Digital Contract CLI

Introduction

This project is a Command-Line Interface (CLI) tool for managing and signing digital contracts. The system enables users to create, load, sign, verify, import, and export digital contracts using the PrivateSky platform, specifically leveraging the OpenDSU framework for secure and decentralized data handling.
It supports both text and PDF contract formats, ensuring flexibility for different use cases. The contracts are stored and retrieved using Self-Sovereign Identities (SSIs) and Data Sharing Units (DSUs), central to the PrivateSky model.

Why Use PrivateSky (OpenDSU)?

The decision to use PrivateSky technologies, particularly OpenDSU, comes from the following advantages:

- Each contract is stored in its own DSU, allowing users full control over their data with cryptographic protection.
- OpenDSU provides built-in encryption, integrity verification, and digital signing capabilities that match the project's need for contract authenticity.
- SSIs enable easy sharing and identity linkage across systems without centralized control.
- The ability to begin and commit batch operations guarantees consistency even in asynchronous or failure-prone environments.
- The framework's APIs are modular and flexible, facilitating clean, maintainable code.

Used OpenDSU Technologies

This project leverages several key technologies and modules from the OpenDSU framework to ensure secure, decentralized, and modular operations:

- SeedSSI - Used to generate unique identifiers for DSUs. These are cryptographic identifiers that support self-sovereignty and enable secure identity management.
- KeySSI - Used for securely identifying and accessing DSUs. KeySSIs act as pointers to DSUs while embedding access rights and encryption parameters.
- DSU (Data Sharing Unit) - Used for storing contract data. Each contract is stored in its own DSU, encapsulating all related files and metadata.
- Batch Operations - Utilized to ensure atomicity during write operations on DSUs. This guarantees consistency and avoids partial writes.
- Resolver - Used for creating, loading, and managing DSUs via SeedSSI/KeySSI. It handles the linking between identifiers and actual data units.
- Crypto Module - Used for signing and verifying digital signatures, ensuring authenticity and integrity of contracts.

Design & Implementation

The project is structured as a Node.js CLI application with modular command handling:

Core Modules

- main.js: Initializes the CLI application, registers commands, and parses user input.
- dsuHelper.js: Contains core utility functions for interacting with DSUs, including loading, creating, signing, and verifying.

Features

- **Create Contract:** Allows users to create contracts either by typing text or uploading a PDF. The file is then written to a newly created DSU.
- **Sign Contract:** Digitally signs the contract using a generated SeedSSI and stores the signature in the DSU.
- **Verify Contract:** Retrieves the signature and verifies it against the contract contents to ensure integrity and authenticity.
- **Export/Import Contract:** Enables sharing contracts by exporting their content and metadata to a JSON file and re-importing into a new DSU.
- **Load Contract:** Downloads and saves a contract locally from a given KeySSI.

Usage

Help

Running the CLI application without any arguments displays the general help menu, listing all available commands and options:

```
$ node main.js
Usage: contract-cli [options] [command]

CLI for managing and signing digital contracts using OpenDSU

Options:
  -V, --version                        output the version number
  -h, --help                           display help for command

Commands:
  sign-contract <keySSI>               Digitally sign a contract stored in a DSU
                                       using a generated SeedSSI
  create-contract [options]            Create a new digital contract as text or
                                       PDF
  load-contract [options] <keySSI>     Load a contract from a DSU
  verify-contract <keySSI>             Verify the contract's signature and
                                       integrity
  export-contract [options] <keySSI>   Export contract DSU content and signature
                                       for sharing
  import-contract <file>               Import a contract from a shared export
                                       file
  help [command]                       display help for command
```

Each command also provides its own help documentation. To view help for a specific command, use the --help flag:

```
$ node main.js create-contract --help
Usage: contract-cli create-contract [options]

Create a new digital contract as text or PDF

Options:
  --pdf <path>  Provide path to a PDF file to store as contract
  -h, --help    display help for command
```

