SeedSigner Enhancement - SeedXOR

Summer of Bitcoin | SEED SIGNER

About Me

Hi, this is Om Amar,

I am a Second Year Computer Science Engineering Student at IIIT Nagpur and currently also pursuing B.S. in Data Science and Applications from IIT Madras. I am quite passionate about Python and have explored quite a few domains such as Web Development, Machine Learning, Competitive Programming using Python and reached Candidate Master on Codeforces as well as 6⭐ on Codechef. I have been recently exploring Blockchain and am quite intrigued by it and want to go more deep into this domain as well.

I have worked with bitcoinrpc, and a few hashing related libraries like hashlib, binascii, ecdsa, base58 and other python libraries/frameworks.

This project really piqued my interest, especially given my background in Competitive Programming. The idea of using XOR operations to create and recover keys is both fascinating and innovative. It reminds me of how similar techniques are applied in competitive problem-solving, where efficient and clever manipulation of data often leads to elegant solutions.

Contact Information

Discord : Om_Amar
Time Zone : India Standard Time (GMT+5:30)
Email Address : emailomamar@gmail.com
University Name : Indian Institute of Information Technology,Nagpur
Country : India
Github : OmAmar106
Linkedin : om-amar

Education

1. University : IIIT Nagpur
   Branch : Computer Science and Engineering
   Degree Level : 2nd Year UG
   Graduation Year : 2027

2. University : IIT Madras
   Branch : Data Science and Applications
   Degree Level : Diploma
   Graduation Year : 2027

Synopsis

This project enhances SeedSigner by implementing a SeedXOR feature that allows users to generate as many independent seed phrases as required by the user, whose XOR yields the original BIP-39 seed. The approach involves enabling both the creation of new seed shares and the regeneration of alternative sets for the same seed, allowing recovery even if one share is lost — leveraging the fundamental reversibility of XOR for secure, fault-tolerant key management.

Project Plan

The fundamental principle behind the project is the reversibility of the XOR operation, which allows for both the creation and recovery of seed phrases from multiple shares. This approach enhances both security and efficiency, as no single phrase can authorize access when multiple shares are required. Furthermore, even if one set of shares is lost, the seed can still be recovered using other independently generated sets by the property of the XOR operation.

1. Creating New Phrases

   The first phase of the project involves implementing functionality to generate new mnemonic phrases (shares) such that their cumulative XOR equals the original seed. This allows users to securely split their BIP-39 seed into multiple parts, where no single part is useful on its own, but all parts can be recombined to recover the original seed.

The implementation will follow these steps and has been depicted in Fig.1 as well:

1. Original Seed: We start with the original seed, denoted as a_{seed}.
2. Generate New Seed: A new random seed, denoted as b_{seed}, is created.
3. Calculate XOR: We then compute a new seed $c_{\text{seed}} = a_{\text{seed}} \oplus b_{\text{seed}}$, where $\oplus$ represents the XOR operation.
4. Convert to Mnemonic: Both b_{seed} and c_{seed} are converted to mnemonic phrases using BIP-39 encoding. These mnemonics will be stored as the two independent keys (b_{phrase} and c_{phrase}).
5. Recovery Process: To recover the original a_{seed}, the user will need to convert b_{phrase} and c_{phrase} back to their corresponding seeds using BIP-39. Then, the user simply XORs these two seeds together: $b_{\text{seed}} \oplus c_{\text{seed}} = a_{\text{seed}}$, which will give them the original seed for authorization and in case of multiple such phrases, they will need to calculate the XOR of all of their seeds as shown in Fig.2 .

This process can be repeated recursively: just as we created b_{phrase} and c_{phrase} from a_{seed}, we can do the same for b_{seed}, generating another set of phrases. This approach allows for an unlimited number of phrases, all of whose cumulative XOR of their BIP-39 values will yield the original key. This recursive process ensures flexibility and security, and they can create more than one set of pairs of phrases which can be used to get the original seed.

