Encryption Dokumentáció

Ujhelyi Bence

1. fejezet

Hierarchikus mutató

1.1. Osztályhierarchia

Majdnem (de nem teljesen) betűrendbe szedett leszármazási lista:

2. fejezet

Osztálymutató

2.1. Osztálylista

Az összes osztály, struktúra, unió és interfész listája rövid leírásokkal:

3. fejezet

Fájlmutató

3.1. Fájllista

Az összes dokumentált fájl listája rövid leírásokkal:

4. fejezet

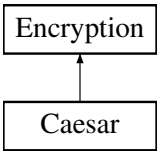
Osztályok dokumentációja

4.1. Caesar osztályreferencia

Caesar osztály.

`#include <Caesar.hpp>`

A Caesar osztály származási diagramja:



Publikus tagfüggvények

- **Caesar** (int shift)
- std::string encrypt (const std::string &text) const override

  encrypt függvény
- std::string decrypt (const std::string &secrettext) const override

  decrypt függvény
- std::string get_public_key () const override

  get_public_key függvény
- std::string get_private_key () const override

  get_private_key függvény

Publikus tagfüggvények a(z) Encryption osztályból származnak

- virtual ~Encryption ()=default

  Destruktor.
- virtual std::string encrypt (const std::string &plaintext) const =0

  encrypt függvény.
- virtual std::string decrypt (const std::string &ciphertext) const =0

  decrypt függvény.
- virtual std::string get_public_key () const =0

  get_public_key() függvény.
- virtual std::string get_private_key () const =0

  get_private_key() függvény .

4.1.1. Részletes leírás

Caesar osztály.

4.1.2. Tagfüggvények dokumentációja

4.1.2.1. decrypt()

```
std::string Caesar::decrypt (
            const std::string & secrettext ) const  [override], [virtual]
```

decrypt függvény

Paraméterek

secrettext	Titkos szöveg

Visszatérési érték

Visszafejtett szöveg

Visszafejti a titkos szöveget

Megvalósítja a következőket: Encryption.

4.1.2.2. encrypt()

```
std::string Caesar::encrypt (
            const std::string & text ) const  [override], [virtual]
```

encrypt függvény

Paraméterek

text	Titkosítandó szöveg

Visszatérési érték

Titkosított szöveg Titkosítja a beérkező sztringet és visszaadja a titkosított sztringet;

Megvalósítja a következőket: Encryption.

4.1.2.3. get_private_key()

```
std::string Caesar::get_private_key ( ) const  [override], [virtual]
```

get_private_key függvény

A kompatibilitás miatt muszáj inicializálni a függvént, még akkor is ha nem ad vissza semmit

Megvalósítja a következőket: Encryption.

4.1.2.4. get_public_key()

```
std::string Caesar::get_public_key ( ) const  [override], [virtual]
```

get_public_key függvény

A kompatibilitás miatt muszáj inicializálni a függvént, még akkor is ha nem ad vissza semmit

Megvalósítja a következőket: Encryption.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

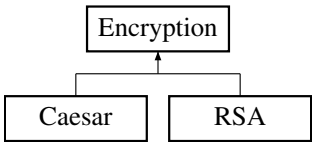
- Caesar.hpp
- Caesar.cpp

4.2. Encryption osztályreferencia

Encryption osztály.

```
#include <Encryption.hpp>
```

Az Encryption osztály származási diagramja:



Publikus tagfüggvények

- virtual ~Encryption ()=default

  Destruktor.
- virtual std::string encrypt (const std::string &plaintext) const =0

  encrypt függvény.
- virtual std::string decrypt (const std::string &ciphertext) const =0

  decrypt függvény.
- virtual std::string get_public_key () const =0

  get_public_key() függvény.
- virtual std::string get_private_key () const =0

  get_private_key() függvény .

4.2.1. Részletes leírás

Encryption osztály.

Absztrakt titkosító osztály.

4.2.2. Konstruktorok és destruktorok dokumentációja

4.2.2.1. ~Encryption()

```
virtual Encryption::~Encryption ( )  [virtual], [default]
```

Destruktor.

Virtuális destruktor.

4.2.3. Tagfüggvények dokumentációja

4.2.3.1. decrypt()

```
virtual std::string Encryption::decrypt (
            const std::string & ciphertext ) const  [pure virtual]
```

decrypt függvény.

Titkosítás feloldásáért felelős függvény.

Megvalósítják a következők: Caesar és RSA.

4.2.3.2. encrypt()

```
virtual std::string Encryption::encrypt (
            const std::string & plaintext ) const  [pure virtual]
```

encrypt függvény.

Titkosító függvény.

Megvalósítják a következők: RSA és Caesar.

