BIP39 Seed Phrase Obfuscation (BSPO)

Michael Mezher (MCS)
Nabil Mezher (MCS)

Abstract

The ability to self-custody Bitcoin is arguably one of its most important and distinguishing features. However, participating in self-custody of Bitcoin exposes asset holders to a number of risks, namely the potential for loss and theft. This paper introduces BSPO, a simple method aimed at improving the security of the BIP39 protocol. We discuss the underlying cryptographic security of the BIP39 protocol along with the real-world vulnerabilities of storing seed phrases using currently recommended methods. We postulate that applying BSPO dramatically improves the security of the underlying assets tied to a seed phrase and support our claim through rudimentary parametric statistical analysis.

Introduction

BIP39 describes a widely used method of generating a mnemonic code, frequently referred to as a seed phrase (Palatinus et al., 2022). This phrase can then be converted to a binary seed, which is used in BIP32 to produce a Hierarchical Deterministic Wallet (Wuille, 2022) in order to enable the owner of the wallet reception of Bitcoin (or other cryptocurrency – herein referred to as Bitcoin) via derivative public addresses. Although BIP39 is cryptographically secure, the use of a seed phrase as the primary method of storing Bitcoin introduces several real world security vulnerabilities. BIP39 Seed Phrase Obfuscation (BSPO) is a method aimed at resolving these issues while remaining interpretable and usable by cryptocurrency holders without a technical background.

Cryptographic Security

BIP39 seed phrases are generated by a process starting with generating a random binary seed B of between 128-256 bits with a length N divisible by 32. Next SHA256 is used to hash B, and the first $N/32$ bits of this hash are used as a checksum C. This checksum is then appended to the end of B, whose summation we will call R. Finally, the bits are segmented into groups of 11 bits encoding an integer $b \in [1, 2048]$. This integer is used as an index into the BIP39 word list of length 2048.

Although the aforementioned checksum reduces the number of valid seed phrases that can be generated (the checksum typically being specific to an underlying wallet protocol), since there is no known way to identify the subset of valid seed phrases a priori, an attacker looking to identify an existing seed phrase via brute force would still need to consider all potential seed phrase permutations. This is obvious once one considers that SHA256 remains cryptographically secure.

It is clear that BIP39 is a cryptographically secure protocol. To illustrate this, consider the following: If attempting to identify a seed phrase of a specific existing wallet in a brute forcing manner, without any additional information, it would take $A = (2048^{(N+(N/32))/11})/2$ attempts for a 50% probability of generating a

collision, defined as when a seed phrase generated matches *any* previously generated seed phrase. For N=128 bits (a 12-word seed phrase, the least cryptographically secure), $A = 2.723 \times 10^{39}$. For N=256 bits (a 24-word seed phrase, the most secure) $A = 1.482 \times 10^{79}$. For perspective, this is roughly equal to the number of atoms in the observable universe. This means that *your* wallet is highly secure. Of course, an attacker is likely more interested in generating a seed collision with *any* wallet.

If 12 word seed phrase are generated at random, in order for the probability of collision to exceed 50%, approximately 8.688*10^{19} seed phrases would have to be in use. This result, obtained from equations 1, is based on the Generalized Birthday Problem formulation referenced in (Brink, 2012) Theorem 2. Note that this does not take into account wallet specific checksums.

$$\frac{3-(2*ln(2))}{6} < n(d) - \sqrt{2d * ln2} \leq 9 - \sqrt{86 * ln2} \quad (1)$$

Since the number of seed phrases likely ever to be used is many orders of magnitude smaller than this value, an unintentional or intentional (birthday attack) collision highly improbable, even when using the least cryptographically seed phrase variation of length 12.

How improbable? Let us make the following assumptions.

- There are currently 8x10^9 12 word seed phrases in use, roughly one for each person on earth.
- The only compute intensive operation an attacker needs to perform to generate and test seed phrases for collision is SHA-256. This assumption is highly beneficial for the hypothetical attacker.
- The attacker is able to perform SHA-256 at a rate of 100 TH/s (roughly the hashing capacity of the AntMiner S19, the latest major model refresh from BITMAIN, the leading producer of SHA-256 ASIC hardware).

