todo_project_name

Jakub Kaczor, Wojciech Michalczuk, Łukasz Pawlak

Jan 30, 2023

CONTENTS:

CHAPTER

ONE

THE GENERAL IDEA

CHAPTER

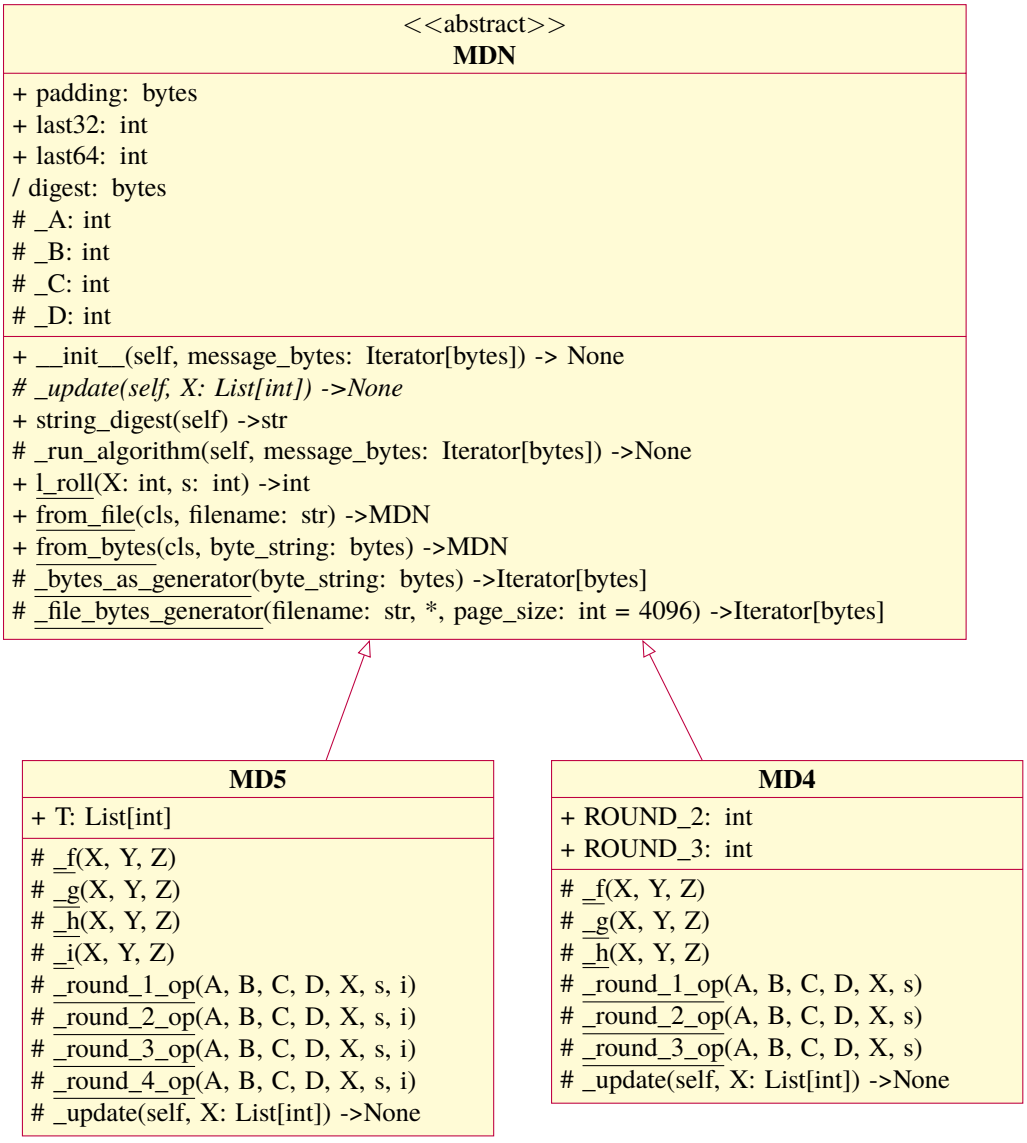
TWO

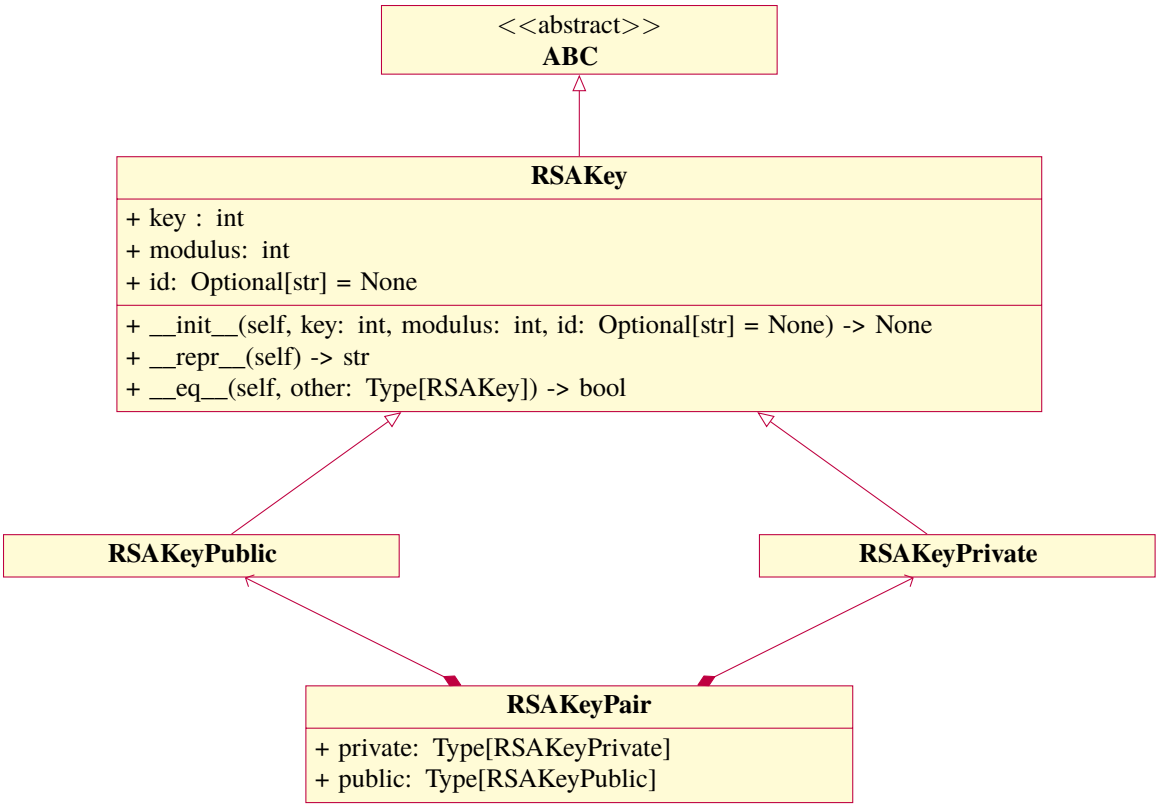
HOW TO USE THE PROGRAM

CHAPTER

THREE

CODE STRUCTURE




<<abstract>>
ABC
RSAKey
+ key : int
+ modulus: int
+ id: Optional[str] = None
+ __init__(self, key: int, modulus: int, id: Optional[str] = None) -> None
+ __repr__(self) -> str
+ __eq__(self, other: Type[RSAKey]) -> bool
RSAKeyPublic
RSAKeyPrivate
RSAKeyPair
+ private: Type[RSAKeyPrivate]
+ public: Type[RSAKeyPublic]

CHAPTER

FOUR

TODO_PROJECT_NAME PACKAGE

4.1 todo_project_name.core module

todo_project_name.core.**md4_string**(*message: str*) → str

Returns md4 digest of given string encoded as UTF-8 byte strings.

4.1.1 Parameters

message : string whose hash is to be computed.

todo_project_name.core.**md5_string**(*message: str*) → str

Returns md5 digest of given string encoded as UTF-8 byte strings.

4.1.2 Parameters

message : string whose hash is to be computed.

4.2 todo_project_name.find_prime module

todo_project_name.find_prime.**find_prime**(*n: int*) → int

Return *n*-bit probable prime.

4.2.1 Parameters

n : number of bits, must be greater than 1,

because otherwise such a prime doesn't exist.

todo_project_name.find_prime.**is_probable_prime**(*candidate: int*) → bool

Check if *candidate* is a probable prime.

4.2.2 Notes

This function uses Rabin-Miller test under the hood.

4.3 todo_project_name.md4 module

class todo_project_name.md4.**MD4**(*message_bytes: Iterator[bytes]*)

Bases: *MDN*

Class computing MD4 message digest. Works for little-endian architecture.

It is recommended to use methods *MD4.from_bytes* or *MD4.from_file* to create new objects.

To get message digest as *str* use *string_digest* method. To get message digest as *bytes* read *digest* property.

ROUND_2 = 1518500249

ROUND_3 = 1859775393

4.4 todo_project_name.md5 module

class todo_project_name.md5.**MD5**(*message_bytes: Iterator[bytes]*)

Bases: *MDN*

Class computing MD5 message digest. Works for little-endian architecture.

It is recommended to use methods *MD5.from_bytes* or *MD5.from_file* to create new objects.

To get message digest as *str* use *string_digest* method. To get message digest as *bytes* read *digest* property.