4.2.3.3. get_private_key()

```
virtual std::string Encryption::get_private_key ( ) const  [pure virtual]
```

get_private_key() függvény .

RSA-hoz a privát kulcs lekérdezése.

Megvalósítják a következők: Caesar és RSA.

4.2.3.4. get_public_key()

```
virtual std::string Encryption::get_public_key ( ) const  [pure virtual]
```

get_public_key() függvény.

RSA-hoz a nyilvános kulcs lekérdezése.

Megvalósítják a következők: Caesar és RSA.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

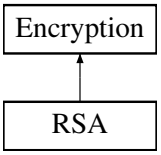
- Encryption.hpp

4.3. RSA osztályreferencia

RSA osztály.

```
#include <RSA.hpp>
```

A RSA osztály származási diagramja:



Publikus tagfüggvények

- RSA ()
  Érték nélküli konstruktor.
- std::string encrypt (const std::string &eredeti) const override
  encrypt függvény
- std::string decrypt (const std::string &titkos) const override
  decrypt függvény
- std::string get_public_key () const override
  get_public_key függvény
- std::string get_private_key () const override
  get_private_key függvény

Publikus tagfüggvények a(z) Encryption osztályból származnak

- virtual ~Encryption ()=default
  Destruktor.
- virtual std::string encrypt (const std::string &plaintext) const =0
  encrypt függvény.
- virtual std::string decrypt (const std::string &ciphertext) const =0
  decrypt függvény.
- virtual std::string get_public_key () const =0
  get_public_key() függvény.
- virtual std::string get_private_key () const =0
  get_private_key() függvény .

4.3.1. Részletes leírás

RSA osztály.

4.3.2. Konstruktorok és destruktorok dokumentációja

4.3.2.1. RSA()

```
RSA::RSA ( )
```

Érték nélküli konstruktor.

1. Létrehoz két véletlenszerű prímszámot, p-t és q-t. A generateRandomPrime() függvényt használja ezek generálására. A p a [10000, 20000] tartományban generált prímszám lesz, míg a q a [20000, 30000] tartományban generált prímszám lesz.
2. Kiszámítja a pí (totiense) értékét, amely a (p - 1) $*$ (q - 1) eredménye lesz.
3. Beállítja a nyilvános kulcsot (publicKey) a 65537 értékre, amely egy gyakran használt nyilvános relatív prím.
4. Iteratív módon növeli az e értékét egészen addig, amíg nem talál olyan értéket, amely relatív prím pí-vel. Ehhez a gcd(e, phi) függvényt használja, amely az e és pí legnagyobb közös osztóját adja vissza. Amint talál egy ilyen értéket, a ciklus megszakad.
5. Kiszámítja a privát kulcsot (privateKey) a publicKey moduláris inverzével pí modulo szerint. Ehhez a modularInverse() függvényt használja.

Ez a konstruktor tehát egy új RSA objektumot hoz létre és inicializálja a nyilvános és privát kulcsokat a fenti lépések szerint.

4.3.3. Tagfüggvények dokumentációja

4.3.3.1. decrypt()

```
std::string RSA::decrypt (
            const std::string & titkos ) const  [override], [virtual]
```

decrypt függvény

Paraméterek

titkos	A visszafejtendő sztring

Visszatérési érték

Eredeti üzenet

Visszafejti a titkosított sztringet.

1. Létrehoz egy üres stringet, decryptedText, amelybe majd beilleszti a visszafejtett karaktereket.

2. Létrehoz egy üres stringet, token, amelybe az aktuális titkosított karaktert tárolja.
3. Az endPos változóba elmenti az első szóköz pozícióját a 'titkos' stringben, vagy std::string::npos-t(végtelen értéket), ha nem talál szóközt.
4. Amíg van újabb szóköz a 'titkos' stringben (vagyis van újabb token), a következő lépéseket végzi el: -A token értékét beállítja a 'titkos' stringben található részletre (startPos és endPos között). -A token-t átkonvertálja unsigned long long típussá a std::strtoull() függvény segítségével, amely az értékét 10-es számrendszerben olvassa ki a stringből. -Az m értékét inicializálja. -Ha az c értéke megegyezik a RSA_ALPHABET_SIZE-szal, akkor az m értékét visszaállítja szóközként (visszaállítja a korábban speciális értékként kezelt szóközt), Különben kiszámolja az m értékét az RSA algoritmus segítségével. Ehhez a modularExponentiation() metódust használja a c karaktert átkonvertálva az 'a' értéktől kezdve számmá, majd a privateKey kitevővel emeli hatványra a RSA_ALPHABET_SIZE modulon belül. -Az m karaktert hozzáadja a decryptedText stringhez, miután explicit konvertálja char típusra. -Beállítja az startPos értékét az aktuális szóköz pozíciójának + 1 értékre. -Újra meghatározza az endPos értékét a következő szóköz pozíciójára a 'titkos' stringben, ha van még token. Ehhez az endPos értékét úgy módosítja, hogy az endPos-t az aktuális részleten belül adjon hozzá a startPos értékhez.
5. Visszaadja a decryptedText stringet, amely tartalmazza

Megvalósítja a következőket: Encryption.

