BlockSettle

BLOCKSETTLE

Signer

Table of Contents

Introduction

Introduction

The BlockSettle Signer is the program that signs Bitcoin transactions generated by the BlockSettle Terminal. By default, the BlockSettle Terminal uses the BlockSettle Signer transparently, such that the user is never aware of the BlockSettle Signer's existence. However, there are use cases where the BlockSettle Terminal may not have direct control over the BlockSettle Signer. Such cases typically involve the BlockSettle Terminal and BlockSettle Signer running on separate computers.

This document will explain the BlockSettle Signer functionality and how the BlockSettle Signer is invoked. This document is meant primarily for users who have a decentralized network setup, where the BlockSettle Terminal and the BlockSettle Signer will run on separate computers. However, the document may be read by anybody who wishes to understand how the BlockSettle Signer works, and how the BlockSettle Signer interacts with the BlockSettle Terminal.

BlockSettle

Introduction

BlockSettle Network Design

By default, the BlockSettle Terminal and the BlockSettle Signer run on the same machine. The BlockSettle Terminal simply runs the BlockSettle Signer in the background. The average user will never even be aware that the BlockSettle Signer exists. Everything will presented via a clean, inuitive GUI that gives users a powerful interface to the Bitcoin network. Such a setup is designed to make BlockSettle products usable immediately upon installation, all while providing excellent security. However, more advanced users, or users with a more advanced network setup, have the option of running BlockSettle products with even more security.

The BlockSettle Terminal manages external connections, such as the Matching Engine, Chat Server, ArmoryDB, Charting, and the BlockSettle Signer. The BlockSettle Terminal has no inherent need for access to full Bitcoin wallets or sign transactions. This enhances security. Any compromise of a machine with only the BlockSettle Terminal means the attacker has no inherent ability to spend coins or view wallet balances controlled by the user.

The BlockSettle Signer generates, holds, and operates all wallet materials and private keys. For the Terminal to register a wallet with ArmoryDB, the BlockSettle Terminal must be connected to a BlockSettle Signer as all wallets, full or watching-only, are kept within the BlockSettle Signer.

Binaries

The following binaries are used by the Terminal and Signer, and are titled as follows:

- **BlockSettle Terminal:**
  blocksettle_terminal *(Windows and Linux)*
  BlockSettle Terminal.app *(macOS)*
- **BlockSettle Signer:**
  blocksettle_signer *(Windows, Linux, and macOS)*
- **BlockSettle Signer (GUI):**
  bs_signer_gui *(Windows and Linux)*
  BlockSettle Signer GUI.app *(macOS)*

File Locations

Unless otherwise stated, all files used by the Terminal and Signer can be found in the following locations, which also serve as default directories:

- **Windows:**
  C:\Users\<username>\AppData\Roaming\blocksettle\
- **Linux:**
  /home/<username>/.local/share/blocksettle/
- **macOS:**
  /Users/<username>/Library/Application Support/blocksettle

BlockSettle

BlockSettle Signer Functionality

BlockSettle Signer Modes

The BlockSettle Signer can be launched in two different modes: *Headless or GUI*.

- **Headless Mode**
  The Signer is programmatically launched and may never be visible to the user except for when signing dialogs are spawned in front of the Terminal GUI. When the Signer is invoked by the Terminal, the Signer is invoked in Headless Mode.

- **GUI Mode**
  The Signer is manually controlled and launched by the user. All operations relating to private keys and transaction signing take place locally within the Signer GUI. When the Signer is run independently of the Terminal, GUI Mode is the default mode for the Signer.

In addition, the BlockSettle Signer and BlockSettle Terminal have three modes of communication: *Local, Remote, and Offline Signing.*

- **Local Communication**
  The Terminal and the Signer reside on the same machine. The Terminal is responsible for launching and closing the Signer. The Signer runs in the background. All wallet operations (e.g., creation, deletion, backups, encryption method, and password entry) take place via the Terminal. In other words, the user never has to directly interact with the Signer; the Terminal will transparently handle the Signer as needed. This is the default communication mode for both the Terminal and the Signer.

