

Coins Sent to Mistyped or Wrongly Restored Wallets

October 15, 2026 · Document #354 · bip39-phrase.com



Photo by Shubham Dhage on Unsplash

Funds sent to an incorrectly restored wallet remain safe on-chain but can only be accessed by reproducing the exact derivation path and passphrase.

Contents

Mechanisms of Wallet Divergence	2
Comparison of Divergence Vectors	2
Immediate Actions Before Closing the Session	2
Step-by-Step Retracing Protocol	3
FAQ	3
Sources	4
Sample sequences of BIP39 words	4
Reading a BIP39 phrase	5

A cryptocurrency transaction sent to an address derived from a mistyped passphrase or an unintended derivation path is permanently recorded on the blockchain. The network cannot detect user intent; it validates transactions solely against cryptographic proofs. If you generate a receiving address from an incorrectly restored wallet, the funds reside under the private keys of that specific, unintended key tree. The assets remain secure and unspent on-chain, but spending them requires recreating the exact cryptographic path and inputs that generated that receiving address.

Mechanisms of Wallet Divergence

Deterministic wallets implement **BIP39** and **BIP32** specifications. An altered parameter at any stage changes the resulting keys completely.

- **Passphrase divergence:** The BIP39 passphrase (often called the 13th or 25th word) acts as the salt (prepended with the literal string "mnemonic") in the PBKDF2 key-stretching function, while the mnemonic sentence serves as the password. Unlike the mnemonic wordlist, the passphrase has no checksum, no predefined dictionary, and strict case-sensitive formatting. Entering Passphrase instead of passphrase, or adding an accidental trailing space, produces an entirely distinct 512-bit seed without throwing an error.
- **Derivation path divergence:** The master seed produces child keys via hierarchical deterministic paths. If a wallet defaults to BIP44 (m/44'/0'/0'/0/0) instead of BIP84 (m/84'/0'/0'/0/0), the generated addresses differ, even when using the identical seed phrase and passphrase.
- **Account index divergence:** Most software defaults to account 0', but entering account 1' (m/84'/0'/1'/0/0) generates a separate chain of addresses.
- **Address script differences:** Even along the same path, wrapping a public key in P2PKH, P2SH-P2WPKH, P2WPKH, or P2TR produces legacy (1...), nested SegWit (3...), native SegWit (bc1q...), or Taproot (bc1p...) addresses respectively.

Comparison of Divergence Vectors

Divergence Vector	Detectable by Software Checksum	Resulting Key Tree	Difficulty to Retrace
Mnemonic Word Typo	Yes (wordlist and checksum validation)	Wallet rejects restoration	Low (checksum flags error)
BIP39 Passphrase Typo	No (arbitrary string allowed)	Completely different master seed	Moderate to High
Derivation Scheme Mismatch	No	Same seed, alternate derivation branch	Low (standardized paths)
Non-zero Account Index	No	Same seed, parallel branch	Low (sequential integer scan)

Immediate Actions Before Closing the Session

If you discover an address received funds inside an interface with questionable settings, your working memory and runtime environment contain critical forensic data. Do not shut down the application or clear your operating system clipboard.

- 1 Copy the exact receiving address that received the transaction.
- 2 Copy the transaction identifier (txid) from the blockchain explorer.
- 3 Export the extended public key (xpub, ypub, or zpub) displayed in the current wallet settings, if available.
- 4 Document the exact derivation path shown in the wallet's technical details menu.

- 5 Note the exact keystrokes, keyboard layout, and caps lock status used during passphrase entry.

Step-by-Step Retracing Protocol

When standard wallet interfaces display a zero balance because the incorrect profile was closed, use an offline recovery process to systematically retrace the parameters.