4.3.3.2. encrypt()

```
std::string RSA::encrypt (
            const std::string & eredeti ) const  [override], [virtual]
```

encrypt függvény

Paraméterek

eredeti	A titkosítandó sztring

Visszatérési érték

Titkosított sztring Titkosítja a paraméterként kapott stringet

1. Először ellenőrzi az eredeti üzenet minden karakterét, hogy azok betűk vagy szóközök-e. Ha talál olyan karaktert, ami nem betű és nem szóköz, akkor kiírja a "Nem szabályos karakter" üzenetet, és "Error" értéket ad vissza.
2. Létrehoz két üres stringet: titkos és eredeti2. Az eredeti2 a toLowerCase() függvény segítségével átalakítja az eredeti üzenetet kisbetűssé.
3. Iterál az eredeti stringen a karakterek szerint.
4. Az aktuális karaktert (aktualis) leellenőrzi, hogy egy szóköz-e. Ha igen, akkor a c értékét a RSA_↩ ALPHABET_SIZE-ra állítja, ami különleges értékkel jelöli a szóközt. Ha nem szóköz, akkor kiszámolja a karakter titkosított értékét az RSA algoritmus segítségével. Ehhez a modularExponentiation() metódust használja a karaktert átkonvertálva az 'a' értéktől kezdve számmá, majd a publicKey kitevővel emeli hatványra a RSA_ALPHABET_SIZE modulon belül.
5. A c értékét átkonvertálja sztringgé (cStr), melyben minden számjegyet külön karakterként tárol.
6. Hozzáadja a cStr-t és egy szóközt a titkos stringhez, így az aktuális karakter titkosított értékét hozzáadja a titkos üzenethez.
7. Amikor végzett az összes karakterrel, visszaadja a titkos stringet, ami tartalmazza a titkosított üzenetet.

Megvalósítja a következőket: Encryption.

4.3.3.3. get_private_key()

```
std::string RSA::get_private_key ( ) const  [override], [virtual]
```

get_private_key függvény

Visszatérési érték

A titkos kulcs A függvény visszaadja a titkos kulcsot

Megvalósítja a következőket: Encryption.

4.3.3.4. get_public_key()

```
std::string RSA::get_public_key ( ) const  [override], [virtual]
```

get_public_key függvény

Visszatérési érték

A nyílt kulcs A függvény visszaadja a nyílt kulcsot

Megvalósítja a következőket: Encryption.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- RSA.hpp
- RSA.cpp

5. fejezet

Fájlok dokumentációja

5.1. Caesar.cpp fájlreferencia

```
#include <iostream>
#include <string>
#include "Caesar.hpp"
```

5.1.1. Részletes leírás

Szerző

Ujhelyi Bence (ujhelyibence@gmail.com)

Verzió

0.1

Dátum

2023-05-15

5.2. Caesar.hpp fájlreferencia

```
#include <iostream>
#include <string>
#include "Encryption.hpp"
```

Osztályok

- class Caesar

  Caesar osztály.

5.2.1. Részletes leírás

Szerző

Ujhelyi Bence (ujhelyibence@gmail.com)

Verzió

0.1

Dátum

2023-05-15

5.3. Caesar.hpp

Ugrás a fájl dokumentációjához.

```
00001
00009 #ifndef CAESAR_HPP
00010 #define CAESAR_HPP
00011
00012 #include <iostream>
00013 #include <string>
00014 #include "Encryption.hpp"
00015
00017 class Caesar : public Encryption {
00018 private:
00019
00020   // shift_char függvény
00021   static char shift_char(char c, int shift) {
00022     if (!isalpha(c)) return c;
00023     char base = isupper(c) ? 'A' : 'a';
00024     return (c - base + shift + 26) % 26 + base;
00025   }
00026   // shift változó
00027   int shift_;
00028 public:
00029
00030   // Konstruktor
00031   Caesar(int shift) : shift_(shift) {}
00032
00033   // encrypt függvény deklarációja
00034   std::string encrypt(const std::string& text) const override;
00035
00036    // decrypt függvény deklarációja
00037   std::string decrypt(const std::string& secrettext) const override;
00038
00039    // get_public_key függvény deklarációja
00040   std::string get_public_key() const override;
00041
00042    // get_private_key függvény deklarációja
00043   std::string get_private_key() const override;
00044
00045 };
00046
00047 #endif
```

5.4. Encryption.hpp fájlreferencia

```
#include <iostream>
#include <string>
#include <stdexcept>
```

Osztályok

- class Encryption

  Encryption osztály.