- **Remote Communication**
  The Signer is launched manually on any machine connected by any kind of local network to the machine running the Terminal, including a loopback interface on the same machine. When using remote communication, the Signer will sign any transactions sent by the Terminal; the Terminal will merely signal to the Signer that a transaction needs to be signed. The Signer will also be able to perform the same wallet operations as the Terminal (e.g., change wallet encryption, create wallet backups, etc.). It is highly recommended, but not required, that the Signer and the Terminal run on separate machines.

BlockSettle

BlockSettle Signer Functionality

- **Offline Signing**
  The Terminal is connected to a Signer which contains one (or more) Watching-Only wallets. No network connectivity exists between the Watching-Only wallets and their private keys. In this case, the signer connected to the Terminal is limited to holding the Watching-Only wallet and creating unsigned transactions which can be saved and exported via any means preferred by the user (e.g., USB drive or a burned CD) to the "air-gapped" computer containing the BlockSettle Signer which holds the private keys required to sign the transaction (the full wallet). Any media read by the computer running by the Signer should be checked for malware before the Signer computer accesses the media.

In all communication modes, the Signer can run in Headless or GUI Mode. However, with Offline Signing, the Signer holding the full wallet must always run in GUI Mode. The Signer's Headless Mode has limited abilities to sign transaction requests.

Running The BlockSettle Signer

Command-Line Options (including Headless Mode)

To launch the Signer, the user will launch the *blocksettle_signer* binary. The following command line options are available. Note that not all Signer functionality is available when in headless mode.

--log=<logfile>	Path and name of the Signer's log file. (Default: <AppData>/blocksettle/bs_signer.log)
--listen <ip address / hostname>	IP address or hostname listening to incoming connections. By default, the Signer listens on all network interfaces. (Default value: 0.0.0.0)
--port=<port>	Port number for incoming connections. (Default: 23456)
--dirwallets=<dir>	The Directory containing the full (non-watching-only) wallets which will be used for signing. (Default: The current directory.)
--auto_sign_spend_limit <XBTs>	The maximum number of bitcoins that may be used in auto-signing operations. The functionality applies only to SpotXBT and SpotCC transactions.
--sign=<filename>	The Signer will do nothing but sign transaction(s) in the specified file and exit on completion. The option implies that the Signer will be in Offline Communication mode. (Default: Not used.)

BlockSettle

Running The BlockSettle Signer

--mainnet	Run the Signer on the ”mainnet” Bitcoin network, where bitcoins have fiat value and should be used only as needed or desired. (Default: Active.)
--testnet	Run the Signer on the ”testnet3” Bitcoin network, where bitcoins have no fiat value. The network is intended for testing purposes. If mistakes occur and bitcoins are lost, no fiat value is lost. (Default: Not active.)
--guimode [lightgui|fullgui]	When the headless Signer binary spawns the Signer GUI, the GUI is opened in ”full” mode, displaying the entire GUI. When the Terminal connects to the Signer in Local Mode, the Signer GUI is spawned in a ”light” mode with no significant user interface. (Default: fullgui)
--help	Display the help text.

BlockSettle

Running The BlockSettle Signer

The following command line options are used by the Terminal and Signer in Local Mode. Please do not use these options manually. The options are described here only so that users understand what is happening.

--server_id_key <compressed public key>	In Local Mode, the headless Signer binary spawns the Signer GUI. The headless Signer passes its BIP 150 identity key (compressed secp256k1 key) to the GUI via this option. The key is expressed as a 33 byte hex string.
--terminal_id_key <compressed public key>	In Local Mode, the Terminal spawns the headless Signer binary. The Terminal passes its BIP 150 identity key (compressed secp256k1 key) to the Signer binary via this option. The key is expressed as a 33 byte hex string.

The user can start multiple Signer processes on the same host. The user must make sure each process uses different wallet directories, log files and listening sockets. Failure to separate the required items will cause unexpected behavior, or cause duplicate Signer instances to not start.

Note that command line options prevail over any GUI settings.

Running The BlockSettle Signer

GUI Operation

There are four primary tabs in the Signer's GUI Mode: *Dashboard, Settings, Auto-Sign, and Wallets.*

1 DASHBOARD

Sign Offline From File – When selected, a file containing an unsigned transaction may be chosen for signing. When loaded and signed, a new file containing a signed transaction will be generated and must be supplied to the Terminal for broadcasting onto the Bitcoin network.

