SDES with CBC encryption

SDES with CBC encryption implementation.

Build and compile

With Makefile

Just execute make in order to compile the source code. The output files will be inside `/build`

```
make
```

Manually

```
g++ ./src/sdes.cpp -o ./build/sdes -std=c++17
```

Usage

Change to the `build` folder to a easier usage:

```
cd build/
```

This project has its own `Usage` Linux like, to view this:

```
./sdes --help
```

Encrypting / Decrypting

> **NOTE**: Sample JSON file already exist

To encrypt, just use `--encrypt` option, with a JSON configuration file as argument.

```
./sdes --encrypt [JSON file]
```

To decrypt, just use `--decrypt` option, with a JSON configuration file as argument.

```
./sdes --decrypt [JSON file]
```

View JSON file help on console

To view a JSON file example:

```
./sdes --json example
```

To view the JSON file description:

```
./sdes --json desc
```

To view all above information:

```
./sdes --json all
```

JSON file

JSON file example:

```
{
    "iv": " 01100101",
    "encrypt": {
        "inputFile": "message.txt",
        "outputFile": "message.sdes"
    },
    "decrypt": {
        "inputFile": "message.sdes",
        "outputFile": "message.txt"
    },
    "keyConfig": {
        "random": false,
        "key": "1010000010",
        "p10": [2, 4, 1, 6, 3, 9, 0, 8, 7, 5],
        "p8": [5, 2, 6, 3, 7, 4, 9, 8]
    },
    "cryptConfig": {
        "initialPermutation": [1, 5, 2, 0, 3, 7, 4, 6],
        "expansion": [3, 0, 1, 2, 1, 2, 3, 0],
        "p4": [1, 3, 2, 0],
        "s0": [
            [1, 0, 3, 2],
            [3, 2, 1, 0],
            [0, 2, 1, 3],
            [3, 1, 3, 2]
        ],
        "s1": [
            [0, 1, 2, 3],
            [2, 0, 1, 3],
            [3, 0, 1, 0],
            [2, 1, 0, 3]
        ]
    }
}
```

NOTE: All the permutations must begin in 0.

`iv`: An string with the binary representation of the initialization vector. Used for decrypt.

`encrypt` and `decrypt`: Define the files to work with. - `inputFile`: Set the input file, path relative to execution environment. - `outputFile`: Set the output file, path relative to execution environment.

`keyConfig`: Define the key config. - `random`: Define if the program will use random mode key for encryption, i.e., will generate a random key. Its absence represents falsehood. - `key`: Binary form of the key. Necesary for decryption. - `p10`: An array of length 10, represents the 10 permutation. · `p8`: An array of length 8, represents the compression permutation.

`cryptConfig`: The main process config. - `initialPermutation`: An array of length 8. Its the initial permutation. - `expansion`: An array of length 8, it's the expansion permutation. - `p4` An array of length 4, It's the 4 permutation. - `s0`: An array of length 4, with 4 arrays of length 4 with the integer reoresentation of the s0 box's values. - `s1`: An array of length 4, with 4 arrays of length 4 with the integer reoresentation of the s1 box's values.