Seed phrases are uniformly distributed and the attacker generates seeds in a deterministic and sequential manner. This is beneficial over randomly generating the seed phrases, as the probability of collision in the former case would be appropriately modeled by a hypergeometric distribution whereas in the later case, it would be modeled by a binomial distribution.

Under these conditions, it would take the attacker just under 15 million years to achieve a 50% probability of generating a single collision. This follows from the hypergeometric distribution PMF shown in equation 2 below.

$$\sum_{k=1}^{n} \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}} = 1 - \frac{\binom{K}{0}\binom{N-K}{n-0}}{\binom{N}{n}} \quad (2)$$

Where K is the number of existing wallets, k denotes the number of collisions, N is the size of the possible seed phrase output space, and n is the number of seed attempts made (ie the hashrate multiplied by the duration of the attack in seconds). The notation $\binom{a}{b}$ represents the binomial coefficient of variables *a* and *b*. For a more intuitive rough order of magnitude estimate of this collision probability, consider that if 12 word seed phrases are generated uniformly in their possible output space, the size of each partition created between 8 billion existing seed phrases would be $\frac{N}{(K+1)}$ where N represents the size of the seed phrase output space and K represents the number of existing wallets. This value comes out to 6.81e+29 in the previously mentioned scenario. Continuing our previous assumption of a 100 TH/s ASIC, starting our generation of seed phrases from one end of the output space, it would take approximately $\frac{6.81e+29}{1e+16 * (86400*365)} = 2.16e + 7$ years to encounter the first existing seed phrase.

Note that this assessment is aimed only at assessing vulnerabilities to conventional computers. Quantum based birthday attacks may be feasible, particularly to the 12 word seed phrase variation. However, such hardware is, as of 2022, still theoretical.

BIP39 Vulnerabilities

Although the method of generating seed phrases outlined in BIP39 is secure from a cryptographic perspective, the resulting phrase can easily be misplaced or stolen, resulting in a permanent loss of Bitcoin. It's estimated that over 1500 Bitcoin are lost daily (Cane Island, 2020), in most cases due to the mishandling of private keys. Naive mitigation of these vulnerabilities through seed phrase replication and/or distributed storage results in additional threat exposure.

Consider the following primitive methods of BIP39 seed phrase storage and their obvious vulnerabilities.

- Storage in a home safe.
  - Clear target during theft.
- Storage in a single hidden location.
  - Increases the probability of lost seed phrase, potential for unintentional discovery by third parties.
- Storage of seed phrase replicas in multiple locations.
  - Although the risk of phrase loss is decreased, the risk of successful theft and/or unintentional discovery is increased.
- Storage of partial seed phrase segments in multiple locations.
  - Although the risk of theft by an actor of low sophistication is decreased, a sophisticated attacker may be able to use the partial information to brute force the remaining seed terms. Additionally, depending on how the seed is distributed, risk of loss may also be increased. Note that this method does not refer to more intricate methods of distribution such as Shamir's Secret Sharing.
- Mental storage.
  - Risk of forgetting.
- Digital storage.
  - Seed phrases are highly susceptible to crawling applications and should never be stored digitally. In general, seed phrases should never be entered in a digital format unless the user is looking to recover their funds.

There's currently no easily implemented method of seed phrase storage that mitigates accidental loss and theft, arguably the two primary inherent risks of self-custody. BSPO aims to provide this.

Existing Work

The creators of BIP39 make some clear attempts at designing a user friendly protocol. First, rather than using an integer between 0-2047, these values are used to index a known list of commonly used terms, which may be easier to remember than a sequence of random integers. Additionally, risk of misidentification is reduced by the BIP39 requirement that the first 4 characters of each word be unique, regardless of word length.

