Match Unlabeled Seed Backups to Their Wallets

November 28, 2026 · Document #442 · bip39-phrase.com

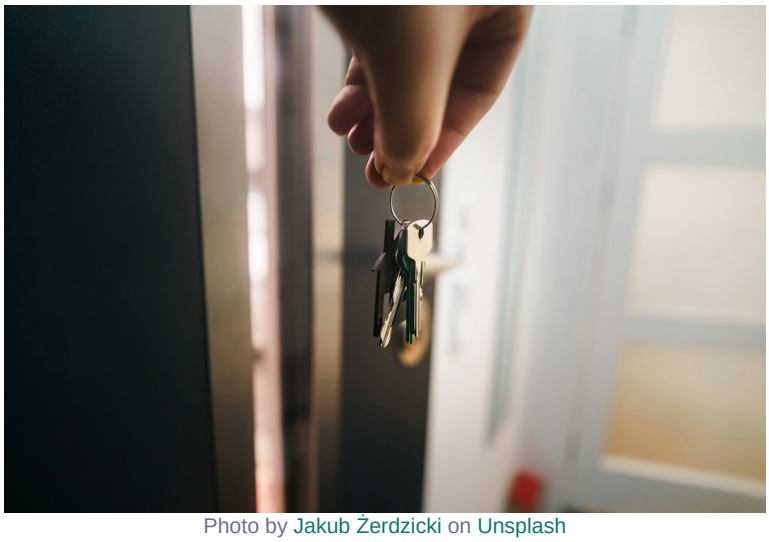

Photo by Jakub Żerdzicki on Unsplash

Match unlabeled seed backups offline using checksum checks, master fingerprints and known addresses, then label them without relying on balances.

Contents

Inventory the backups without combining them	2
Check the backup format and checksum offline	2
Derive fingerprints and candidate addresses	3
Confirm the match and label by fingerprint	4
FAQ	4
Sources	4
Example sequences from the BIP39 wordlist	5
Reading a BIP39 phrase	6
See also	6

Several unlabeled word backups cannot be identified reliably by checking which one displays a balance. The same phrase can produce different wallets through passphrases, derivation paths, account numbers and address types. Match each backup using offline-derived identifiers and addresses from your existing records, then label it without exposing its words. ([raw.githubusercontent.com](https://raw.githubusercontent.com/bitcoin/bips/master/bip-0039.mediawiki))

Inventory the backups without combining them

Assign each physical backup a temporary reference such as `Backup A`, `Backup B` and `Backup C`. Keep every phrase's words and ordering separate. Do not combine similar-looking copies or assume that two phrases differing by one word represent the same wallet.

Gather non-secret records before opening any recovery tool:

- Receiving addresses from invoices, withdrawal confirmations or transaction records you already trust.
- Wallet software names, hardware-wallet models and approximate creation dates.
- Known networks, Bitcoin address types, account numbers and any separately stored passphrase instructions.
- Existing public-only wallet descriptors or master fingerprints, if available. Descriptors containing extended public keys reveal financial activity; descriptors can also contain private keys, so check before treating them as non-secret. ([raw.githubusercontent.com](https://raw.githubusercontent.com/bitcoin/bips/master/bip-0032.mediawiki))

Do not photograph the backups, upload them to an identification service or enter them into a block explorer. If you check an address online, consider the privacy exposure; keep seed phrases offline.

Check the backup format and checksum offline

Use a trusted hardware wallet with on-device recovery, or verified recovery software on a dedicated offline computer. Obtain and verify software before introducing secrets. Disconnect networking and avoid cloud storage, clipboard synchronization and shared folders. Keep the recovery environment offline afterward, too.

1. Identify the backup format. BIP39 uses `12`, `15`, `18`, `21` or `24` words. Electrum 2.0 and later generate seeds using a different versioning scheme, while SLIP39 backups use mnemonic shares rather than BIP39 phrases. Word count alone does not establish the format. ([raw.githubusercontent.com](https://raw.githubusercontent.com/bitcoin/bips/master/bip-0039.mediawiki))
2. For BIP39, check every word against the correct language wordlist and preserve the exact order. Each BIP39 word represents an 11-bit index into a 2,048-word list. ([raw.githubusercontent.com](https://raw.githubusercontent.com/bitcoin/bips/master/bip-0039.mediawiki))
3. Verify the checksum offline. BIP39 appends `ENT / 32` checksum bits from the entropy's SHA256 hash before encoding it: 12 words have 4 checksum bits; 24 words have 8. ([raw.githubusercontent.com](https://raw.githubusercontent.com/bitcoin/bips/master/bip-0039.mediawiki))
4. Record only the result beside the temporary reference: passes BIP39 validation, fails BIP39 validation, or another identified format.

