Aion Guide for Ledger Nano S application

Technical Details

Here are the technical details of Aion application for ledger Nano S.

HD Derivation Path

44’/425’

Key generation algorithm

Ed25519

Address mapping algorithm

Address is generated by getting 32-byte blake2b hash of 32-byte public key and then substituting the first byte with ‘0xa0’ hex character.

Signing algorithm

Eddsa with SHA512

Protocols supported

HID and U2F

Features

Aion application provides two features

a) Get public key and account details
b) Signing the transaction

Get public Key and account details

This feature takes HD derivation path as an input and returns 64-byte hex String where former 32 bytes is public key and latter is account address.

Input is strictly in this sequence:

1st Byte: Beginning of the message (‘0xeo’ for Aion)

2nd Byte: Signifies the message intention (02 for getting public key and account address)

3rd Byte: For future use (if address needs to be confirmed or just returned)

4th Byte: For future use (00 chain code)

5th Byte: Length of derivation path + 1

6th Byte: Length of derivation path / 4

7th Byte to End: Hex String of derivation path

Example:

Input (for 44’/425’/0’/0’/0’): e002010015058000002c800001a9800000008000000080000000

Output:
b808763388bc601f5138e310c0b80c0db1efc9f9fb107697c209fe1e2d698e96a0208c72fad2b444b7d7195bd0452c0f6c2f34cfa94a3691c1637da46430d196

where:

Public-key: b808763388bc601f5138e310c0b80c0db1efc9f9fb107697c209fe1e2d698e96

Account: a0208c72fad2b444b7d7195bd0452c0f6c2f34cfa94a3691c1637da46430d196

Sign the Transaction

This feature takes derivation path and RLP encoded transaction as an input and returns 64-byte signature as an output

Input is strictly in this sequence:

1st Byte: Beginning of the message (‘0xeo’ for Aion)

2nd Byte: Signifies the message intention (04 for signing transaction)

3rd Byte: For future use (Identify the beginning but not required now as max Tx size is 230 bytes)

4th Byte: For future use (00 chain code)

5th Byte: Length of derivation path + 1 + encoded transaction

6th Byte: Length of derivation path / 4

7th Byte till end of dongle path: Dongle path bytes

End: RLP encoded transaction bytes

Example:

Input (for 44’/425’/0’/0’/0’):
e0040000cf058000002c800001a9800000008000000080000000f8b800a0a0185ef98ac4841900b49ad9b432af2db7235e09ec3755e5ee36e9c4947007dd89056bc75e2d63100000b87ca0f2daa8de60d0e911fb468492242d60e15757408aff2902a0f2daa8de60d0e911fb468492242d604e1e11ec6f142bfee15757408aff2902a0f2daa8de60d0e911fb468492242d604e1e11ec6f142bfee15757408aff2902a0f2daa8de60d0e911fb468492242d604e1e11ec6f14a0f2daa0f2daa0f2daa0f2daa0f28332298e8252088502540be40001

Output (Signature)**:**
1a4879c0de1c9fbabcfd0d4c0f792d9feea831d4223a2425bd6a3d360913c2d65fdda4efafa2f6eaf624e4be7cbb9e42e1c13b6415b4d37d88dade8f11224800

Instructions for installing Aion on Ledger Nano S:

- Git clone dev branch of https://github.com/aionnetwork/aion_ledger/
- Configure BOLOS and Nano S SDK as mentioned here: https://ledger.readthedocs.io/en/latest/userspace/getting_started.html
- Configure python virtual environment (optional to run python test cases): https://ledger.readthedocs.io/en/0/nanos/setup.html
- App Installation
    - Connect Ledger Nano S via USB
    - Change directory of Aion ledger repository
    - Execute “make load BOLOS_SDK={SDK_LOCATION} BOLOS_ENV={ENV_LOCATION}”
- App Deletion
    - Connect Ledger Nano S via USB
    - Change directory of Aion ledger repository
    - Execute “make delete BOLOS_SDK={SDK_LOCATION} BOLOS_ENV={ENV_LOCATION}”

Testcases:

There are testcases in python and JavaScript for the above-mentioned two features using HID and U2F protocols.

Python Testcases:

Following three test cases can be run over HID by executing execute_testcases.py location at aion_ledger/test/python-testcases

1. **Verify public key and account** (test_get_public_key_and_address): This testcase verifies whether the public key and account address received from Ledger Nano S is consistent with the intended seed key and derivation path. Please ensure you enter the correct Nano S seed key and HD derivation path in the test case.
2. **Verify signature with contract data** (test_verify_signature_with_data): This testcase generates RLP encoded transaction and verifies the signature received from Nano S. Please ensure correct RLP parameters and correct HD derivation path. Also ensure that the ‘Contract Data’ is set to ‘yes’ in Aion application.
3. **Verify signature without contract data** (test_verify_signature_without_data): This testcase generates RLP encoded transaction and verifies the signature received from Nano S. Please ensure correct RLP parameters and correct HD derivation path.

Javascript Testcases:

Javascript testcases can be run over HID or U2F. These testcases are located at aion_ledger/test/javascript-testcases

1. **Testcases over HID**: Execute ‘npm run test:node-hid’. The parameters can be modified in node-hid.js .
2. **Testcases over U2F**: Execute ‘npm run test:u2f’. The parameters can be modified in u2f.js .

HID Drivers

These drivers are written for Aion Desktop Wallet for communicating with Ledger Nano S over HID. Located here

- **Ubuntu Driver Usage Example**:
  ./Aion-HID --param=e002010015058000002c800001a9800000008000000080000000

- **Windows Driver Usage Example**:
  npm run get:aion-hid e002010015058000002c800001a9800000008000000080000001