Splitting a Passphrase Between Two People: Weak Points

November 11, 2026 · Document #425 · bip39-phrase.com



Photo by Álvaro Serrano on Unsplash

Splitting a passphrase exposes partial information when one half leaks. Compare the risks with complete backups and Shamir shares.

Contents

Why one leaked half makes guessing easier	2
How this mirrors naive seed splitting	2
Choose backups or threshold protection deliberately	3
FAQ	4
Sources	4
BIP39 format examples	4
Reading a BIP39 phrase	5
See also	6

Giving two people half of a BIP39 passphrase creates a fragile security boundary: once either half leaks, an attacker who also has the mnemonic can focus on guessing the missing part. The remaining protection depends on the unpredictability of that part given what has leaked, not the strength of the original complete passphrase.

Why one leaked half makes guessing easier

A BIP39 passphrase is an additional input used with the mnemonic seed phrase to derive a wallet seed. BIP39 applies PBKDF2-HMAC-SHA512 with `2048` iterations, using the normalized mnemonic as the password and `mnemonic` followed by the normalized passphrase as the salt. Both text inputs use UTF-8 with NFKD normalization, and the output is a `512`-bit seed. ([github.com](https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki))

Every passphrase accepted by the BIP39 derivation produces a seed. There is no built-in “wrong passphrase” result. An attacker who has the mnemonic can test guesses locally, derive candidate addresses, and compare them with a known address or blockchain history, accounting for the wallet’s derivation paths and address types. A wallet app’s login limits do not restrict this offline process. ([github.com](https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki))

Suppose a passphrase contains eight words selected independently and uniformly from a list of `2048` distinct words, with a fixed separator. Each selection contributes `11` bits of entropy, giving the complete passphrase `88` bits, provided normalization does not merge distinct choices. If one person receives the first four words and another receives the last four, either half contains only `44` bits of uncertainty.

After one half leaks, the search space falls from `2^88` possibilities to `2^44`, assuming the attacker knows the selection method and split. Testing distinct candidates until the correct one is found takes about half the remaining possibilities on average under these uniform assumptions. The work per guess still includes BIP39 derivation and checking candidate wallets, but the reduction in possibilities is enormous.

That calculation assumes genuinely random words. A quotation, familiar sentence, or personal expression can be much easier to complete. Knowing the beginning may reveal the ending through grammar, context, or a searchable source. Character count alone does not establish entropy.

How this mirrors naive seed splitting

Cutting a mnemonic into two sections has the same structural weakness: each section directly reveals part of the secret. A stolen section removes those words from the attacker’s search for that mnemonic.

BIP39 mnemonics also contain checksum bits. A `12`-word mnemonic encodes `128` bits of entropy plus a `4`-bit checksum. If its first six words are exposed, the remaining six word positions contain `66` bits, but exactly `2^62` completions satisfy the checksum. The original `128`-bit uncertainty has shrunk substantially. The checksum detects many errors, but an incorrect mnemonic can still pass it. ([github.com](https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki))

The passphrase’s words have no BIP39 mnemonic checksum, even if they come from the BIP39 word list. Nevertheless, both splitting methods share the central defect: a fragment reveals actual secret content.

Splitting also creates an availability problem. If both pieces are required and no other recovery route exists, losing either person’s record can make recovery impossible. An unavailable custodian,

unreadable note, forgotten ordering rule, or disagreement can block access just as effectively as theft.

A BIP39 passphrase must be reproduced precisely after normalization. Capitalization, spaces, and punctuation can change the resulting wallet; normalization makes some different character sequences equivalent. Separately stored fragments therefore need an unambiguous joining rule, which becomes another recovery dependency.

Choose backups or threshold protection deliberately

Full copies held separately address loss and availability. Each authorized person or location holds the complete passphrase, so recovery does not require assembling fragments. This can be appropriate when either custodian is trusted to possess the whole secret.

However, full copies provide no threshold confidentiality. Anyone obtaining one complete copy has the passphrase. If they also obtain the corresponding mnemonic, they can derive the wallet's keys. Separating mnemonic backups from passphrase backups helps avoid a single location exposing both inputs, but the recovery plan must explain how authorized recovery brings them together.