A valid checksum establishes internal consistency, not ownership or correct transcription. Uniformly random sequences of valid words pass a 12-word checksum with probability `1/16`; for 24 words, it is `1/256`. These rates follow from the checksum sizes and are not universal error-detection rates for particular transcription mistakes. A failed BIP39 check does not prove that an Electrum backup is damaged. ([raw.githubusercontent.com](https://raw.githubusercontent.com/trezor/python-mnemonic/

master/src/mnemonic/mnemonic.py))

If validation fails, recheck handwriting and order before considering recovery tools. Never let a website “repair” the phrase.

Derive fingerprints and candidate addresses

For BIP39, the mnemonic and optional passphrase generate a 512-bit seed using PBKDF2-HMAC-SHA512 with `2048` iterations and Unicode NFKD normalization of both inputs. Every passphrase produces a seed; there is no built-in passphrase correctness check. Start with the empty passphrase if your records do not indicate another one, then test plausible, known alternatives offline. ([raw.githubusercontent.com](https://raw.githubusercontent.com/bitcoin/bips/master/bip-0039.mediawiki))

A **BIP32 master fingerprint** is the first four bytes of HASH160 of the compressed master public key, usually displayed as eight hexadecimal characters. Derive it for each candidate phrase-and-passphrase combination. Do not confuse it with a child key’s parent-fingerprint field. ([raw.githubusercontent.com](https://raw.githubusercontent.com/bitcoin/bips/master/bip-0032.mediawiki))

The fingerprint is a short reference to a root, not an individual account or a unique identifier. Its 32-bit size permits collisions. Confirm an identification with addresses as well. ([raw.githubusercontent.com](https://raw.githubusercontent.com/bitcoin/bips/master/bip-0032.mediawiki))

1. Restore one candidate offline and record its master fingerprint beside its temporary reference, noting which passphrase variant it represents without recording the passphrase. Do not copy private keys or the seed into your worksheet.
2. Derive addresses using the likely network, account and address type. For Bitcoin mainnet account zero, common first receiving paths are `m/44'/0'/0'/0/0` for legacy P2PKH, `m/49'/0'/0'/0/0` for nested SegWit P2SH-P2WPKH, `m/84'/0'/0'/0/0` for native SegWit P2WPKH and `m/86'/0'/0'/0/0` for BIP86 single-key Taproot. Use software that implements the corresponding address construction, including the BIP86 Taproot key tweak. ([raw.githubusercontent.com](https://raw.githubusercontent.com/bitcoin/bips/master/bip-0044.mediawiki))
3. Compare the exact derived address with an address in your records. Derive several receiving addresses when necessary; a previously used wallet may not have exposed index zero in the records you retained.
4. If nothing matches, check documented account numbers and change addresses. In these Bitcoin conventions, the change branch replaces the next-to-last `0` with `1`. Expand searches systematically rather than changing several parameters at once. ([raw.githubusercontent.com](https://raw.githubusercontent.com/bitcoin/bips/master/bip-0044.mediawiki))

Other chains require their own derivation and address-encoding rules. For a typical Bitcoin multisignature wallet, also recover the other cosigners’ public keys, derivation details and wallet policy, preferably from a trusted public descriptor. One cosigner’s seed alone generally cannot reproduce its receiving addresses. ([raw.githubusercontent.com](https://raw.githubusercontent.com/bitcoin/bips/master/bip-0044.mediawiki))

Confirm the match and label by fingerprint

Once an address matches, record the fingerprint, network, wallet policy or path, account number, receiving or change branch, and matching address index. Prefer confirmation against a second known address or an existing trusted public descriptor.

Label the physical backup with its **master fingerprint**, not its balance. If the same words support multiple passphrase wallets, distinguish their fingerprints in the separate inventory. Add the wallet role and derivation details there. If a passphrase is required, note that requirement without storing the passphrase beside the words.

A zero balance does not disqualify a backup: funds may have moved, or the wrong account may be selected. Matching known addresses is stronger evidence than a wallet's balance display. Retain unmatched backups until their format and intended wallet are understood. ([raw.githubusercontent.com](https://raw.githubusercontent.com/bitcoin/bips/master/bip-0044.mediawiki))

FAQ

Can two backups have the same fingerprint?