5.4.1. Részletes leírás

Szerző

Ujhelyi Bence (ujhelyibence@gmail.com)

Verzió

0.1

Dátum

2023-05-15

5.5. Encryption.hpp

Ugrás a fájl dokumentációjához.

```
00001
00009 #ifndef ENCRYPTION_HPP
00010 #define ENCRYPTION_HPP
00011
00012 #include <iostream>
00013 #include <string>
00014 #include <stdexcept>
00015
00017
00020 class Encryption {
00021 public:
00022
00024
00028   virtual ~Encryption() = default;
00029
00031
00034   virtual std::string encrypt(const std::string& plaintext) const = 0;
00035
00037
00040   virtual std::string decrypt(const std::string& ciphertext) const = 0;
00041
00043
00046   virtual std::string get_public_key() const = 0;
00047
00049
00052   virtual std::string get_private_key() const = 0;
00053 };
00054
00055 #endif
```

5.6. main.cpp fájlreferencia

```
#include <iostream>
#include <string>
#include "Encryption.hpp"
#include "RSA.hpp"
#include "Caesar.hpp"
```

Függvények

- int main ()
  Main függvény.

5.6.1. Részletes leírás

Szerző

Ujhelyi Bence (ujhelyibence@gmail.com)

Dátum

2023-05-15

5.6.2. Függvények dokumentációja

5.6.2.1. main()

```
int main ( )
```

Main függvény.

Teszteli a komponenseket

5.7. RSA.cpp fájlreferencia

```
#include "RSA.hpp"
#include <iostream>
#include <cstring>
#include <string>
#include <random>
#include <sstream>
#include <cstdlib>
#include <ctime>
```

Változók

- const unsigned long long RSA_ALPHABET_SIZE = 26
  RSA_ALPHABET_SIZE.

5.7.1. Részletes leírás

Szerző

Ujhelyi Bence

Dátum

2023-05-28

5.7.2. Változók dokumentációja

5.7.2.1. RSA_ALPHABET_SIZE

```
const unsigned long long RSA_ALPHABET_SIZE = 26
```

RSA_ALPHABET_SIZE.

Az rsa-hoz használt abc nagysága.

5.8. RSA.hpp fájlreferencia

```
#include "Encryption.hpp"
```

Osztályok

- class RSA

  RSA osztály.

5.8.1. Részletes leírás

Szerző

Ujhelyi Bence (ujhelyibence@gmail.com)

Verzió

0.1

Dátum

2023-05-15

5.9. RSA.hpp

Ugrás a fájl dokumentációjához.

```
00001
00009 #ifndef RSA_HPP
00010 #define RSA_HPP
00011
00012 #include "Encryption.hpp"
00013
00015 class RSA : public Encryption {
00016 private:
00017
00019     unsigned long long publicKey;
00020
00022     unsigned long long privateKey;
00023
00024     // generateRandomPrime függvény
00025     unsigned long long generateRandomPrime(unsigned long long min, unsigned long long max) const;
00026
00027     // isPrime függvény
00028     bool isPrime(unsigned long long num) const;
00029
00030     // modularExponentiation függvény
00031     unsigned long long modularExponentiation(unsigned long long base, unsigned long long exponent,
      unsigned long long modulus) const;
00032
00033     // modularInverse függvény
00034     unsigned long long modularInverse(unsigned long long a, unsigned long long m) const;
00035
00036     // gcd függvény
00037     unsigned long long gcd(unsigned long long a, unsigned long long b) const;
00038
00039     // findSpace függvény
00040     size_t findSpace(const std::string& str) const;
00041
00042     // getRandomNumber függvény
00043     unsigned long long getRandomNumber(unsigned long long min, unsigned long long max) const;
00044
00045     // toLowerCase függvény
00046     std::string toLowerCase(const std::string& str) const;
00047
00048 public:
00049
00050     // Default konstruktor
00051     RSA();
00052
00053     // encrypt függvény
00054     std::string encrypt(const std::string& eredeti) const override;
00055
00056     // decrypt függvény
00057     std::string decrypt(const std::string& titkos) const override;
00058
00059     // get_public_key függvény
00060     std::string get_public_key() const override;
00061
00062     // get_private_key függvény
00063     std::string get_private_key() const override;
00064 };
00065
00066 #endif
```

Tárgymutató