

Why There Is No BIP39 Version 2

October 16, 2026 · Document #353 · bip39-pharse.com



Photo by Invest Europe on Unsplash

Bitcoin Improvement Proposals cannot introduce breaking changes once deployed; fixes and alternative designs must publish under new proposal numbers.

Contents

The Immutable Lifecycle of Deployed BIPs	1
Permissible Additions vs. Incompatible Changes	2
Separate Standards for New Architecture	2
FAQ	3
Sources	3
Example BIP39 word sequences	4
Reading a BIP39 phrase	5

A standard that defines cryptographic key derivation cannot introduce breaking changes once deployed in software and hardware devices. Changing the underlying mathematical rules of Bitcoin Improvement Proposal 39 (**BIP39**) would alter the resulting 512-bit binary seed for millions of existing seed phrases, leading to silent derivation mismatches and inaccessible funds. For this reason, specifications like BIP39 do not receive a "version 2" update under the same proposal identifier.

The Immutable Lifecycle of Deployed BIPs

Under the Bitcoin Improvement Proposal process defined in BIP 2, standards move through states such as Draft, Proposed, and Final. Once a BIP reaches widespread deployment across multiple independent implementations—even if its formal repository status remains Proposed—the specification becomes effectively immutable.

Updating a deployed derivation standard carries catastrophic failure modes:

- **Loss of Determinism:** A user entering abandon abandon abandon abandon abandon abandon abandon abandon abandon abandon about must deterministically arrive at the exact same master extended private key across compliant implementations, regardless of whether their device was manufactured years ago or today.
- **Silent Fund Inaccessibility:** If a modified standard altered PBKDF2 iterations, salt generation, or checksum rules, a wallet using the "new" BIP39 would derive different addresses from identical words. Users would observe a balance of zero with no clear indication that derivation parameters differed.
- **Ecosystem Fragmentation:** Wallet software from competing vendors must remain interoperable. Introducing incompatible behavior under the same identifier destroys trust in seed phrase portability.

Because the specification process prioritizes backwards compatibility, errata submitted to BIP39 are restricted to typographical corrections, clarifying notes, and additive elements that do not invalidate existing phrases.

Permissible Additions vs. Incompatible Changes

Modifications to BIP39 are strictly evaluated on whether they maintain determinism for all phrases generated under the original text.

Additive Enhancements

The primary changes accepted into BIP39 post-publication have been additional localized wordlists. Each new wordlist—such as Japanese, Spanish, Italian, Czech, Korean, Traditional Chinese, Simplified Chinese, French, and Portuguese—provides a dedicated set of exactly 2,048 sorted words. These additions do not alter the generation or interpretation of existing English phrases. Furthermore, implementations apply Unicode normalization (NFKD) to mnemonic sentences and passphrases so character variations evaluate to consistent binary representations.

Incompatible Modifications

Security researchers and developers have proposed several improvements to address BIP39's known design limitations. Such changes cannot be merged into BIP39 because they alter cryptographic output:

- 1 **Upgrading Key Derivation Functions:** BIP39 relies on PBKDF2-HMAC-SHA512 with 2048 rounds and a salt prefixed by mnemonic. Replacing PBKDF2 with memory-hard functions like Argon2id or increasing the iteration count immediately changes the derived seed.
- 2 **Altering Checksum Formulas:** BIP39 computes its checksum using SHA256 on the raw entropy, appending between 4 and 8 bits to the initial payload. Restructuring the checksum location or length invalidates mnemonic parsing.
- 3 **Adding Embedded Version Flags:** BIP39 contains no internal byte indicating the target coin, account structure, or address derivation path.

Separate Standards for New Architecture

When developers require functionality that fundamentally breaks compatibility with BIP39, they must create an entirely new specification with its own designated number or identifier.