There also exists auxiliary development to the BIP39 original implementation aimed at improving its security. Notably, seed passphrases are supported by several widely used hardware wallets including Trezor, Ledger and Coldcard. These passphrases are an additional set of characters that must be used in conjunction with the original seed to facilitate derivation of the underlying deterministic hierarchical wallet (Trezor, 2022). Though this work certainly improves upon the security of the BIP39 seed phrase, particularly with respect to physical theft by an unsophisticated attacker, several shortcomings exist. Since the passphrase can be

flexibly generated (in the Trezor implementation), it is not guaranteed to be cryptographically secure and is susceptible to brute force or dictionary attacks. In another scenario, if the user attempts to memorize the passphrase, they may lose their funds due to forgetting the phrase. This method does not resolve any issues associated with losing a seed phrase, unless the seed phrase is more freely distributed to additional locations, in which case, security may be compromised due to the previously mentioned potential vulnerabilities to the passphrase itself. Other methods of improving the practical security of BIP39 exist (such as use of Shamir's Secret Sharing algorithm (Streit, 2021)), but tend to be difficult to interpret and leverage by laypersons and generally require the digital entry of seed phrases.

Proposed Solution

BSPO is implemented in a manner that's both easy to understand and to leverage. At a high level, BSPO converts a user's original seed phrase to an encrypted seed phrase, which can be reversed using a simple lookup table. The initial conversion of the seed phrase into its encrypted form is also done using a lookup table. This enables a user to encrypt/decrypt their seed phrase without ever having to digitally enter their seed phrase or perform any sort of calculations. In addition, the risks inherent to self-custody (loss/theft) discussed earlier may both be mitigated using BSPO.

Usage of BSPO is conducted as follows:

1. A user with a written seed phrase P navigates to bipshuffle.com.
    - A table of 2048 x N+1 is created, where N is the length of the user-determined seed phrase. The first column contains an alphabetically sorted version of the BIP39 term list. Each subsequent column contains a shuffled version of the BIP39 term list (shuffled in a cryptographically secure manner – our implementation uses an implementation of Fisher Yates with a Linux OS CSPRNG as the source of entropy). Call this table *T1*.
    - A corresponding table *T2* to *T1* is concurrently generated with dimensions being 2048 x N+1. Its first column is an alphabetically sorted version of the BIP39 term list. However, each subsequent column contains the reverse mapping between the alphabetically sorted first column and the columns in *T1*. That is, if term $a \rightarrow b_i$ in *T1,* then in *T2* $b_i \rightarrow a$; where a represents the BIP39 term in the first column of *T1* and b_i represents the BIP39 term in the ith column of *T1*. "Mapping" in this context is always considered to be matching a term in the first (alphabetically sorted) column of a table to a term in another index specified column.
2. Using a pen and paper, the user then maps their original seed phrase to its encrypted form using *T1*. To do this, they simply find each term t_x for x in [1,N] in $T1_{:,1}$, (where t_x represents the term at index x in P and $T1_{:,1}$ represents the first column of *T1*) and find the corresponding term $T1_{k,\,x+1}$ (where $T1_{k,\,x+1}$ represents the term in *T1* at the k^{th} row and the $x+1^{\text{th}}$ column; the k^{th} row being the row containing t_x in $T1_{:,1}$) – call this corresponding term $t1_x$. The user then writes each encrypted term $t1_x$ in order on the sheet of paper. Call this encrypted phrase P_e.
3. Next, the user digitally saves or prints out table *T2*.

The user then verifies that they are able to decrypt P_e into P using table *T2*. The process for this is identical to what is outlined in step 2 above, but P_e is used in place of P and *T2* is used in place of *T1*.

5. Optional (suggested): The user then repeats the entire process (at least once) such that they have two encrypted keys and two decryption tables (*T2*). The encrypted key and decryption table should be marked such that the user knows that they correspond to one another. This reasoning for this (suggested) repetition is outlined in the *Approximation of Risk Reduction* section of this paper.
6. Once the user has followed the above steps and confirmed the ability to successfully decrypt P_e into *P* using *T2* (for all of their created pairs of P_e and *T2*) , they can safely discard *P*.
7. Finally, it's suggested that the user then stores their encrypted seed phrase(s) and decryption tables in separate, known locations.