- 1 **Verify the target script type:** Look at the prefix of the funded address. For Bitcoin, an address starting with 1 indicates P2PKH, 3 indicates P2SH (such as P2SH-P2WPKH), bc1q indicates P2WPKH, and bc1p indicates P2TR. This narrows down standard paths defined in BIP44, BIP49, BIP84, and BIP86.
- 2 **Check standard derivation paths without passphrase:** Restore the known mnemonic phrase on an isolated, offline machine. Check if common paths such as $m/84'/0'/0'/0$ or $m/44'/0'/0'/0$ generate the target address within the first 20 address indices (the standard gap limit).
- 3 **Scan higher account numbers:** Check account indices $m/84'/0'/x'/0/0$ up to $x = 10$. Some hardware and software clients switch accounts automatically if previous accounts contain past transactions.
- 4 **Generate typo permutations for the passphrase:** If standard paths fail, the deviation is likely in the BIP39 passphrase. Compile a candidate list of common input errors based on your intended passphrase:
 - Missing or doubled characters.
 - Shift-key errors (such as @ instead of 2).
 - Accidental leading or trailing whitespace.
 - Capitalization discrepancies.
 - Autocorrect modifications introduced by mobile keyboards.
- 1 **Perform offline derivation testing:** Using an offline, open-source cryptographic derivation utility, compute the master seed and target child addresses for each candidate passphrase until an address matches the target receiving address. Once matched, the correct passphrase and path combination is identified.

FAQ

Can miners or nodes return coins sent to an address created with a typo?

No. The blockchain consensus rules enforce valid transaction signatures over spending scripts. The network cannot reverse a confirmed transaction or detect that an address was derived from unintended parameters.

Why does BIP39 allow passphrases with typos to open an empty wallet?

The BIP39 specification treats the passphrase as an arbitrary byte sequence used directly in PBKDF2 hashing. Because any string—including an empty string—is cryptographically valid, the protocol has no mechanism to distinguish an intended passphrase from a mistyped one.

If I know my mnemonic and the exact funded address, can I calculate the missing passphrase?

No direct mathematical inversion is possible because PBKDF2 with HMAC-SHA512 is a one-way function. You cannot compute the passphrase from the address; you must generate candidate passphrases and check if their derived addresses match the target.

Sources

- [BIP 39: Mnemonic code for generating deterministic keys \(github.com\)](#)
- [BIP 44: Multi-Account Hierarchy for Deterministic Wallets \(github.com\)](#)
- [BIP 84: Derivation scheme for P2WPKH based accounts \(github.com\)](#)

Continue reading on bip39-phrase.com: [How to Fix Seed Phrase Errors](#)

More on this topic: [Recovery](#)

Sample sequences of BIP39 words

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	black	2	layer	3	material
4	crack	5	state	6	joy
7	imitate	8	reveal	9	merry
10	witness	11	erase	12	bread

Example 2

1	craft	2	immune	3	soul
4	loop	5	found	6	practice
7	avoid	8	sustain	9	across
10	bomb	11	answer	12	theme

Example 3

1	decide	2	benefit	3	lake
4	cake	5	stomach	6	month
7	volume	8	endless	9	consider
10	hood	11	six	12	boil

Example 4

1	drum	2	utility	3	try
4	work	5	animal	6	once
7	hole	8	family	9	brown
10	cream	11	black	12	giant

Example 5

1	nominee	2	strike	3	payment
4	rug	5	raw	6	kick
7	basic	8	useful	9	more
10	fortune	11	wait	12	company

Example 6

1	slight	2	race	3	total
4	trim	5	ticket	6	good
7	garlic	8	expand	9	neck
10	drip	11	pilot	12	perfect

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	black	183	184	00010110111
2	layer	1010	1011	01111110010
3	material	1095	1096	10001000111
4	crack	398	399	00110001110
5	state	1702	1703	11010100110
6	joy	964	965	01111000100
7	imitate	907	908	01110001011
8	reveal	1475	1476	10111000011
9	merry	1117	1118	10001011101
10	witness	2021	2022	11111100101
11	erase	611	612	01001100011
12	bread	219	220	00011011011

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): 16efca2398ed4cf11c5dc38bbf95318d

This sequence ends in the checksum bits 1011, but SHA-256 of its entropy starts with 0111, 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.

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