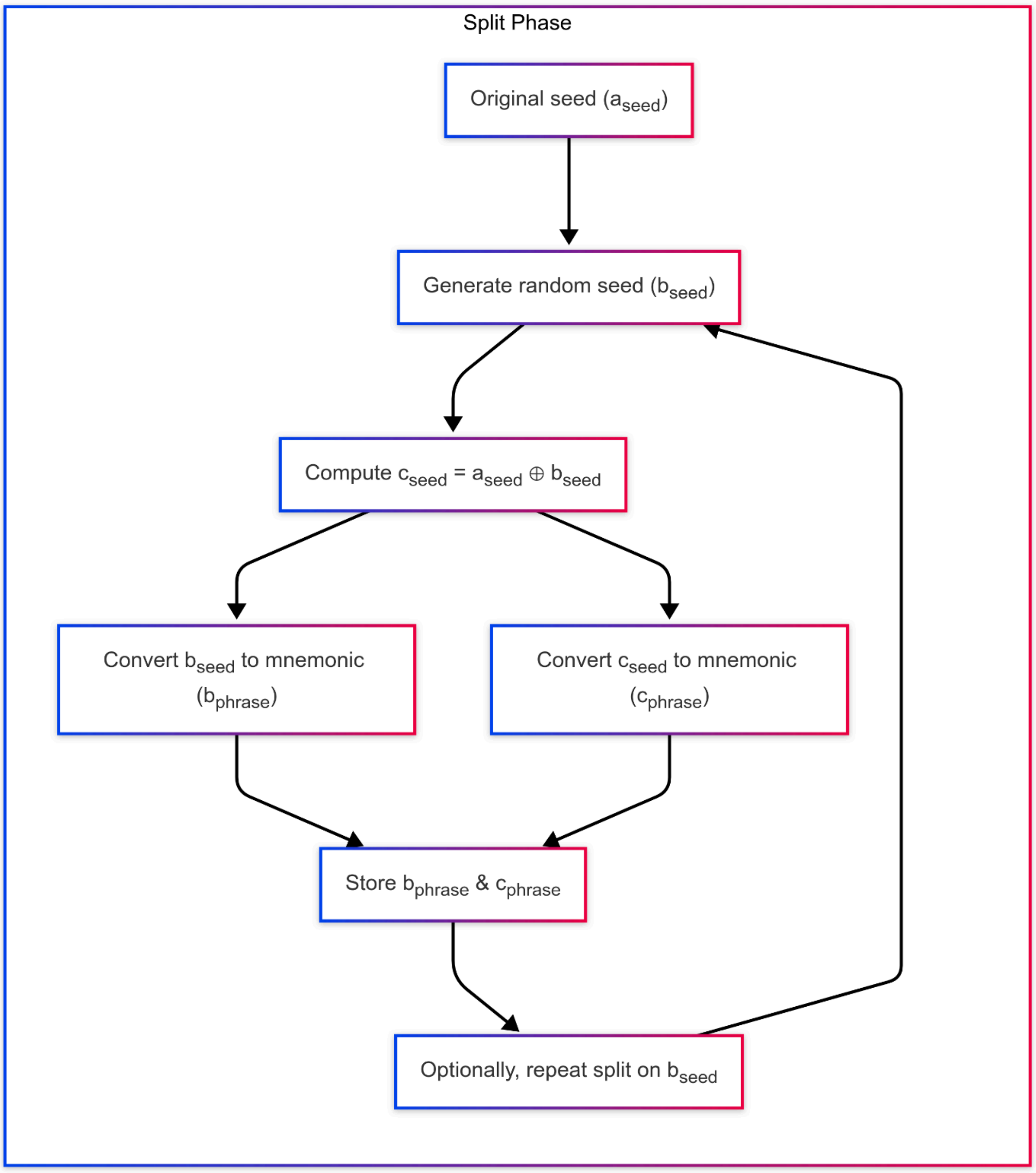

Split Phase
Original seed (a_{seed})
Generate random seed (b_{seed})
Compute $c_{seed} = a_{seed} \oplus b_{seed}$
Convert b_{seed} to mnemonic (b_{phrase})
Convert c_{seed} to mnemonic (c_{phrase})
Store b_{phrase} & c_{phrase}
Optionally, repeat split on b_{seed}