   The resulting pairs of P_e and *T2* contain all the information required to reconstruct *P*, but neither P_e or *T2* contain any information useful for reconstructing *P* by themselves. To be more specific, say that the user created *n* pairs of encrypted seed phrases and decryption tables denoted $(P_e^1, T2^1)$, $(P_e^2, T2^2)$, ... $(P_e^n, T2^n)$. Then gaining access to any encrypted seed P_e^x would provide absolutely no information useful for reconstructing *P* unless access to $T2^x$ was also acquired. Similarly, gaining access to any decryption table $T2^y$ would provide absolutely no information useful for reconstructing *P* unless access to P_e^y was also acquired. This property not only dramatically improves security by requiring an attacker to locate two separate pieces of information, it also enables relatively safe storage of either $T2^i$ or P_e^i in a digital format (due to the size of *T2*, this is the piece of information we recommend for digital storage). Additionally, because BSPO can be repeated as many times as desired, the potential for seed phrase loss is dramatically reduced, despite two pieces of information being required to reconstruct *P*.

Pragmatically, the end result is nearly identical to applying Shamir's Secret Sharing to the *P* with a two share split and requiring both shares for reconstruction of *P*. However, our method never requires a user to digitally enter their seed phrase, and requires no complex operations to be conducted by the user. The method remains scalable in the same manner as Shamir's Secret Sharing strategy, and *P* can be apportioned to *up to N*-1 unique entities.

Although the process outlined above may seem complex (in part, due to the notation used); in practice, it's a very straightforward practice to carry out and is easily illustrated through a guided user interface.

Approximation of Risk Reduction

Although, it's impossible to concretely quantify the risk reduction provided by BSPO, this section aims to provide a practical parametric approximation by making the assumption that adverse events can be categorized as either loss or theft and occur with some assumed probability in an IID manner.

Let P_L represent the probability that loss of a seed phrase or decryption table occurs over some finite duration *D*. Let P_T represent the probability that theft of a seed phrase or decryption table occurs over the same duration *D*. Then the probability that each seed phrase or decryption table is not lost or stolen (ie safe) is $P_S=1-(P_L+P_T)$.

Note that loss and theft are modeled separately, because in the case of "loss", losing either the seed phrase or decryption table from a pair causes loss of the underlying asset; whereas in the case of theft, both the seed phrase and decryption table must be compromised. In this approximation, it can thus be said that cases of theft where the seed phrase or decryption table is physically stolen and lost to the user should be lumped in with P_L.

Generalizing over an arbitrary number of BSPO repetitions, for funds to be compromised due to loss, *one* or more pieces of information from *every* seed/decryption table pair must be lost; For funds to be compromised due to theft, *both* pieces of information from *one* seed/decryption table pair must be stolen.

Consider this simplified model applied to several scenarios as outlined below:

1. A seed phrase stored on a physical sheet of paper without BSPO applied.
   - In this scenario, the probability that the seed phrase remains safe over some finite duration is simply $P_S = 1 - (P_L + P_T)$.
2. One round of BSPO is applied and the seed phrase and decryption table are stored separately (note that if stored together, the scenario essentially reduces to scenario 1 above).
   - In this scenario, the probability that the seed phrase remains safe is $P_S = 1 - ((1 - \overline{P}_L^2) + P_T^2$.
   - It can be said that in this scenario, the term $(1 - \overline{P}_L^2)$ represents the probability that the funds are compromised due to loss and P_T^2 represents the probability that funds are compromised due to theft.
3. Two rounds of BSPO are applied and all four pieces of information are stored separately.
   - In this scenario, the probability that the seed phrase remains safe is $P_S = 1 - ((1 - \overline{P}_L^2)^2 + (1 - (1 - P_T^2)^2))$
   - Once again breaking apart this scenario, the term $(1 - \overline{P}_L^2)^2$ represents the probability that the funds are compromised due to loss (two separate events where either a seed or decryption table is lost from each created pair must occur.) and $1 - (1 - P_T^2)^2$ represents the probability that the funds are compromised due to theft (either of the pairs of seed phrases and encryption tables must be stolen).

