Chapter 3

Wallets

Introduction

In this chapter, we will explore the different types of wallets available for the Solana blockchain. These wallets offer secure storage for your Solana tokens and provide easy access to the Solana network. We will cover some of the widely used wallets, describe specific features and support to look out for, and follow up with a very basic discussion on keypairs, seed phrases, and using the Solana CLI to manage a wallet directly at the command line.

Popular Wallets

The following is a list of the most popular wallets used on Solana, but it is not exhaustive. Here, the user really has a great selection to choose from. Inclusion in this list is not an endorsement of any wallet, but aims to describe some of their defining features and reasons users may choose one over another.

Solflare Solflare is a wallet that provides a user-friendly interface for storing and managing Solana tokens. Designed to be compatible with Solana's seamless on-chain transactions, Solflare offers secure storage for your tokens along with access to relevant transaction history. It also supports token swapping and staking SOL tokens in the Solana network. It allows in-wallet conversion to liquid staked tokens (LST). With its intuitive interface, this wallet is ideal for both novice and expert users. The wallet has impressive mobile support (iOS), and is the only wallet to currently support the Keystone hardware wallet.

Phantom Phantom is a fast and secure desktop wallet that supports both Solana and other (EVM-based) chains. Furthermore, Phantom's user interface is sleek and easy to navigate. It has features like a built-in token swap and staking function for SOL tokens in the Solana network. Phantom supports integration with dApps and NFT marketplaces, making it a popular choice for users who want flexibility and accessibility. Phantom wallet is now multi-chain, supporting the Ethereum network, and others.

Backpack Backpack is an innovative multi-chain wallet that supports various blockchains, including Solana. Apart from standard transactional and management functions, it pioneered the concept of xNFTs–executable NFTs. xNFTs are a novel type of NFT-based on smart contracts, which can carry out various actions based on specific triggers. This allows the use of NFTs beyond just as collectibles but as access tokens, event tickets, and other creative use cases. The ability to have executable, on-chain code within the NFT token account is as of yet, an under-explored design space. Backpack's xNFT innovation empowers creators and developers with new ways to make their NFTs dynamic, interactive and engaging.

Solana CLI Solana CLI is a command-line interface wallet that lets users access the Solana network through the terminal window. It is a popular choice for developers and users who prefer to interact with the network using command lines, allowing for automation and customization of tasks. It may be a good fall-back for users of wallets that rely on internet gatekeepers like CloudFlare. It's also an excellent tool for debugging and testing, making it an indispensable asset for developers. It can be used with the Ledger Nano S and Nano X, after installing the udev rules. The author highly recommends becoming comfortable using the CLI as both a wallet, and a tool for general use!

Wallet Features

Wallet selection is not a trivial task, and this is by no means a definitive how-to guide. Our best advice is to try many wallets (with good reputations), and see which are most intuitive to you, and that you are the most comfortable with. Many wallets can provide adequate security, and this is *mostly* contingent on the user protecting their seed phrase (there are exceptions that will go unnamed here). However, there are other features that may increase safety, and user enjoyment. We'll go through these features in the following sections.

SPL tokens All wallets should support the use of SPL-tokens, Solana's equivalent to an ERC-20 on Ethereum. SPL-tokens each have one or more addresses called an `associated token account`, or ATA. The ATA is where information related to the token is stored, including the balance and base58 mint address for the token. It is unclear how wallet support will differ for things like `Token22`–an updated SPL token standard, as well as for compressed NFTs (cNFT).

NFTs A type of SPL-token with a mint quantity of one (1) is a non-fungible token, or NFT. Again, every wallet should support this type of token. An NFT is slightly different in that it contains metadata, and will likely contain information such as a link to an image (probably a monkey or dog), and various traits of the NFT. Wallets have different means of organizing NFTs within the wallet, which can help the user separate legitimate NFTs from spam, which is plentiful, due to Solana's low mint and transaction costs.