Fig 1. Generating Phrases from the seed

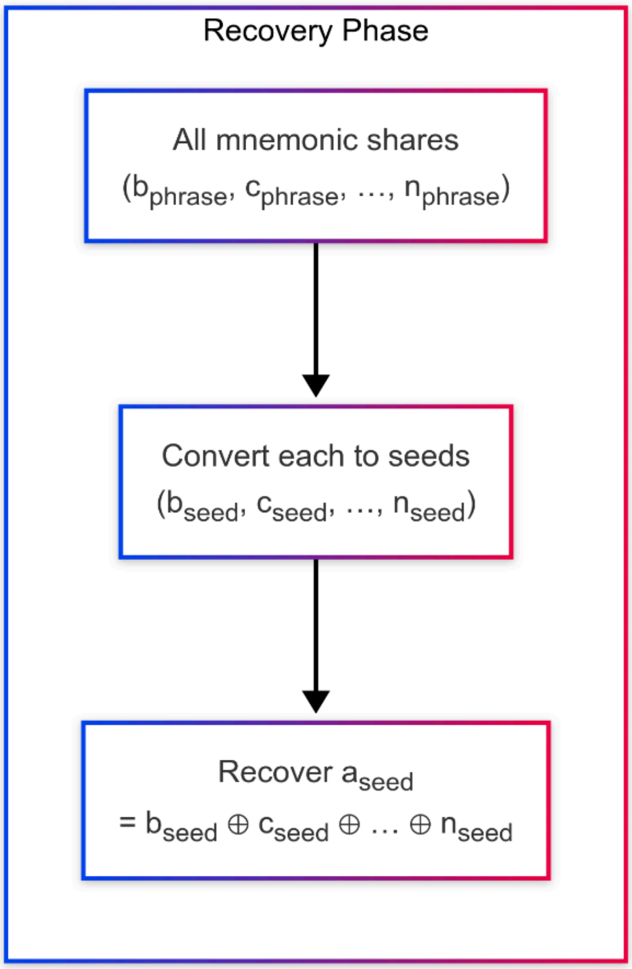

Recovery Phase
All mnemonic shares
$(b_{phrase}, c_{phrase}, \ldots, n_{phrase})$
Convert each to seeds
$(b_{seed}, c_{seed}, \ldots, n_{seed})$
Recover a_{seed}
$= b_{seed} \oplus c_{seed} \oplus \ldots \oplus n_{seed}$

Fig 2. Recovering the Seeds from the Phrases

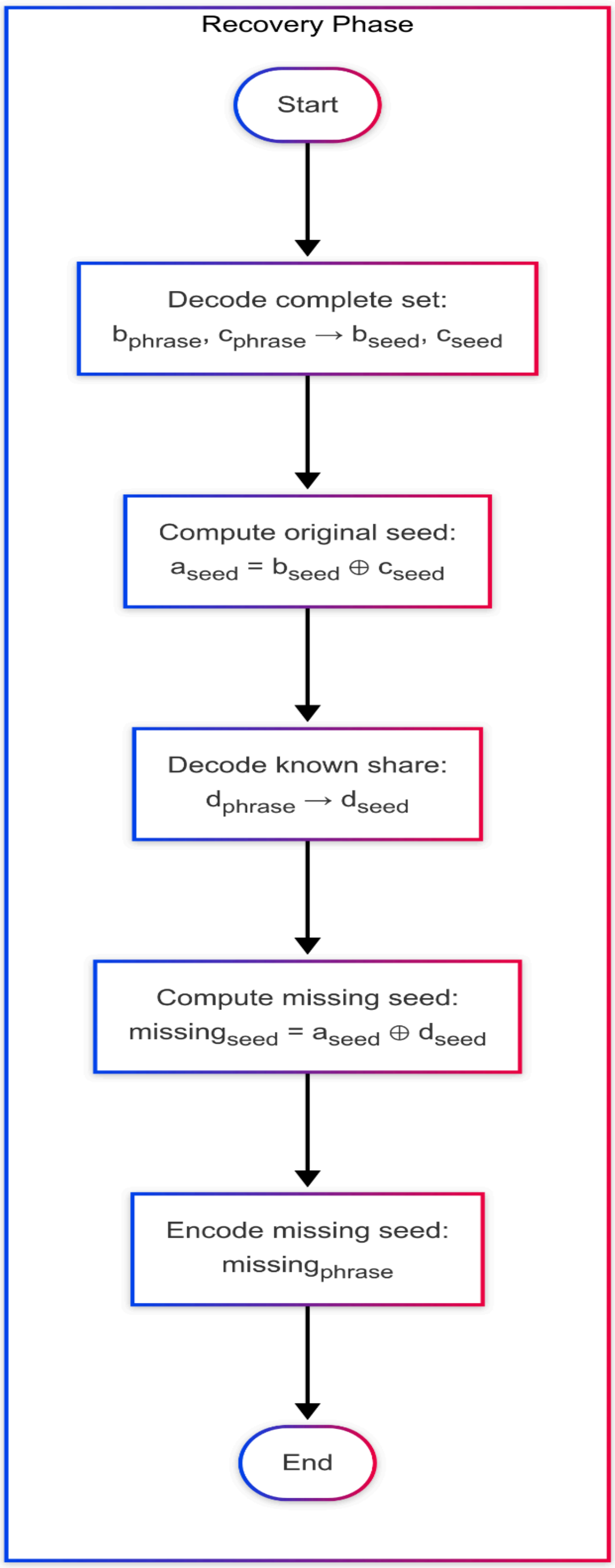
2. Recovering Lost Phrases

The second phase of the project involves implementing the process for recovering of lost phrases, which in an ordinary environment would cause us a lot of problems, but can be dealt with quite easily if more than one pair of phrases have been created with the properties of XOR.