Commands

- create-contract [--pdf]

```
$ node main.js create-contract
Enter contract title: My Contract
Enter contract content: This is my contract
Contract created and stored.
Contract KeySSI:
BBudGH6ySHG6GUHN8ogNrTWcBVXEdDHbmwWbDHHta4KSKkKn2fmWgdGCkMGdRschPcbnJuUuGg9TjQen1J5U4uK5H
```

```
$ node main.js create-contract --pdf ../test.pdf
PDF contract stored.
Contract KeySSI:
BBudGH6ySHG6GUHN8ogNrTWd6C3owrewGiamCmWAP1PAcQgjMR5GnpqHCc7maF9dw7gjymtKhWMWnAF2Sgs22AnHd
```

- sign-contract <keySSI>

```
$ node main.js sign-contract BBudGH6ySHG6GUHN8ogNrTWd6C3owrewGiamCmWAP1PAcQgjMR5GnpqHCc7maF9dw7
gjymtKhWMWnAF2Sgs22AnHd
Signing PDF contract...
Contract signed by BBudGH6ySHG6GUHN8ogNrTWdAjqqRsWqv3e3bz72Wk6wesZiYpzcQ1gb3kvs9DakDE8tt1hP4HXe
7BCPGixMFgDbM
```

- verify-contract <keySSI>

```
$ node main.js verify-contract BBudGH6ySHG6GUHN8ogNrTWd6C3owrewGiamCmWAP1PAcQgjMR5GnpqHCc7maF9d
w7gjymtKhWMWnAF2Sgs22AnHd
Loaded PDF contract for verification.
Signature is valid.
Signed by: BBudGH6ySHG6GUHN8ogNrTWdAjqqRsWqv3e3bz72Wk6wesZiYpzcQ1gb3kvs9DakDE8tt1hP4HXe7BCPGixM
FgDbM
Date: 2025-05-19T19:46:11.203Z
```

- export-contract [--file] <keySSI>

```
$ node main.js export-contract BBudGH6ySHG6GUHN8ogNrTWd6C3owrewGiamCmWAP1PAcQgjMR5GnpqHCc7maF9d
w7gjymtKhWMWnAF2Sgs22AnHd
Contract exported to: contract_export.json
```

```
$ node main.js export-contract --file myContract.json BBudGH6ySHG6GUHN8ogNrTWd6C3owrewGiamCmWAP
1PAcQgjMR5GnpqHCc7maF9dw7gjymtKhWMWnAF2Sgs22AnHd
Contract exported to: myContract.json
```

- import-contract <file.json>

```
$ node main.js import-contract myContract.json
Signature restored.
Contract imported successfully.
New KeySSI: BBudGH6ySHG6GUHN8ogNrTWdK4HxaY98HHMiacmFxdegqJnc7Ckgimz8DuK5h7aTDDgwwA5ZaLwEMhvnZbW
cNHjLB
```

- load-contract [--name] <keySSI>

```
$ node main.js load-contract BBudGH6ySHG6GUHN8ogNrTWd6C3owrewGiamCmWAP1PAcQgjMR5GnpqHCc7maF9dw7
gjymtKhWMWnAF2Sgs22AnHd
PDF contract saved as: E:\Desktop\Facultate\Master\AN1_SEM2\PCD\Tema3-4\tutorial-workspace\digi
tal-contract-cli\contracts\downloaded_contract.pdf
```

```
$ node main.js load-contract --name importedContract BBudGH6ySHG6GUHN8ogNrTWdK4HxaY98HHMiacmFxd
egqJnc7Ckgimz8DuK5h7aTDDgwwA5ZaLwEMhvnZbWcNHjLB
PDF contract saved as: E:\Desktop\Facultate\Master\AN1_SEM2\PCD\Tema3-4\tutorial-workspace\digi
tal-contract-cli\contracts\importedContract.pdf
```