xNFT A novel type of NFT called the xNFT (executable NFT) was introduced to Solana by the Backpack wallet team. xNFT are similar to NFTs, with the distinction that their token data may contain arbitrary or executable code inside. This allows the wallet to behave more like an app store, to provide a more moderated experience, enhanced security for executables, and for the user to benefit from greater trust between an xNFT app or collection. The executable data can be updated to create interactive and more feature-rich experiences with users.

cNFT Compressed NFTs are a recent development in non-fungible token technology. These compressed NFTs enable faster and cheaper transactions with few trade-offs versus standard NFTs. With cNFTs, instead of storing the entire metadata of an NFT on the blockchain, only a hash of the metadata is stored, which is much smaller in size. This hash allows for quick verification of ownership without having to access and process large amounts of data. The full metadata, which can include images, videos, or audio files, is then hosted off-chain on various storage solutions like Arweave, Shadow Drive, IPFS (InterPlanetary File System), or other decentralized file storage platforms. Compressed NFTs (cNFT) have recently become extremely plentiful, popularized by teams like Drip Haus, with the ability to scale NFTs into the millions, at minting costs in the tens to hundreds of dollars. When a cNFT is purchased, the contract executes the off-chain metadata retrieval process, and the buyer receives a receipt to confirm their ownership of the NFT. The smart contract then manages the token's royalties and other functionalities of the asset. cNFTs open up new possibilities for creators to monetize their works in various ways on the Solana blockchain.

Staking Solana is a proof-of-stake network, so a big part of network participation is staking. That is, committing coins to the consensus ritual, and in exchange, the user is paid both a portion of the chain's transaction fees, as well as inflationary emissions. Staking allows the user to put their coins to work in securing the network, and also stave off devaluation by the natural token emissions of the protocol. The best in-wallet staking experience is currently with Solflare (Author's opinion), but Phantom is also painless and pleasant to use here.

Liquid Staked Tokens (LST) LSTs enable users to stake their tokens without having to worry about waiting for coins to deactivate or cool-down, and being locked out of the market. Staking to a pool will give the user a liquid-staked token–a claim ticket, essentially–that can be traded on exchanges, or used as collateral for loans. This allows users to earn rewards from staking while still maintaining the flexibility to use their assets for other purposes.

Priority Fees At the current time, there are not many wallets that support the use of the priority fee ("additional fee"), but this number should only grow. The Backpack wallet does in fact support setting of a priority fee rate when enabled using the developer mode. The priority fee can also be set by a dApp, so it is not expressly necessary to have it supported in the wallet to benefit from it (although we greatly prefer wallets that can do it out of the box). Using priority fees is likely not necessary for basic transfers, but may make the difference when submitting trades on DeFi protocols, NFT trading platforms, or trying to win a hotly contested NFT mint. In general, priority fee support is great insurance that you can get what you need to do done, even during times of congestion, or when interacting with popular pieces of state (dApps).

A brief word on priority fees: each transaction has a maximum number of "compute units" (cu) available for it to use. This limit can be set by the `setComputeUnitLimit` instruction of the `ComputeBudget` program, and there are global limit to the maximum compute units a transaction can consume (currently 1.4M cu/transaction). The unused fee paid for compute units will not be collected from the user. To set the "price" for an additional fee, you send a `SetComputeUnitPrice` instruction. The units for expressing this "price" for the `additional_fee` is painfully obscure, as it is measured in microlamports-per-compute unit. Further, you will notice it is not actually a fee, but a rate, which further complicates estimating how much you can expect to pay for a transaction. Good wallets are starting to abstract this away, but until it's the norm, it is worthwhile to understand how to do it. Each lamport is 10^-9 SOL, and a microlamport (uL) is 10^-6 lamports, so each microlamport (or priority rate unit) is 10^-15 SOL (10^-15 SOL/cu). To figure out the additional fee, multiply the rate by 10^-15 SOL/uL, then by the max compute units you expect the transaction to consume. Add in the base fee, and you'll have the total cost of the transaction fee. Like so:

```
total fee = base fee + priority fee
total fee = base fee + compute unit limit * additional_fee * 10^-15
units:
sol = sol + cu * uL/cu * 10^-15 SOL/cu = sol
Example: calculate the (maximum) total fee assuming an additional fee rate of
1000 uL/cu, and a max compute units of 200,000 cu.
(maximum) total fee = 0.000005 + 200,000 * 1,000 * 10^-15 = 0.0000052 SOL = 5.2
* 10^-6 SOL
```