This approach to modeling adverse events and the corresponding probability that the funds remain safe can be applied to any number of BSPO rounds. The general formula for calculating the probability that the funds remain safe as a function of the number of BSPO rounds is shown in equation 3 below.

$$P_S = 1 - ((1 - \overline{P}_L^2)^R + (1 - (1 - P_T^2)^R)) \quad (3)$$

The variable R represents the number of times BSPO is leveraged. Figure 1 below illustrates the approximation of a fund safety subject to different hypothetical values of P_L and P_T.

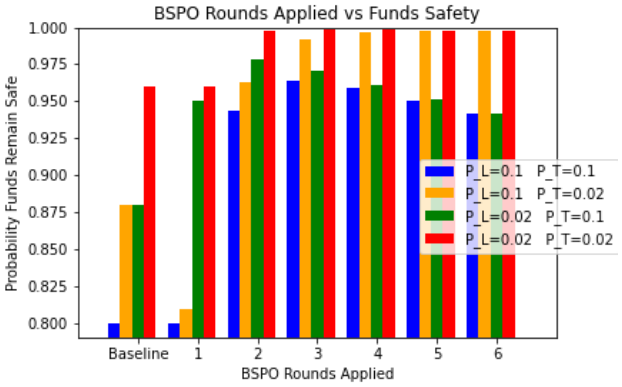

Figure 1

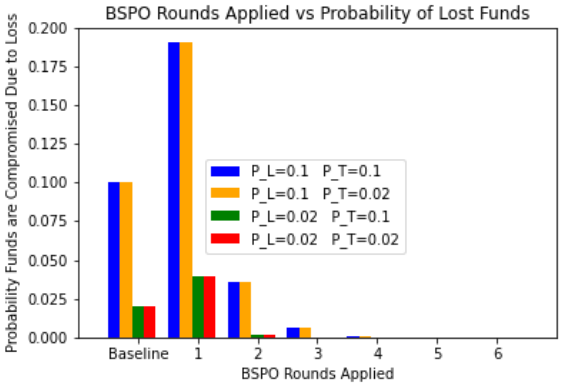

Figure 2

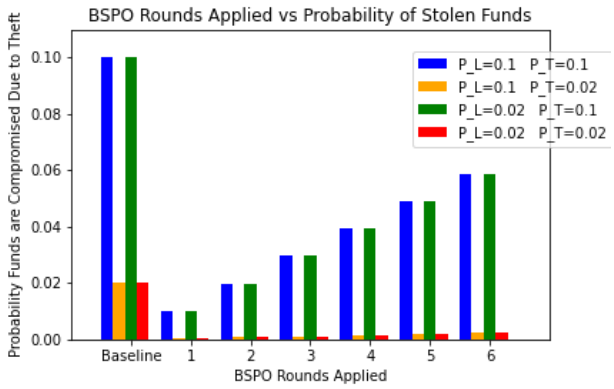

Figure 3

Notice that in all cases, fund safety dramatically improves after 2 rounds of BSPO. Note that the only instance where BSPO causes *decreased* safety is when a single round is applied while assuming $P_T < P_L$. This case emphasizes the importance of applying BSPO at least twice.

As expected, for all cases, the probability of loss occurring decreases relative to the number of BSPO repetitions and the probability of theft occurring increases (shown in Figures 2 and 3 respectively). However, since successful theft requires access to two separate pieces of information, the decreased loss probability more than makes up for the increased probability of theft.

Note that this approximation is only provided as a thought experiment, and that real world complexities make it nearly impossible to accurately quantify fund safety except through retrospective study. Still, this model demonstrates the notion that BSPO (especially applied at least twice) is likely to dramatically reduce the chance of property loss.

Shortcomings

Although the BSPO method is secure and attempts to minimize obstacles for users, imperfections still exist. Firstly, the user must still be able to accurately map terms using a lookup table. We've tried to make our website (bipshuffle.com) convenient to use for this purpose by fixing row and column indices and providing a scrollable table, but this only helps with encrypting the seed phrase. During decryption, the user must tediously use their printed or digitally saved table to recover the original phrase.

