionally, as it turns out, increasing the dictionary size does significantly less for increasing entropy than increasing the number of words you are made to guess/remember from that dictionary. Still, the passphrases you get from BIP39 are quite long, and it would be nice to have something you can easily remember and type out when necessary, without needing to store it anywhere. So can we find a better way?

Firstly, we need to understand how Passphrase Entropy is calculated. You get the passphrase entropy by adding together the entropy *per word*. A Passphrase’s entropy *per word* is found by applying the binary log function($\log_2$) to the number of words in that dictionary. So the total entropy depends on two things, dictionary size, and number of words you take from that dictionary. For BIP39 the entropy per word is:

$S = \log_2 2048 = 11bits$

This means that every word in a BIP39 Passphrase contributes 11bits to the entropy. For a 12 word BIP39 passphrase, the entropy is $11bits \times 12 = 132bits$. In BIP39 specifically, it’s actually not 132bits, but 128bits, because the final word is used as a checksum, and doesn’t contribute 11bits, but this detail isn’t particularly important for us now, and we’re just trying to see the big picture here. A BIP39 passphrase can come in a a few different lengths: 12, 15, 18, 21, and 24 words. Below is an example of a 12 word BIP39 phrase.

```
marine ignore shoulder rack rule assume
monkey escape network inflict visual script
```

Can we get a higher amount of entropy per word, thus enabling us to have a shorter phrase that’s easier to type or tap out? We can do this by increasing the dictionary size. How big of a dictionary do we need? The amount of entropy per word is added to get the final entropy of the phrase, so to reduce

the size of a passphrase by half of BIP39, we need to get double the entropy of BIP39 per word. We need 22 bits of entropy per word.

Dictionary Size	Entropy Bits Per Word	Similar Size Found In
2,048	11	BIP39
4,096	12	
8,192	13	EFF Password Generator
16,384	14	Adult Vocabulary
65,536	16	
262,144	18	An Unabridged English Dictionary
1,048,576	20	
4,194,304	22	
16,777,216	24	

What does this tell us? To meet our goal of having 22bits of entropy per word, we need to have over 4million words in our dictionary! That's impossible right? Sure! Maybe? Maybe not? What if they are not dictionary words, but junk words, and follow simple rules for construction based on the kinds of syllables (or graphemes) that are available in English. This way, we would still be able to pronounce them, or mentally pronounce them, so they would be easier to remember than just disordered letters. How big would a dictionary be that is composed of words of all the possible combinations of easy to remember English syllables? This depends on how many syllables there are in English, and how many syllables we allow per word.

So, how many syllables are there in English? It isn't completely trivial to work out how many syllables are in English, but we dont need a perfect answer, and we dont need to answer a linguists questions about what a word or syllable *really is*, we just need some rules for generating words, and an idea of how many syllables we can make using this process. To do this, we can alternate consonants and vowels to make words, and include also consonant blends such as "sh" "ch" and dipthongs "au" "ow". Lets say there are 20 consonants in English, 8 consonant blends, 28 in total. Likewise there are 5 vowels and 5 dipthongs, 10 in total. We can alternate consonants and vowels to form syllables. This way, how many syllables are there in English?

$$number\_of\_english\_syllables = (20 + 8) \times (5 + 5) = 280$$

Great! So how many words would there be in a dictionary where words are made up from these syllables strung together and were restricted to a certain number of syllables per word?

Syllables per Word	Number of Words
1	$280^1 = 280$
2	$280^2 = 78,400$
3	$280^3 = 21,952,000$
4	$280^3 = 6,146,560,000$

There are more than 6 Billion 4 syllable words! That's a lot more than the 4million vocabulary we need, and words that long could look very confusing, so let's use the three syllable words instead. If we need more entropy than the three syllable words provide we can use more words in the passphrase, rather than longer words. In the end, all of our words are garbage anyway, so the distinction between longer words and more words is really only a matter of how we chunk groups of syllables together. One 3 syllable word will give us $log_2(21,952,000) = 24bits$ of entropy. We only need 22, but 24 is even better, so we'll take that. Although the two syllable words are too short to give us the entropy we need *by themselves*, we could include them *in addition* to the three syllable words. If we were to generate words this way, here is an example of what a six word phrase looks like:

```
lobari denacha bovano lypera cinato kideme
```

And the entropy would be

$S = \log_2(21,952,000^{6words}) = 146.3bits$

Recall that a 12word BIP39 passphrase has 128bits of entropy, but our passphrase has 146bits, but its half as long! Would you rather remember this:

> lobari denacha bovano lypera cinato kideme

Or this:

> marine ignore shoulder rack rule assume monkey escape network inflict visual script

This *might* be a matter of preference, as dictionary words are easier to remember. But are they *twice* as easy to remember as junk words? This is an open question, and the shorter phrase is probably much faster to type, and especially faster to *tap* on a mobile device. If you want to never have to write your passphrase down, or you worry about your passphrase seed being wiped from disk or the physical copy lost or stolen, then having a high entropy passphrase that you remember enables you to do this.

Security Comparison

Lets see this side by side. How does the security of DIP39 stack up next to BIP39 ?

Standard	Words	Entropy/Security	Use Case
BIP39	12	128	Deriving Keys, Not Quantum Secure
BIP39	15	160	As Above
BIP39	18	192	As Above
BIP39	21	224	As Above
BIP39	24	256	Deriving Keys, Quantum Secure
-	-	-	
DIP39	2	48	Pin Unlock
DIP39	4	97	Pin Unlock
DIP39	6	146	Deriving Keys, Not Quantum Secure
DIP39	8	195	As Above
DIP39	10	243	As Above
DIP39	12	292	Deriving Keys, Quantum Secure

Examples

Lowest BIP39 Standard (Still a lot of entropy!)

BIP39 128bit

quiz include march retreat wink key dog dismiss answer solution prefer arrange vacant clown tumble

DIP39 146bit

lobari denacha bovano lypera cinato kideme

Highest BIP39 Standard

BIP39 256bit

quiz include march retreat wink key dog dismiss answer solution prefer arrange vacant clown tumble female turn patch limit account giggle sugar clip foil

DIP39 292bit

lobari denacha bovano lypera cinato kideme velada morani teravo chonala qunaba tolabau

2 or 4 Word DIP39

I included an example of a 2 word DIP39 phrase, of course, this is far too low for something like deriving large numbers of keys, or to use in globally unique way, however, there is a usecase here for unlocking devices with cryptographic keys derived from a passphrase, even with very low entropy keys. This is usually done with a pin number, or a password, with *software security* ensuring the device is unlocked but this action could also be done with *cryptographic security* with a cryptographic key derived from a very short DIP39 passphrase. Any password could perform this role, DIP39 just describes one way to choose such an unlock passphrase.

Summary

This proposal aims to make these seeds easier to remember, and to type, due to their smaller size, but it also facilitates the use of the seeds more directly, so that you could use them to unlock a mobile application, rather than relying on a much weaker pin. *I announce to you DIP39, Duck Improvement Proposal 39.*