Controls

- ”Online mode” toggle – A toggle button determines whether or not the Signer will maintain any network communication. It is possible to start in Local or Remote mode and then deactivate network communication.
- ”Auto sign” toggle – A toggle button that allows for the automatic signing of user-specified particular transaction requests.

Details

The ”Details” section is read-only, and does not allow the BlockSettle Signer configuration to be changed.

- Listen socket – Lists the IP address and TCP port where the BlcokSettle Signer is listening for incoming connections.
- Network type – Lists the Bitcoin network (Mainnet or Testnet) where the BlockSettle Signer is operating.
- Connection[s] – Lists the IP address(es) where clients have connected with the BlockSettle Signer.
- Transaction[s] signed – Lists the number of transactions signed by the BlockSettle Signer since the current instance of the Signer started running.
- Manual spend limit – The amount of coins that can be spent per transaction by the BlockSettle Signer.
- Auto-Sign spend limit – The maximum amount of bitcoins (XBT) that may be spent in an automatically signed SpotXBT/SpotCC transaction.
- Auto-Sign time limit – The length of time that Auto-Sign functionality will be enabled.

Running The BlockSettle Signer

2 SETTINGS

General Settings

- ”Online Mode” toggle – A toggle that allows the Signer to shut down its connection to any Terminals.
- ”Testnet” toggle – A toggle used to run the Signer on Bitcoin’s Testnet network. Testnet is a special Bitcoin network used for testing software in a special environment with bitcoins that have no fiat value.

Network Settings

- Signer Public Key – The user may copy (hex string) or export a binary file (33 bytes) of the Signer’s BIP 150 identity key.
- Listening IP address – The address where the Signer is listening. By default, the Signer listens on all available addresses. (Default: 0.0.0.0)
- Listening Port – The port where the Signer is listening. (Default: 23456)

Terminal Settings

- Terminal ID Key Authentication - Toggle which allows the user to disable the requirement that the Signer must import the Terminal ID Key prior to connecting. BlockSettle strongly discourages users from disabling this option as it will allow any Terminal to connect to your signer and have access to your wallets.
- Terminal ID Keys – The user may manage which BIP 150 identity keys of clients are whitelisted. The user will copy in a hex string of the key, along with an associated key name.

3 AUTO-SIGN

Controls

- Wallet selection – The wallet that is currently having its Auto-Sign settings modified.
- ”Auto Sign” toggle – The toggle that controls whether or not automatic signing of incoming SpotXBT or SpotCC transactions will occur.

Details

- XBT spend limit – Sets the maximum amount of bitcoins (XBT) that may be spent in an automatically signed SpotXBT/SpotCC transaction. (Default: Unlimited XBT)
- Time limit – The length of time that Auto-Sign functionality will be enabled. (Default: 1 hour)

Running The BlockSettle Signer

4 WALLETS

- Wallet List – Displays basic information about all wallets loaded into the Signer.
- New Wallet – A button that allows the user to create a new wallet or import a wallet (paper or digital) to be used by the Signer.
- Manage Encryption (Wallet selected only) – Manages a wallet's encryption. The user may change the wallet encryption method (password or Auth eID), change the wallet password, or add or delete devices tied to a particular Auth eID account.
(For more information on Auth eID, consult BlockSettle for appropriate documentation.)
- Delete Wallet (Wallet selected only) – Delete a selected wallet. The user has the option of making a backup (paper or digital) of the wallet's root private key (RPK) before deleting the wallet. The RPK may be used later to import the wallet into the Signer.
- Backup Private Key (Wallet selected only) – The user creates a backup (paper or digital) of the wallet's root private key (RPK). The RPK may be used later to import the wallet into the Signer.
- Export Watching-Only Wallet (Wallet selected only) – A watching-only (WO) wallet file is created. The file may be used to import the WO wallet into the Terminal.

In GUI mode, the Signer will always ask for wallet passwords or Auth eID authorizations in the Signer process, and not in the Terminal. All wallet decryption is performed locally in the Signer.

System Requirements

The following operating systems are supported.

- Ubuntu 16.04, 18.04, and any supported ”short-term” releases post-18.04.
- Windows 7, 8, 8.1, and 10.
- macOS 10.12 or higher.