4.5 todo_project_name.mdn module

class todo_project_name.mdn.**MDN**(*message_bytes: Iterator[bytes]*)

Bases: ABC

Superclass of MD4 and MD5. Works for little-endian architecture.

property digest

The message digest as bytes.

classmethod from_bytes(*byte_string: bytes*) → *MDN*

This function serves as constructor, which allows to compute hash of *bytes*.

4.5.1 Parameters

byte_string : message whose digest is to be computed.

classmethod from_file(*filename: str*) → *MDN*

This function serves as constructor, which allows to compute hash of file under given path.

4.5.2 Parameters

filename : path to existing file whose digest is to be computed.

static l_roll(*X: int*, *s: int*) → int

Roll (rotate) bits of 32-bit unsigned integer *s* positions to the left.

4.5.3 Parameters

X: integer to be rolled. Its binary representation cannot exceed 32 bits.

s: number of digits to roll. Must be integer in [0, 32].

last32 = 4294967295

last64 = 18446744073709551615

string_digest() → str

Returns string representation of message digest.

4.6 todo_project_name.rsa module

class todo_project_name.rsa.**RSAKey**(*key: int*, *modulus: int*, *id: Optional[str] = None*)

Bases: ABC

class todo_project_name.rsa.**RSAKeyPair**(*public:* todo_project_name.rsa.RSAKeyPublic, *private:* todo_project_name.rsa.RSAKeyPrivate)

Bases: object

private: *RSAKeyPrivate*

public: *RSAKeyPublic*

class todo_project_name.rsa.**RSAKeyPrivate**(*key: int*, *modulus: int*, *id: Optional[str] = None*)

Bases: *RSAKey*

class todo_project_name.rsa.**RSAKeyPublic**(*key: int*, *modulus: int*, *id: Optional[str] = None*)

Bases: *RSAKey*

todo_project_name.rsa.**read_key**(*path: Path*, *key_type: Type[RSAKeyVar]*) → RSAKeyVar

Read RSA key from the file.

todo_project_name.rsa.**rsa_key_gen**(*N: int*) → *RSAKeyPair*

Generate RSA key pair.

Takes number *N* and returns RSAKeyPair with (2 * N)-bit modulus.

4.6.1 Parameters

N : determines the strength of the protocol.

todo_project_name.rsa.**rsa_sign**(*message: str*, *key: ~todo_project_name.rsa.RSAKeyPrivate*, *algorithm: ~typing.Type[~typing.Union[~todo_project_name.md4.MD4, ~todo_project_name.md5.MD5]] = <class 'todo_project_name.md4.MD4'>*) → str

Function returns a digital singnature based on the RSA protocol.

4.6.2 Parameters

message : string message to be singed

key : RSA private key

algorithm : hash method. Default: MD4. Available algorithms: MD4, MD5.

todo_project_name.rsa.**rsa_sign_file**(*filename: str*, *key: ~todo_project_name.rsa.RSAKeyPrivate*, *algorithm: ~typing.Type[~typing.Union[~todo_project_name.md4.MD4, ~todo_project_name.md5.MD5]] = <class 'todo_project_name.md4.MD4'>*) → str

Function returns a digital singnature based on the RSA protocol.

4.6.3 Parameters

filename : path to existing file to sign

key : RSA private key

algorithm : hash method. Default: MD4. Available algorithms: MD4, MD5.

todo_project_name.rsa.**rsa_verify**(*message: str*, *signature: str*, *key: ~todo_project_name.rsa.RSAKeyPublic*, *algorithm: ~typing.Type[~typing.Union[~todo_project_name.md4.MD4, ~todo_project_name.md5.MD5]] = <class 'todo_project_name.md4.MD4'>*)

Function verifies digital singnature of a message basing on the RSA protocol. It compares decoded signature with hashed message and returns True if they are the same, otherwise False.

4.6.4 Parameters

message : string message

signature : signature for verification

key : RSA public key

algorithm : hash algorithm. Default: MD4. Available algorithms: MD4, MD5.

todo_project_name.rsa.**rsa_verify_file**(*filename: str*, *signature: str*, *key: ~todo_project_name.rsa.RSAKeyPublic*, *algorithm: ~typing.Type[~typing.Union[~todo_project_name.md4.MD4, ~todo_project_name.md5.MD5]] = <class 'todo_project_name.md4.MD4'>*)

Function verifies digital singnature of a message basing on the RSA protocol. It compares decoded signature with hashed message and returns True if they are the same, otherwise False.

4.6.5 Parameters

filename : path to file against which signature is being checked

signature : signature for verification

key : RSA public key

algorithm : hash algorithm. Default: MD4. Available algorithms: MD4, MD5.

todo_project_name.rsa.**save_key**(*key:* RSAKey, *path: Path*) → Path

Save RSA key to the file.

PYTHON MODULE INDEX

t