Specification	Primary Innovation	Compatibility with BIP39	Key Derivation Method
BIP39	Standardized 12-24 word mnemonic encoding	Native baseline	PBKDF2-HMAC-SHA512 (2048 rounds)
Electrum Mnemonic	Embedded version prefix inside the phrase	Incompatible	PBKDF2-HMAC-SHA512 (2048 rounds with <code>electrum</code> salt prefix)
SLIP-0039	Shamir's Secret Sharing (split recovery phrases)	Incompatible	PBKDF2-HMAC-SHA256 with custom rounds
BIP85	Child entropy derivation from an existing root	Additive via BIP32	Uses standard BIP32/BIP39 layers

Electrum chose not to adopt BIP39 when it was introduced, noting that BIP39 lacked protocol versioning and forced wallets to guess derivation paths. Electrum implemented its own mnemonic standard where an embedded hash prefix dictates the phrase version and wallet script type.

Similarly, SatoshiLabs developed SLIP-0039 (Shamir Mnemonic) to allow seed phrases to be split into multiple threshold shares (for example, 3-of-5). Because SLIP-0039 incorporates custom wordlists, an expanded 10-bit per word scheme, and a distinct checksum structure, it was published as an entirely distinct standard rather than an update to BIP39.

When enhancements are fully compatible with existing wallet infrastructures, they build on top of BIP39 rather than replacing it. BIP85 is a clear example: it uses an existing BIP32 master key derived from a BIP39 phrase to deterministically output child seeds, including standard BIP39 phrases for use on secondary devices, leaving the base BIP39 specification completely untouched.

FAQ

Why can't BIP39 simply increase its PBKDF2 iterations to improve security against brute-force attacks?

Increasing the iteration count beyond 2048 would yield a completely different 512-bit seed from the same recovery phrase, preventing existing wallets from recovering existing funds.

Why does BIP39 not include an internal version number like Electrum seed phrases do?

BIP39 was published and widely adopted without an internal version indicator, relying on fixed word count and checksum ratios; adding a version number requires altering the data payload format, which breaks standard parsing.

If a wallet creates a custom 12-word phrase using a different hashing algorithm, is it still BIP39?

No. A phrase derived using different hash functions, iteration counts, or salts violates the BIP39 specification and cannot be recovered in standard BIP39-compliant hardware or software.

Sources

- [BIP 39: Mnemonic code for generating deterministic keys \(github.com\)](#)

- [BIP 2: BIP Process, Revised \(github.com\)](#)
- [SLIP-0039: Shamir's Secret-Sharing for Mnemonic Codes \(github.com\)](#)

Continue reading on bip39-pharse.com: [Bitcoin Improvement Proposal \(BIP\) & Full List](#)

More on this topic: [BIP Standards](#)

Example BIP39 word sequences

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 online	2 dinosaur	3 borrow	4 fold
5 domain	6 gaze	7 opera	8 kitten
9 runway	10 cushion	11 injury	12 horror

Example 2

1 roof	2 soft	3 scrap	4 cabbage
5 crack	6 mix	7 nasty	8 left
9 angry	10 caution	11 noble	12 captain

Example 3

1 foam	2 ahead	3 foam	4 believe
5 spy	6 limb	7 hedgehog	8 shadow
9 cave	10 wire	11 spike	12 like

Example 4

1 length	2 symbol	3 labor	4 chat
5 sister	6 valve	7 skin	8 drive
9 spell	10 firm	11 bulk	12 gather

Example 5

1 bacon	2 super	3 radio	4 harsh
5 safe	6 bar	7 divert	8 around
9 pause	10 card	11 spot	12 buzz

Example 6

1 casual	2 grace	3 drip	4 strong
5 ivory	6 build	7 course	8 vehicle
9 parent	10 include	11 need	12 provide

Example 7

1	affair	2	gather	3	average	4	usual
5	maid	6	brain	7	carpet	8	sudden
9	chef	10	hurry	11	alley	12	project

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	online	1239	1240	10011010111
2	dinosaur	499	500	00111110011
3	borrow	208	209	00011010000
4	fold	724	725	01011010100
5	domain	519	520	01000000111
6	gaze	773	774	01100000101
7	opera	1242	1243	10011011010
8	kitten	986	987	01111011010
9	runway	1515	1516	10111101011
10	cushion	435	436	00110110011
11	injury	929	930	01110100001
12	horror	877	878	01101101101

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): `9ae7cc682d440ec166d3dabd66cdd0b6`

This sequence ends in the checksum bits 1101, 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-phrase.com/security/