The Signer has no special hardware requirements other than a 64-bit x86 processor and a proper GUI display if using the Signer in GUI mode. At least 2 GB of RAM is highly recommended.

BlockSettle

Appendix

BIP 150/151 Public/Private Key Information

The connection between the Signer and the Terminal is protected by the ZMQ messaging library. In particular, a cryptographic mechanism called BIP 150/151 is placed over the connection.
https://github.com/bitcoin/bips/blob/master/bip-0150.mediawiki and
https://github.com/bitcoin/bips/blob/master/bip-0151.mediawiki have more information.

Important points to note include:

- BIP 151 is applied to establish cryptographic keys to be used when communicating between the Terminal and the Signer. BIP 151 is applied first in order to establish a shared symmetric key (x_{sym}) used to encrypt and decrypt messages sent over the network. No keys are exposed to users. Other than x_{sym}, none of the keys used in BIP 151 are used once the BIP 151 handshake is complete.
- BIP 150 is applied after BIP 151. BIP 150 ensures that both parties are authorized to communicate with each other. The client and server each have a public (X_{ID}) and private (x_{ID}) identity key. Both keys are used as part of BIP 150. In addition, the Terminal and the Signer expose X_{ID} to the user. X_{ID} is required when using Remote Signer Mode. The Signer must add all X_{ID} keys for ”whitelisted” Terminals, and vice versa.
- Once the BIP 150 handshake is complete, the client and server may send data to each other.
- x_{sym} is automatically rotated every 10 minutes (600 seconds) according to the BIP 151 standard.
- Before BIP 151 is applied, server connections send X_{ID} to client connections. In other words, the headless Signer binary sends X_{ID} to the Terminal and the Signer GUI. The key is sent so that clients know which identity key will be used by the server.
- A special ”public mode verification” mode has been added to the code. When activated, the client verifies the server during BIP 150 but the signer doesn’t verify the client. The idea is to give servers an option to be more like TLS servers, where clients can verify a server but the opposite isn’t true.

Both standards rely on elliptic curve cryptography (the secp256k1 curve), the same cryptography that underpins Bitcoin’s cryptographic functionality. The difference is that Bitcoin uses public and private keys to sign and verify transactions. BIP 150 and 151 use public and private keys to establish a final, symmetric key to secure communication channels.

BlockSettle

Appendix

One problem that must be solved is how to determine which keys are authorized by each party. Both modes use different methods to establish authorized keys.

In Remote Mode, users may ”whitelist” particular keys; if the keys are used during the BIP 150 handshake, the handshake will be successful. The Terminal and the Signer GUI both display the key in their respective settings (hex string), and allow for the export of files containing X_{ID} in binary form (33 bytes). The respective settings also allow the user to import authorized X_{ID} keys, via import of files containing X_{ID} (binary form only) or as a hex string.

Local Mode presents a different challenge. There’s no way to whitelist keys beforehand. The solution is to rely on a combination of cookies and command line arguments. The combinations are as follows:

- *Terminal opens the headless Signer:* The terminal spawns the Signer with the ”terminal_id_key” command line argument. The Terminal will include a compressed secp256k1 identity key to be checked during the BIP 150 handshake. The Signer will whitelist only the key supplied in the argument. Once spawned, the Signer will generate a cookie (”signerServerID”) with its own compressed secp256k1 identity key. The Terminal will read the cookie and whitelist only the key that is read. The key will then be checked during the BIP 150 handshake. If both sides are satisfied and no other errors occur, the secured connection between the Terminal and Signer will be completed.
- *Headless Signer opens the Signer GUI:* The headless Signer spawns the Signer GUI with the ”signer_id_key” command line argument. The headless Signer will include a compressed secp256k1 identity key to be checked during the BIP 150 handshake. The Signer GUI will whitelist only the key supplied in the argument. Once spawned, the Signer GUIwill generate a cookie (”clientID”) with its own compressed secp256k1 identity key. The headless Signer will read the cookie and whitelist only the key that is read. The key will then be checked during the BIP 150 handshake. If both sides are satisfied and no other errors occur, the secured connection between the headless Signer and the Signer GUI will be completed.

The cookies are located in the same directory as all other BlockSettle data.