Yes. Identical master public keys share fingerprints, and unrelated roots can collide. Compare derived addresses before treating backups as duplicates. ([raw.githubusercontent.com](https://raw.githubusercontent.com/bitcoin/bips/master/bip-0032.mediawiki))

Does a valid phrase prove the passphrase is correct?

No. BIP39 has no built-in passphrase correctness check. Different passphrases normally produce different seeds, but strings that become identical after NFKD normalization produce the same seed for the same mnemonic. Confirm against trusted records. ([raw.githubusercontent.com](https://raw.githubusercontent.com/trezor/python-mnemonic/master/src/mnemonic/mnemonic.py))

Must I connect the recovered wallet to check it?

No. Fingerprints and addresses can be derived offline. Compare them locally with your records; use a separate watch-only setup containing only public wallet data if transaction history is needed. ([raw.githubusercontent.com](https://raw.githubusercontent.com/bitcoin/bips/master/bip-0032.mediawiki))

Sources

- BIP39: Mnemonic code for generating deterministic keys (github.com)
- BIP32: Hierarchical Deterministic Wallets (github.com)
- BIP44: Multi-Account Hierarchy for Deterministic Wallets (github.com)

> Continue reading on bip39-phrase.com: Derivation Path Explained: What m/44'/0'/0'/0/0 Means
> More on this topic: Recovery

Example sequences from the BIP39 wordlist

Randomly generated sequences of words from the BIP39 English wordlist. None of them passes the BIP39 checksum, so no wallet will accept them: they show the format only. Never store funds with a phrase you found in a document.

Example 1

1 east	2 shoot	3 artist	4 panther
5 direct	6 warfare	7 slab	8 spy
9 royal	10 chalk	11 clown	12 purse

Example 2

1 obey	2 soccer	3 abuse	4 upon
5 wall	6 maple	7 poem	8 moral
9 wink	10 ocean	11 action	12 battle

Example 3

1 teach	2 sun	3 matter	4 wreck
5 forget	6 property	7 nest	8 clump
9 virtual	10 jazz	11 warm	12 source

Example 4

1 gentle	2 inmate	3 assist	4 unlock
5 magic	6 alien	7 wink	8 loan
9 print	10 term	11 setup	12 bicycle

Example 5

1 ahead	2 buffalo	3 divide	4 donkey
5 vital	6 hawk	7 wasp	8 rare
9 drop	10 shock	11 indoor	12 verify

Example 6

1 hybrid	2 increase	3 install	4 employ
5 street	6 hospital	7 guard	8 media
9 giggle	10 faint	11 any	12 course

Example 7

1 mask	2 indicate	3 fish	4 office
5 citizen	6 flush	7 cinnamon	8 involve
9 horror	10 elevator	11 core	12 luggage

Example 8			
1 doctor	2 learn	3 depart	4 shine
5 uphold	6 ladder	7 burden	8 fossil
9 liquid	10 sunny	11 ancient	12 alter

Reading a BIP39 phrase

Every word stands for an 11-bit number: its position in the 2048-word list, counted from 0. Here is Example 1 decoded word by word. Each word links to its page on bip39-phrase.com, where the list is numbered from 1.

#	Word	Index (0-2047)	Word no.	11 bits
1	east	557	558	01000101101
2	shoot	1588	1589	11000110100
3	artist	104	105	00001101000
4	panther	1278	1279	10011111110
5	direct	500	501	00111110100
6	warfare	1976	1977	11110111000
7	slab	1623	1624	11001010111
8	spy	1690	1691	11010011010
9	royal	1509	1510	10111100101
10	chalk	303	304	00100101111
11	clown	352	353	00101100000
12	purse	1396	1397	10101110100

The 12 words carry 132 bits: 128 bits of entropy followed by a 4-bit checksum. The checksum must equal the first 4 bits of the SHA-256 hash of the entropy.

`Entropy (hex): 45b8d0344fe3e9ee32be9abca4bcb057`

This sequence ends in the checksum bits 0100, but SHA-256 of its entropy starts with 0011, so wallets reject it as an invalid phrase. That is the point of the checksum: about 15 of every 16 random or mistyped sequences fail it.

See also

- What to Write Next to Your Seed Phrase · June 11, 2025
- What a Seed Phrase Restores—and What It Leaves Out · March 31, 2025
- Seed Phrases Don't Expire: What Ages Instead · June 15, 2025
- Seed Word Not on the BIP39 List: Find the Right Word · November 25, 2026

Educational document. The example sequences are random and fail the BIP39 checksum: they are not wallets and must never hold funds. Security notes: bip39-phrase.com/security/