These will be steps for recovering a lost phrase:

1. To recover a lost phrase, we assume that the user has one complete set of phrases and another set that is incomplete (i.e., one or more phrases are missing). Let the complete set be denoted as $\{b_{phrase}, c_{phrase}\}$, and the incomplete set as $\{d_{phrase}, null, null, ...\}$, where null represents one or more missing phrases. Using the complete set, we can reconstruct the original seed through the XOR of the decoded seed values from b_{phrase} and c_{phrase}, that is $b_{seed} \oplus c_{seed} = a_{seed}$ as shown in Fig. 3 as well, now we will our original seed to get the back the lost phrase.
2. Once the original seed is recovered, it can be XORed with the known phrases in the incomplete set to retrieve the missing phrases. This method allows for recovery even when multiple phrases are lost, provided that the cumulative XOR value of the original seed and the known shares is available., where null can be one or more than one lost phrases. So we will XOR all the values in the incomplete set, except the unknown values, that is d_{phrase} in this case, so now we know that $d_{seed} \oplus missing_{seed} = a_{seed}$. So we can find out $missing_{seed} = d_{seed} \oplus a_{seed}$. This way we find the missing seed, then encode it into the $missing_{phrase}$. Which we could then perform the creation of a phrase process to divide it into more phrases.

This way we can easily recover a missing phrase as long as we have one set of valid phrases, making it secure as well as efficient, since then losing one of the keys does not become much of a problem.



Recovery Phase
Start
Decode complete set:
$b_{phrase}, c_{phrase} \longrightarrow b_{seed}, c_{seed}$
Compute original seed:
$a_{seed} = b_{seed} \oplus c_{seed}$
Decode known share:
$d_{phrase} \longrightarrow d_{seed}$
Compute missing seed:
$missing_{seed} = a_{seed} \oplus d_{seed}$
Encode missing seed:
$missing_{phrase}$
End

Fig 3. Recovering the Missing Phrases

Project Timeline

1. Week 1-2 - Discussing and Finalizing with mentor

    - Make contact with the mentor over telegram and connect with him.
    - Discuss all the details regarding the project over the telegram
    - Finalize each and every functionality and details regarding the project.
    - Start working on the project.

2. Week 3-4 - Creating Split Function

    - Implement xor_split(a_{seed},n), which would divide a_{seed} into n different seeds, whose XOR will be a_{seed}.
    - BIP-39 encode/decode functions for each share.
    - Pure-Python unit tests for all edge cases.

3. Week 5 - Integrating Split Function into SeedSigner

    - Add new module(seedxor.py) under src/seedsigned/models
    - Wire SeedXOR->Generate into SeedSigner’s Menu.
    - Design and Implement screens for entering “# of shares” and displaying resulting mnemonics, along with an option to regenerate the mnemonics.

4. Week 6-7 - Creating Recover Function

- Implement xor_recover(set_a,set_b), which would take two sets of phrases as input, one which would be missing a few phrases.
- Decode both the sets using BIP-39.
- XOR all of these seeds, and then recover the missing seed.
- Encode the missing seed mnemonically and then return it to the user.

5. Week 8 - Integrating Recover Function into SeedSigner

- Wire SeedXOR->Recover into SeedSigner's menu.
- Design and implement screens for inputting missing and known mnemonics.
- Displaying recovered mnemonics.

Benefits to Community

- **Stronger Key Management**
  Enables hobbyists and advanced users to split their BIP-39 seed into as many independent shares as desired, reducing single-point-of-failure risk..
- **Recovery Assurance**
  By leveraging XOR's reversibility, users gain a built-in recovery path even if one or more shares go missing, dramatically lowering the chance of permanent wallet loss.

- **Seamless Integration**
   As a drop-in enhancement to SeedSigner, SeedXOR requires no additional hardware or external tools. It preserves the air-gapped workflow and BIP-39 compatibility that existing SeedSigner users rely on.
- **Speed**
  Since this method makes use of bitwise operations, it is quite fast and easy as compared to other forms of operations. Giving it another reason to be used.

Future Deliverables

- Rather than requiring all the phrases, we could change it to so that any M out of the N possible phrases in a set would be enough to get the final seed, making losing of phrases even less of a problem.
- Adding checksums to the end of the XOR values, to make sure that they are correct and there hasn't been any corruption of data.

Why Me

As a Candidate Master on Codeforces and a 6⭐ coder on CodeChef, I've built complex algorithms and bitwise solutions under strict constraints skills directly applicable to SeedXOR's core logic and have dealt with many hashing related libraries in python and am quite eager to collaborate with SeedSigner mentor's and community.