

Recovering a Solana Wallet From a Key Array

October 19, 2026 · Document #402 · bip39-phrase.com



Photo by [Aniyora J](#) on [Unsplash](#)

Learn how to restore a Solana account from a 64-byte secret key array or base58 string into Solflare or Phantom, and how it differs from a seed phrase.

Contents

Structure of the Solana Key Array	1
Key Array vs. BIP39 Seed Phrase	2
Importing Into Solflare	2
Importing Into Phantom	3
Verification and Safety Rules	3
FAQ	3
Sources	4
Sample sequences of BIP39 words	4
Reading a BIP39 phrase	5
See also	6

Solana CLI and native developer tooling export private keys as 64-byte arrays of unsigned 8-bit integers, stored in a JSON file such as `id.json`. If you backed up this file or copied its contents, you hold the exact keypair required to sign transactions for that address. However, recovering funds using a byte array differs fundamentally from restoring a wallet with a standard recovery phrase.

Structure of the Solana Key Array

A Solana key array file consists of brackets containing 64 integers, each ranging between 0 and 255. This structure represents a raw **Ed25519 keypair**:

- Bytes 0 through 31 constitute the private key seed.
- Bytes 32 through 63 constitute the derived Ed25519 public key.

Wallets and web interfaces frequently display this same 64-byte payload encoded using Bitcoin's Base58 character set. When encoded in Base58, the 64-byte array becomes an 87-to-88-character alphanumeric string.

Public Solana addresses are the last 32 bytes of this array encoded in Base58, resulting in a string between 32 and 44 characters. If you possess only the first 32 bytes, the Ed25519 curve allows deterministic recalculation of the second 32 bytes. If you have the full 64-byte array, you have the entire cryptographic identity of that specific account.

Key Array vs. BIP39 Seed Phrase

A byte array is not a **seed phrase**, and importing one does not regenerate your broader wallet hierarchy.

Feature	64-Byte Key Array / Base58 String	BIP39 Seed Phrase (with SLIP-0010)
Account Scope	Single account only	Multi-account, multi-chain
Derivation Path	None (static raw keypair)	Hierarchical (e.g., m/44'/501'/0'/0')
Wallet Portability	Direct single-key import	Universal master restore
Output Format	JSON array or 88-char Base58 string	12 or 24 mnemonic words

A BIP39 mnemonic uses a key derivation standard like **SLIP-0010** to calculate individual private keys along sequential index paths. When you recover a wallet using a 12- or 24-word phrase, the client software derives Account 1, Account 2, and any subsequent accounts generated under that tree.

Importing a key array bypasses this hierarchy entirely. The software attaches the raw secret key directly to your interface as a standalone account. It cannot locate or derive any sibling accounts, secondary token addresses derived from a parent seed, or accounts on other blockchains.

Importing Into Solflare

Solflare supports importing raw byte arrays directly without external format conversion.

- 1 Open Solflare and authenticate with your PIN or password.
- 2 Click the account selector icon at the top of the interface.
- 3 Select **Add Account**, then choose **Import Private Key**.
- 4 Open your backup file containing the array formatted like `[12, 234, 85, ...]`.
- 5 Paste the entire string, including the opening `[` and closing `]` square brackets, directly into the input field.
- 6 Solflare parses the JSON byte format directly. Confirm the import and name the account.

Importing Into Phantom

Phantom does not parse raw JSON byte arrays directly in its input fields. It expects private keys to be formatted as Base58 strings.

If you have a 64-byte array and need to import it into Phantom, you must convert the byte array to Base58 before importing, or use an offline command-line tool.

To import an existing Base58 private key string:

- 1 Open Phantom and select the menu icon in the top left.
- 2 Click **Manage Accounts**, then select **Add / Connect Wallet**.
- 3 Click **Import Private Key**.
- 4 Enter an account name and paste the Base58 string (approximately 88 characters).
- 5 Click **Import**.

To convert a byte array to Base58 locally on an offline machine using the Solana CLI:

`solana-keygen pubkey id.json` verifies the public address matching the file.

bs58 encoders in Python, Node.js, or Rust can encode the raw bytes into the required Base58 string without internet access. Never use unverified online encoding websites to transform private keys, as web-based converters expose the secret key to external servers.

Verification and Safety Rules

Once imported, confirm that the visible public address inside the wallet matches the address you intended to restore. Because raw Base58 private keys do not include built-in checksums, decoding errors or typos can yield unintended addresses. You can verify the account by checking its public transaction history on an explorer before funding it.

Keep the following security boundaries in place:

- The imported account is marked as **Imported** in modern wallet interfaces. It is excluded from the backup phrase of that wallet app.
- If you uninstall your browser extension or clear application data, imported accounts are removed from the interface and cannot be restored from the master recovery phrase unless you still possess the separate byte array or Base58 string.
- Delete unencrypted `id.json` files from local drives after transferring funds to an address derived from a secured hardware device or a BIP39 master phrase.

FAQ

Why does Phantom display an "Invalid private key" error for my byte array?

Phantom expects a single Base58-encoded string rather than a JSON array of integers. You must encode the 64 integers into Base58 offline before Phantom will accept it.

Can I calculate my original 12-word seed phrase from a 64-byte array?

No. Derivation functions rely on one-way cryptographic hashing (such as PBKDF2 and HMAC-SHA512). You can derive a private key from a seed phrase, but you cannot reverse the process to determine the original mnemonic words.

If I backup my wallet using its recovery phrase, is the imported key included?

No. Standard recovery phrases only restore keys derived from that specific phrase's master seed. Any account added via private key or byte array import must be backed up independently.

Sources

- [Solana Documentation: File System Wallet](https://docs.solana.com) (docs.solana.com)
- [SLIP-0010: Universal Private Key Derivation from Mnemonic](https://github.com) (github.com)

Continue reading on bip39-phrase.com: [Bitcoin Private Key Format: WIF, Base58 and Hex Explained](#)

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	approve	2	wreck	3	deny
4	pelican	5	sausage	6	spice
7	border	8	easily	9	poverty
10	donate	11	mirror	12	maple

Example 2

1	foam	2	sad	3	embody
4	aim	5	people	6	circle
7	sing	8	grit	9	promote
10	collect	11	culture	12	cart

Example 3

1	basket	2	ethics	3	what
4	melt	5	urban	6	fetch
7	fun	8	utility	9	love
10	common	11	forest	12	adult

Example 4

1	neglect	2	tornado	3	nest
4	fame	5	float	6	dish
7	domain	8	gloom	9	diary
10	okay	11	peace	12	genre

Example 5

1	amount	2	lend	3	system
4	accident	5	fancy	6	boost
7	universe	8	belt	9	favorite
10	finish	11	alcohol	12	lounge

Example 6

1	title	2	explain	3	viable
4	half	5	maze	6	law
7	piano	8	parrot	9	couple
10	crack	11	invite	12	acquire

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	approve	86	87	00001010110
2	wreck	2033	2034	11111110001
3	deny	469	470	00111010101
4	pelican	1298	1299	10100010010
5	sausage	1533	1534	10111111101
6	spice	1676	1677	11010001100
7	border	206	207	00011001110
8	easily	556	557	01000101100
9	poverty	1352	1353	10101001000
10	donate	520	521	01000001000
11	mirror	1132	1133	10001101100
12	maple	1084	1085	10000111100

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): 0adfc4ead12bfba306722ca908223643

This sequence ends in the checksum bits 1100, but SHA-256 of its entropy starts with 0001, 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

- [Coins Sent to Mistyped or Wrongly Restored Wallets](#) · October 15, 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/