In the hopes of decreasing user error, we provide a step-by-step user guide for encryption and decryption of seed phrases using BSPO. The seed phrases used in this step-by-step user guide are randomly generated, such that the user can run through the guide multiple times to gain an intuition for BSPO.

One less obvious vulnerability is the potential for bad actor's collect information directly from a user's screen (assuming a user's machine is compromised). If the user is leveraging their cursor as a tool to assist in seed phrase encryption/decryption, it's possible an attacker could identify the seed phrase. To mitigate such an attack, we've provided the ability to download and print both the encryption and decryption tables, providing the ability to conduct BSPO in a completely offline setting.

Addendum

The analysis above is directly applicable to the "full" version of BSPO. Since the origination of this document, a "lite" version of BSPO has been added to the bipshuffle.com website. BSPO Lite was added in the interest of usability. The primary tradeoff is improved simplicity and convenience for a decrease in cryptographic security.

BSPO Lite and BSPO Full differ in the number of shuffles used to generate the final, secured cipher seed. BSPO Full provides a new term selected from a uniquely shuffled enumeration for every index of the seed phrase, whereas BSPO Lite only produces a single shuffled enumeration. This means that in the event there exists a repeat term in the original seed, with respect to BSPO Full, the probability the same term indices in the encrypted seed will also be repeat terms is 1/2048; whereas in BSPO Lite, the newly encrypted terms are guaranteed to be repeat terms at the same indices as the original seed.

Therefore, the encrypted seed resulting from BSPO Lite reveals *some* information about the original seed. Assuming an attacker was in possession of the encrypted seed and knew the seed had been encrypted using BSPO Lite, how much information about the original seed would be revealed?

Consider the following cases applied to a 24 word BIP39 seed phrase (N=2048 and r=24):

1. The original seed has 0 repeat terms
   - The information revealed is simply that the original seed also must not have any repeating terms. The number of possible term combinations (i.e. the space the attacker must search) is thus equal to the number of existing permutations *without* replacement.

$$P_{E0} = \frac{N!}{(N-r)!} = \prod_{k=0}^{r-1} N - k \qquad (4)$$

   Consider that the original seed has an output space equal to the number of existing permutations *with* replacement.

$$P_O = N^r \qquad (5)$$

   Note that $P_O > P_{E0}$, however P_{E0} is *still* very secure. Specifically, for a 24 word seed phrase, $P_O \approx 2.964e + 79$ whereas $P_{E0} \approx 2.589e + 79$. Hardly a meaningful reduction in the possible output space.

2. The original seed has 1 set of repeat terms (2 repeat terms).
   - In this case, the BSPO Lite encrypted seed not only reveals that there exists a set of repeating terms in the original seed, but the indices of where that repeating set occurs. Once again, the output space an attacker must target must be calculated. The attacker knows that there is a set of repeating terms, and that no other terms repeat. The attacker also knows the location of the repeating terms. In effect, the attacker can reduce the encrypted seed into a seed of length N-1. The output space can now be calculated as the set of permutations without replacement (ie equation) while substituting N for N-1. The size of the resulting output space is $P_{E2} \approx 1.279e + 76$. For comparison, following from equation 5, the size of the output space of a 23 word seed phrase would be $P \approx 1.447e + 76$, and a 22 word seed phrase would be $P \approx 7.067e + 72$. Thus, since even 12 word seed phrases are considered cryptographically secure, revealing that a set of repeat terms exists in the original

seed phrase doesn't meaningfully deteriorate the cryptographic security.

3. The original seed has 3 repeat terms.
   - Following the same logic above, as the number of repeating terms increases, the size of the search space the attacker must target decreases. For 3 repeating terms, following from equation 4 (replacing), $P_{E3} \approx 6.311e + 72$.

     We find that for each repeating term, the amount of output space size decrease is approximately the same as reducing the length of the seed phrase by one term. This analysis can be applied for multiple *pairs* of repeating terms.

We can estimate the probability of occurrence of each scenario above (and all possible scenarios) based on the generalized birthday problem. In particular, we're interested in calculating the portion of randomly generated seed phrases with n unique terms, for $n \in [1, 24]$. Because no closed form solution for this problem is known, the results of Monte Carlo Simulation (10^7 trials) are provided in Figures 4 and 5 below.

