CESS

Cumulus Encrypted Storage System

CESS Course Week 3 - Episode 5

dApp Development #2

CESS Official Website

Table of Content

Smart Contract Development: Solidity

CESS
Cumulus Encrypted Storage System

- Developer Familiarity
- Widely Supported Toolchains
- Direct Code Migration

Ink! vs. Solidity Development

CESS
Cumulus Encrypted Storage System

	ink!	Solidity
Virtual Machine	Any Wasm VM	EVM
Encoding	Wasm	EVM bytecode
Language	Rust	Solidity
Tooling	Rust toolings	Solidity toolings
Overflow Protection	Enabled by default	Yes
Storage Entries	Variable	256 bits (**2^{256}** entries)
Has Metadata?	Yes	Yes

src: https://use.ink/ink-vs-solidity/

Other Commonly Used Libraries

CESS
Cumulus Encrypted Storage System

Name	Type	Description
Polkadot SDK	Substrate	An umbrella project encompassing three sub-projects: Substrate, Cumulus, and Polkadot.
Polkadot-js API	Substrate	Javascript/Typescript library to interact with Substrate-based blockchains, with utility libs on cryptographic functions.
ether.js	EVM Smart Contract	Library to interact with EVM-compatible smart contracts.
wagmi	EVM Smart Contract / React hook	React hook for EVM-compatible smart contract.

Substrate and EVM Addresses

CESS
Cumulus Encrypted Storage System

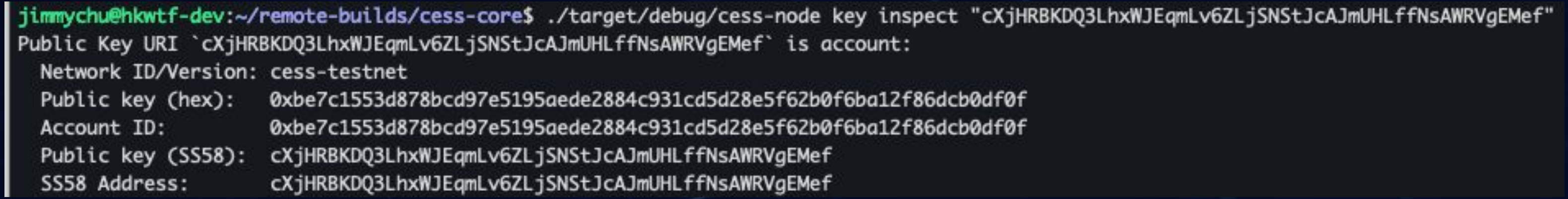
CESS Address:

- SS58 address: cXjHRBKDQ3LhxWJEqmLv6ZLjSNStJcAJmUHLffNsAWRVgEMef
- Decoded using Base58 encoder/decoder (36 bytes):

Base58 encoding

50acbe7c1553d878bcd97e5195aede2884c931cd5d28e5f62b0f6ba12f86dcb0df0f85d0

- Pub. key (32 bytes): **be7c1553d878bcd97e5195aede2884c931cd5d28e5f62b0f6ba12f86dcb0df0f**

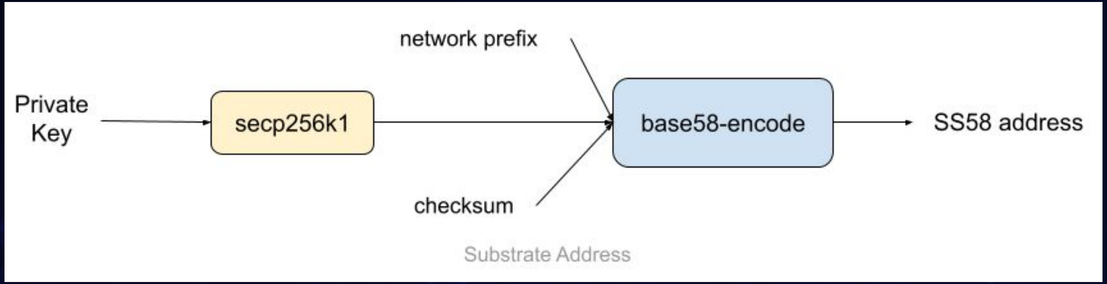


```
jimmychu@hkwtf-dev:~/remote-builds/cess-core$ ./target/debug/cess-node key inspect "cXjHRBKDQ3LhxWJEqmLv6ZLjSNStJcAJmUHLffNsAWRVgEMef"
Public Key URI `cXjHRBKDQ3LhxWJEqmLv6ZLjSNStJcAJmUHLffNsAWRVgEMef` is account:
  Network ID/Version: cess-testnet
  Public key (hex):   0xbe7c1553d878bcd97e5195aede2884c931cd5d28e5f62b0f6ba12f86dcb0df0f
  Account ID:         0xbe7c1553d878bcd97e5195aede2884c931cd5d28e5f62b0f6ba12f86dcb0df0f
  Public key (SS58):  cXjHRBKDQ3LhxWJEqmLv6ZLjSNStJcAJmUHLffNsAWRVgEMef
  SS58 Address:       cXjHRBKDQ3LhxWJEqmLv6ZLjSNStJcAJmUHLffNsAWRVgEMef
```