Multicoin UX Some wallets now support tokens on multiple chains. Again, this list is non-exhaustive, but currently Backpack and Phantom have introduced ether / EVM addresses, that are derived from common seed phrase(s) of the Solana keypair(s) that the wallet creates. At this time, the author is unfamiliar with each wallet's level of hardware wallet support. The prospect of using a single wallet for multiple chains greatly reduces the burden to the user of securing multiple keypairs / seed phrases, however, it may also magnify losses should a private key be compromised. Regardless, being able to access all or most of your coins from a single wallet is undeniably an attractive feature.

Cross-Chain Support No wallets currently support in-wallet bridging from one chain to another, but we expect this may happen in the future. The Wormhole / Portal bridge has an API for bridging now, which, when integrated into a wallet, might provide some abstraction for the Wormhole protocol, and greatly enhance user experience, while enabling a truly multi-chain experience.

Security

While there is a host of wallets to use with your private keys, we will elect to use the Solana CLI for demonstration purposes. While it is a command line tool, it is surprisingly accessible, and can be built from source code and run locally on a user's machine, making it a reasonable choice for basic blockchain interactions such as staking and token transfers.

Seed Phrase Creation There are a number of ways to create a seed phrase. The first, and easiest, is using the solana CLI. The particular tool is called `solana-keygen`. Here, we can create a new private key with accompanying 24-word seed phrase as follows:

```
solana-keygen new --outfile new_keypair.json --word-count 24
#
# Generating a new keypair
#
# For added security, enter a BIP39 passphrase
#
# NOTE! This passphrase improves security of the recovery seed phrase NOT the
# keypair file itself, which is stored as insecure plain text
#
# BIP39 Passphrase (empty for none): hw
# Enter same passphrase again: hw
#
# Wrote new keypair to new_keypair.json
#
========================================================================================
========================================================================
# pubkey: E6QvMkasNftdDBtYaYccKQH38j13XUq4Q4CkBReAoNb8
#
========================================================================================
========================================================================
# Save this seed phrase and your BIP39 passphrase to recover your new keypair:
# cluster oyster rescue finger ugly cotton blossom awake model true crazy pigeon
# obscure human nephew dolphin enact evoke pledge gather throw plunge else oven
#
========================================================================================
========================================================================
```

The user should beware that this is a "hot wallet", as the keypair was created using a computer that is (presumably) connected to the internet. An alternate method for generating a more secure seed phrase offline can be found here: https://seedpicker.net/guide/GUIDE.html. Since Solana uses BIP-39 compatible seed phrases, this should work out fine, and will offer superior protection of digital assets. Still, use at your own risk.

Use a Hardware Wallet

Using a single-signature hardware wallet is a notable security upgrade from a hot wallet, and is recommended. Solana is currently supported by Ledger and Keystone hardware wallets. The Keystone wallet is notable for its air-gapped signing (compatible with Solflare on desktop and mobile) using QR codes. For the command line, the user will need a USB-connected wallet such as Ledger Nano S or Ledger Nano X. Here we'll demonstrate how to use a Ledger Nano X with the Solana CLI.

First, the user will need to download and upgrade the firmware, then install the Solana "app" on the device. Please Google this procedure, as the software and accompanying rituals are apt to change.

Next, the user must install the udev rules so that the device can be recognized on the command line.

1. Connect your Ledger hardware wallet to your computer using a USB cable.
2. Open Terminal or any command-line interface.
3. Clone the Ledger repository that contains the necessary udev rules by running the following command:

```
git clone https://github.com/LedgerHQ/udev-rules.git
# check the correct link, subject to change
```

4. Navigate to the cloned repository using the following command:

```
cd udev-rules
```

5. Run the install.sh script with root privileges to install the udev rules for Ledger devices:

```
sudo ./install.sh
```

6. Once the installation is complete, you need to reload the udev rules by running the following command:

```
sudo udevadm control --reload-rules
```

7. Now you can use the Ledger hardware wallet with the Solana CLI. Set the keypair to the hardware wallet to make transfers and other Solana operations:

```
solana config set --keypair usb://ledger?key=0
# Note: Replace 0 in ?key=0 with the appropriate derivation path index if you
have multiple accounts on your Ledger wallet.
```

You have successfully installed the udev rules for Ledger hardware wallet and configured it for use with the Solana CLI.