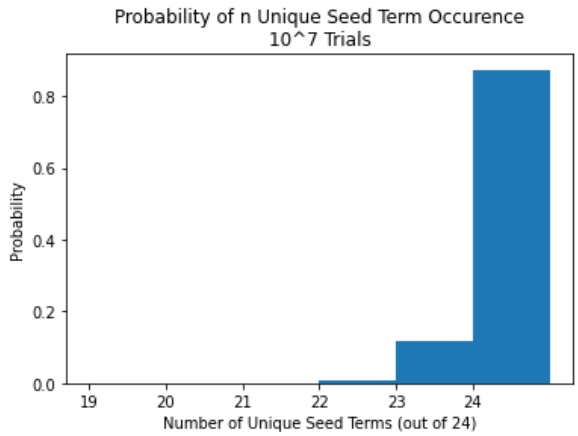

Figure 4

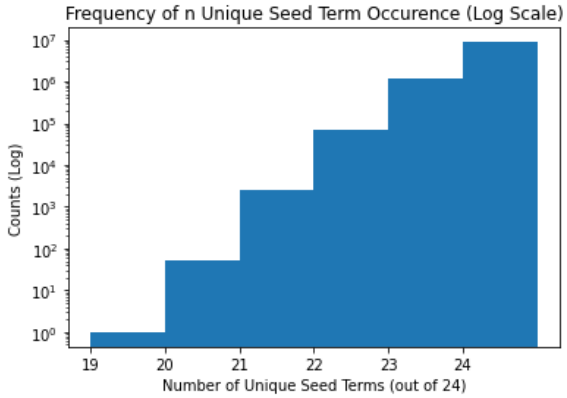

Figure 5

Notice that the probability of each additional non-unique term decreases dramatically. In 10^7 trials, only a single instance of 19 unique (5 colliding terms) occurred. In the worst case (highly improbable), all 5 colliding terms were the same, in which case the BSPO Lite encrypted seed still retains the equivalent security of an approximately 19 word seed phrase. Far more likely, this instance of 19 unique terms represents a collision of a set of 2 terms and another set of 3 terms, thus providing the approximate equivalent security of a 21 word seed phrase. In any case, it's clear that for the vast majority of 24 word seed phrases, BSPO Lite represents a safe way to conveniently encrypt a seed. However, for users with a 12 word seed phrase, we continue to recommend leveraging BSPO Full due to the decreased tolerance for reduced cryptographic security. For users with a 24 word seed with more than 11 non-unique terms (statistically highly improbable), it is recommended to leverage BSPO Full.

References

M. Palatinus, P. Rusnak, Aaron Voisine, and Sean Bowe, “Mnemonic code for generating deterministic keys,” *GitHub*, 25-Jul-2022. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki. [Accessed: 04-Oct-2022].

P. Wuille, “Hierarchical Deterministic Wallets,” *GitHub*, 03-Jan-2022. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki. [Accessed: 04-Oct-2022].

D. Brink, “A (probably) exact solution to the birthday problem,” *The Ramanujan Journal*, vol. 28, no. 2, pp. 223–238, 2012.

Cane Island, *There Will Never Be More Than 14 Million Bitcoins*, 16-Apr-2020. [Online]. Available: https://static1.squarespace.com/static/5d580747908cdc0001e6792d/t/5e98dde5558a587a09fac0cc/1587076583519/research+note+4.17.pdf. [Accessed: 04-Oct-2022].

D. Streit, “EIP-3450: Standardized shamir secret sharing scheme for BIP-39 mnemonics,” *Ethereum Improvement Proposals*, 29-Mar-2021. [Online]. Available: https://eips.ethereum.org/EIPS/eip-3450. [Accessed: 04-Oct-2022].

Trezor, “Passphrase,” *Trezor Wiki*, 2022. [Online]. Available: https://wiki.trezor.io/Passphrase. [Accessed: 04-Oct-2022].