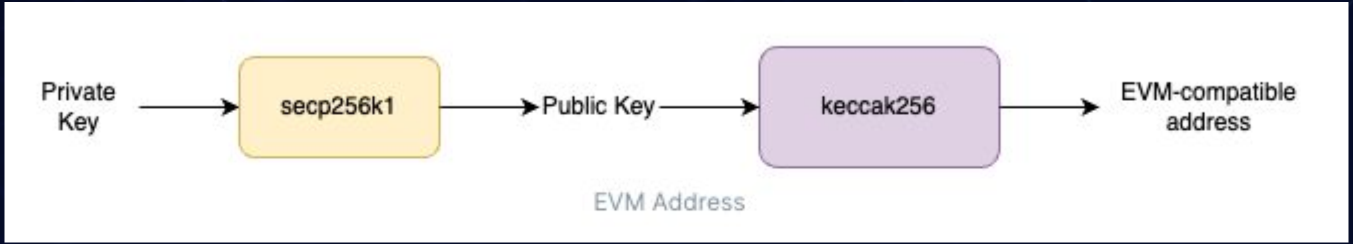
- SS58 address: **base58-encode(network prefix, public key, checksum)**
- Network prefix: **0x50ac** (decimal: 11330, due to additional conversion)
- Checksum: **0x85d0**

Substrate and EVM Addresses

CESS
Cumulus Encrypted Storage System

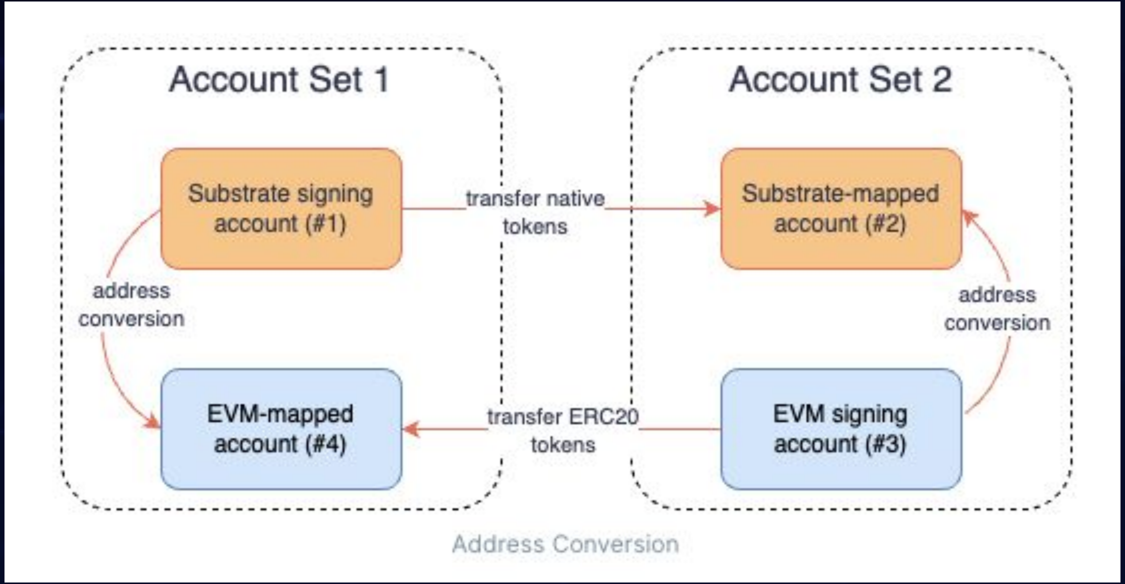


EVM address (H160 addr): `0x9440Abf16a3319E633DA6835d90470ed029D7c0B (20 bytes)`



Substrate and EVM Addresses

We need **TWO** account sets & an address conversion tool:



Demo: Transfer from Substrate Signing Acct to EVM Signing Acct

CESS
Cumulus Encrypted Storage System

- Polkadot Portal & Metamask installation
- The address conversion tool
- Convert EVM signing addr. to Substrate-mapped addr.
- Transfer from Substrate signing addr to Substrate-mapped addr
- Check acct balance in Metamask

Demo: Transfer from EVM Signing Acct to Substrate Signing Acct

CESS

Cumulus Encrypted Storage System

- The address conversion tool
- Convert Substrate signing addr to EVM-mapped addr
- Transfer from EVM signing addr to EVM-mapped addr
- Withdraw in Substrate signing addr with an on-chain transaction
- Check acct balance in Polkadot Portal

Demo: Deploy a Contract on CESS Testnet

CESS
Cumulus Encrypted Storage System

- Flipper.sol in CESS example
- Configure **`hardhat.config.ts`** for CESS Testnet deployment
- Deploy with **`hardhat deploy`**
- Interact with the contract using Remix

Demo: Proof of Existence (PoE) Tutorial

CESS
Cumulus Encrypted Storage System

- src: contract, front end
- PoE: what it does
- Demo: PoE interaction
- PoE: smart contract side (Solidity)
- PoE: front end with wagmi
- ref: CESS documentation

CESS

Cumulus Encrypted Storage System

End