Using the Solana CLI

Get your Public Key (Address) To get your public key, just type the following:

```
solana address
# E6QvMkasNftdDBtYaYccKQH38j13XUq4Q4CkBReAoNb8
```

Get your Balance To get your wallet's balance, just type the following:

```
solana balance
# 0 SOL
```

Verify You Can Sign Transactions

```
solana-keygen verify E6QvMkasNftdDBtYaYccKQH38j13XUq4Q4CkBReAoNb8
./new_keypair.json
# Verification for public key: E6QvMkasNftdDBtYaYccKQH38j13XUq4Q4CkBReAoNb8:
# Success
```

Transfer SOL To send 1 SOL token to an unfunded recipient using the solana transfer command, follow these steps:

1. Make sure you have the Solana CLI installed. If not, you can install it by following the instructions on the Solana documentation.
2. Open Terminal or any command-line interface.
3. Connect to the Solana network you want to use. For example, if you want to connect to the mainnet-beta network, run the following command:

```
solana config set --url https://api.mainnet-beta.solana.com


# Replace the URL with the appropriate cluster you want to connect to
# (testnet, devnet, or your own RPC provider).
```

4. Use the solana transfer command to send 1 SOL token to the recipient. Replace with the public key of the recipient's Solana wallet. Replace with the amount you want to transfer (1 in this case). The command syntax is as follows:

```
solana transfer <recipient-public-key> <amount-in-SOL>
# Example:
solana transfer HD2EH2pqwNitRWdfVSZ4byemrohfpByfEcLBGkaVafrG 1
```

5. Confirm the transaction by following the instructions provided by the CLI. This may involve confirming the transaction fee and signing the transaction with your hardware wallet (if applicable).

6. Once the transaction is confirmed, the recipient will receive the 1 SOL token.

Please note that the recipient's wallet address must be valid and associated with the Solana network you are using. Additionally, make sure you have sufficient funds in your wallet to cover transaction fees. If the wallet has not been funded previously, you must add the `--allow-unfunded-recipient` flag to the command above.

Stake SOL To create, fund, and activate a stake account using the solana stake-account function in Solana CLI, follow these steps:

1. Make sure you have the Solana CLI installed. If not, you can install it by following the instructions on the Solana documentation.
2. Open Terminal or any command-line interface.
3. Create a new stake account by running the following command:

```
# create the stake account using seed = 0
# (can choose whatever, but using seed = 0, 1, ... n is simple)
solana create-stake-account \
<STAKE_ACCOUNT_KEYPAIR> \
<AMOUNT> \
--from <SOURCE OF BALANCE> \
--seed <INSERT A SEED> \
--stake-authority <STAKE AUTHORITY KEYPAIR> \
--withdraw-authority <WITHDRAW AUTHORITY KEYPAIR>
--fee-payer <TRANSACTION PAYER>
# Example:
solana create-stake-account \
~/my-keypair.json \
0.25 \
--from ~/my-keypair.json \
--seed 0 \
--stake-authority ~/my-keypair.json \
--withdraw-authority ~/my-keypair.json \
--fee-payer ~/my-keypair.json
# show the address of the newly created stake account:
solana-stake-accounts addresses `solana-keygen pubkey ~/my-keypair.json`
--num-accounts 1
# 6f3Q3s5G3HEzPqNfapeYHKdj3GjYyazsBkXeg1JhHsCH
# see the balance and info on your new stake account:
solana stake-account 6f3Q3s5G3HEzPqNfapeYHKdj3GjYyazsBkXeg1JhHsCH
# Balance: 0.25 SOL
# Rent Exempt Reserve: 0.00228288 SOL
# Stake account is undelegated
```

```
# Stake Authority: GDPCvZgGzc5thBLfEn5GVJ2FXQT5ysRYfDyCtVEw3EsU
# Withdraw Authority: GDPCvZgGzc5thBLfEn5GVJ2FXQT5ysRYfDyCtVEw3EsU
```

This will generate a new stake account and provide you with the address of the stake account.