Shamir secret sharing serves a different purpose. In the standard mathematical scheme, correctly generated shares below the threshold reveal no information about the secret's contents, although their format can disclose metadata such as its length. A `2-of-3` arrangement requires any two shares for reconstruction and tolerates one unavailable share. A `2-of-2` arrangement requires both and provides no tolerance for a missing share. ([github.com](https://github.com/satoshilabs/slips/blob/master/slip-0039.md))

Use these steps when choosing a design:

1. Define who may recover alone, who must cooperate, and how many lost records the arrangement must survive.
2. For independent recovery, consider complete backups in separate protected locations, accepting that each copy exposes the complete secret if stolen.
3. For cooperation during recovery, use a documented threshold scheme implemented by trusted software or hardware. Reconstruction exposes the secret to the recovery environment; sharing alone does not require cooperation for subsequent spending. Avoid handwritten splitting formulas or online tools that receive real secrets.
4. Confirm exactly what the scheme protects. SLIP-0039 specifies mnemonic shares for backing up an encrypted master secret; it is not interchangeable with BIP39 or an automatic method for sharing an existing BIP39 passphrase. Its optional passphrase is separate from the secret being shared. ([github.com](https://github.com/satoshilabs/slips/blob/master/slip-0039.md))
5. Test recovery before funding a new wallet or relying on a changed backup. Record the required format, threshold, reconstruction procedure, and passphrase handling without placing enough secret material together to defeat the intended separation.

FAQ

Does one leaked passphrase half immediately expose the wallet?

By itself, it usually does not. An attacker using the mnemonic still needs to recover or guess the missing part. A predictable missing half may offer little resistance, and an attacker who already has sufficient private keys can spend without reconstructing the passphrase.

Are complete copies safer than two halves?

They improve availability, while exposing the entire passphrase if one copy is stolen. Choose them when independent recovery is intended.

Can Shamir shares protect a passphrase?

Yes, suitable implementations can share the encoded secret. Verify encoding, reconstruction compatibility, and recovery procedures; do not assume a wallet's mnemonic-sharing feature supports arbitrary passphrase text.

Sources

- BIP39: Mnemonic code for generating deterministic keys (github.com)
- SLIP-0039: Shamir's Secret-Sharing for Mnemonic Codes (github.com)

> Continue reading on bip39-phrase.com: Cold Sharding: Splitting a Private Key Into Offline Shamir Shares
> More on this topic: Seed Phrases

BIP39 format examples

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 build	2 embody	3 knee
4 hand	5 ancient	6 jacket
7 average	8 shell	9 fox
10 type	11 suffer	12 glide

Example 2		
1 among	2 aisle	3 subway
4 chuckle	5 ensure	6 else
7 sauce	8 fluid	9 area
10 crystal	11 crisp	12 survey

Example 3		
1 bracket	2 laptop	3 select
4 size	5 cart	6 cement
7 slush	8 solar	9 letter
10 cream	11 estate	12 tilt

Example 4		
1 sadness	2 cradle	3 ensure
4 program	5 hockey	6 nature
7 grass	8 convince	9 danger
10 inner	11 brush	12 pool

Example 5		
1 isolate	2 donate	3 web
4 balance	5 book	6 wool
7 razor	8 toward	9 verify
10 trend	11 morning	12 wagon

Example 6		
1 clerk	2 task	3 predict
4 give	5 code	6 wish
7 grow	8 wagon	9 ten
10 age	11 alien	12 laptop

Example 7		
1 inch	2 sphere	3 demand
4 change	5 clay	6 roast
7 green	8 know	9 proud
10 siren	11 weasel	12 term

Example 8		
1 blush	2 web	3 feel
4 nuclear	5 sunset	6 hen
7 logic	8 gun	9 surprise
10 purity	11 robot	12 runway

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	build	238	239	00011101110
2	embody	579	580	01001000011
3	knee	988	989	01111011100
4	hand	838	839	01101000110
5	ancient	68	69	00001000100
6	jacket	952	953	01110111000
7	average	126	127	00001111110
8	shell	1579	1580	11000101011
9	fox	738	739	01011100010
10	type	1886	1887	11101011110
11	suffer	1733	1734	11011000101
12	glide	792	793	01100011000

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): 1dc90dee346088ee03f62b5c5d7b62b1`

This sequence ends in the checksum bits 1000, 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

- BIP-39 Seed Phrases: The Most Dangerous Backup You'll Ever Own · June 18, 2025
- The SLIP-39 Digest: Detecting a Bad Share Set · November 4, 2026
- Do You Actually Need a BIP39 Passphrase? · November 9, 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/