4. Next, we want to delegate to a validator. The validators.app or solanabeach.io sites are invaluable for that. Since we are on devnet, we just pick a `vote address` from the list that `solana validators` shows, for demonstration purposes only.

```
solana delegate stake \
--stake-authority <MAIN KEYPAIR> \
<STAKE ACCOUNT PUBKEY> \
<VALIDATOR VOTE PUBKEY> \
--fee-payer <MAIN KEYPAIR>
# Example:
solana delegate stake \
--stake-authority ~/my-keypair.json \
6f3Q3s5G3HEzPqNfapeYHKdj3GjYyazsBkXeg1JhHsCH \
FwR3PbjS5iyqzLiLugrBqKSa5EKZ4vK9SKs7eQXtT59f \
--fee-payer ~/my-keypair.json
# Then we check the stake account to see that the
# stake is delegated and activating:
solana stake-account 6f3Q3s5G3HEzPqNfapeYHKdj3GjYyazsBkXeg1JhHsCH
# Balance: 0.25 SOL
# Rent Exempt Reserve: 0.00228288 SOL
# Delegated Stake: 0.24771712 SOL
# Active Stake: 0 SOL
# Activating Stake: 0.24771712 SOL
# Stake activates starting from epoch: 590
# Delegated Vote Account Address: FwR3PbjS5iyqzLiLugrBqKSa5EKZ4vK9SKs7eQXtT59f
# Stake Authority: GDPCvZgGzc5thBLfEn5GVJ2FXQT5ysRYfDyCtVEw3EsU
# Withdraw Authority: GDPCvZgGzc5thBLfEn5GVJ2FXQT5ysRYfDyCtVEw3EsU
```

6. Once the transaction is confirmed and the stake account is activated (it may take 1-3 days depending on when the next epoch starts), the stake will be eligible for staking and staking rewards based on the Solana network's staking rules.

7. You can deactivate your stake:

```
solana deactivate-stake \
--stake-authority ~/my-keypair.json \
6f3Q3s5G3HEzPqNfapeYHKdj3GjYyazsBkXeg1JhHsCH \
--fee-payer ~/my-keypair.json
```

8. You can withdraw your stake after it has been successfully deactivated:

```
solana withdraw-stake \
--withdraw-authority <MAIN ADDRESS> \
<STAKE_ACCOUNT_ADDRESS> \
<RECIPIENT_ADDRESS> \
<AMOUNT>
# Example:
solana withdraw-stake \
--withdraw-authority ~/my-keypair.json \
6f3Q3s5G3HEzPqNfapeYHKdj3GjYyazsBkXeg1JhHsCH \
GDPCvZgGzc5thBLfEn5GVJ2FXQT5ysRYfDyCtVEw3EsU \
ALL
# The stake account has now been closed:
solana stake-account 6f3Q3s5G3HEzPqNfapeYHKdj3GjYyazsBkXeg1JhHsCH
# Error: AccountNotFound: pubkey=6f3Q3s5G3HEzPqNfapeYHKdj3GjYyazsBkXeg1JhHsCH
```

In all, this process is not the simplest, but is great to know if for some reason the UI of your wallet provider is down. The CLI also offers a much greater level of control for advanced functionality like setting lockups, unique staking authorities, or specific stake account creation using seeds. See `solana create-stake-account --help` for more info on these fine controls.

Summary

As the Solana network continues to grow, the number of wallets available continues to increase, providing users with more and better options to store and manage their tokens. Regardless of your level of experience or needs, there is a wallet suitable for you. From browser and mobile wallets like Solflare, Phantom, and Backpack, to command-line wallets like Solana CLI, there is something for every user. In this chapter we introduced some of the most important wallet features such as staking and priority fees, and compelling new ideas like xNFTs and cNFTs. We also became proficient using the Solana CLI for general transfers and staking.