

# Cry – Project 2

## (Software Requirements Specification): Report

Daniel Dunning, Michael Degraw, Vu Phan

2017-02-13

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Purpose . . . . .	1
1.2	Scope . . . . .	2
1.3	Definitions, acronyms, and abbreviations . . . . .	2
1.4	References . . . . .	2
1.5	Overview . . . . .	2
<b>2</b>	<b>Overall description</b>	<b>2</b>
2.1	Product perspective . . . . .	2
2.2	Product functions . . . . .	2
2.3	User characteristics . . . . .	2
2.4	Constraints . . . . .	2
2.5	Assumptions and dependencies . . . . .	2
<b>3</b>	<b>Specific requirements</b>	<b>2</b>
3.1	External interface requirements . . . . .	3
3.1.1	User interfaces . . . . .	3
3.1.2	Hardware interfaces . . . . .	3
3.1.3	Software interfaces . . . . .	3
3.1.4	Communications interfaces . . . . .	3
3.2	Classes . . . . .	3
3.2.1	cryptosystem/ . . . . .	3
3.2.2	party/ . . . . .	4
3.2.3	cryptoframework.h . . . . .	6
3.3	Performance requirements . . . . .	7
3.4	Design constraints . . . . .	7
3.5	Software system attributes . . . . .	7
3.6	Other requirements . . . . .	7

## 1 Introduction

Michael Degraw

### 1.1 Purpose

Please write here.

## **1.2 Scope**

Please write here.

## **1.3 Definitions, acronyms, and abbreviations**

Please write here.

## **1.4 References**

Please write here.

## **1.5 Overview**

Please write here.

# **2 Overall description**

Daniel Dunning

## **2.1 Product perspective**

Please write here.

## **2.2 Product functions**

Please write here.

## **2.3 User characteristics**

Please write here.

## **2.4 Constraints**

Please write here.

## **2.5 Assumptions and dependencies**

Please write here.

# **3 Specific requirements**

Vu Phan

### 3.1 External interface requirements

#### 3.1.1 User interfaces

#### 3.1.2 Hardware interfaces

#### 3.1.3 Software interfaces

#### 3.1.4 Communications interfaces

### 3.2 Classes

#### 3.2.1 cryptosystem/

cryptosystem.h

```
#ifndef CRYPTOSYSTEM_CRYPTOSYSTEM_H
#define CRYPTOSYSTEM_CRYPTOSYSTEM_H

////////////////////////////////////

enum EnumeratedCryptosystem {rsa};

////////////////////////////////////

using IntPtr = mpz_t; // GNU Multiple Precision Integer Type
using Key = IntPtr;
using Text = IntPtr;

////////////////////////////////////

class Cryptosystem {
public:
    virtual void generateKeys(Key publicKey, Key privateKey); // set these

    virtual void encrypt(Text ciphertext, // set this
        const Text plaintext, const Key publicKey);

    virtual void decrypt(Text plaintext, // set this
        const Text ciphertext, const Key privateKey);

    virtual void cryptanalyze(Text plaintext, // set this
        const Text ciphertext, const Key publicKey);
};

////////////////////////////////////

#endif // CRYPTOSYSTEM_CRYPTOSYSTEM_H
```

rsa.h

```

#ifndef CRYPTOSYSTEM_RSA_H
#define CRYPTOSYSTEM_RSA_H

////////////////////////////////////

#include "cryptosystem.h"

////////////////////////////////////

class Rsa : public Cryptosystem {
public:
    void generateKeys(Key publicKey, Key privateKey); // set these

    void encrypt(Text ciphertext, // set this
        const Text plaintext, const Key publicKey);

    void decrypt(Text plaintext, // set this
        const Text ciphertext, const Key privateKey);

    void cryptanalyze(Text plaintext, // set this
        const Text ciphertext, const Key publicKey);
};

////////////////////////////////////

#endif // CRYPTOSYSTEM_RSA_H

```

### 3.2.2 party/

party.h

```

#ifndef PARTY_PARTY_H
#define PARTY_PARTY_H

////////////////////////////////////

#include "../cryptosystem/cryptosystem.h"

////////////////////////////////////

class Party {
public:
    Cryptosystem cryptosystem;

    Party(EnumeratedCryptosystem enumeratedCryptosystem);
};

```

```
////////////////////////////////////  
#endif // PARTY_PARTY_H
```

sender.h

```
#ifndef PARTY_SENDER  
#define PARTY_SENDER  
  
////////////////////////////////////  
#include "party.h"  
  
////////////////////////////////////  
  
class Sender : public Party {  
public:  
    Text ciphertext;  
  
private:  
    Text plaintext;  
  
    void encrypt(const Key publicKey);  
    // { cryptosystem.encrypt(ciphertext, plaintext, publicKey)}  
};  
  
////////////////////////////////////  
#endif // PARTY_SENDER
```

receiver.h

```
#ifndef PARTY_RECEIVER  
#define PARTY_RECEIVER  
  
////////////////////////////////////  
#include "party.h"  
  
////////////////////////////////////  
  
class Receiver : public Party {  
public:  
    Key publicKey;  
  
private:  
    Key privateKey;  
    Text plaintext;
```

```

    void generateKeys ();
        // { cryptosystem.generateKeys(publicKey , privateKey)}

    void decrypt(const Text ciphertext);
        // { cryptosystem.decrypt(plaintext , ciphertext , privateKey)}
};

////////////////////////////////////

#endif // PARTY_RECEIVER

```

#### eavesdropper.h

```

#ifndef PARTY_EAVESDROPPER
#define PARTY_EAVESDROPPER

////////////////////////////////////

#include "party.h"

////////////////////////////////////

class Eavesdropper : public Party {
private:
    Text plaintext;

    void cryptanalyze(const Text ciphertext , const Key publicKey);
        // { cryptosystem.cryptanalyze(plaintext , ciphertext , publicKey)}
};

////////////////////////////////////

#endif // PARTY_EAVESDROPPER

```

#### 3.2.3 cryptoframework.h

```

#ifndef CRYPTOFRAMEWORK_H
#define CRYPTOFRAMEWORK_H

////////////////////////////////////

#include "party/sender.h"
#include "party/receiver.h"
#include "party/eavesdropper.h"

////////////////////////////////////

```

```

class Cryptoframework {
public:
    Sender sender;
    Receiver receiver;
    Eavesdropper eavesdropper;

    Cryptoframework(EnumeratedCryptosystem enumeratedCryptosystem);

    void testKeyGeneration();
        // { receiver.generateKeys() }

    void testEncryption();
        // { sender.encrypt(receiver.publicKey) }

    void testDecryption();
        // { receiver.decrypt(sender.ciphertext) }

    void testCryptanalysis();
        // { eavesdropper.cryptanalyze(sender.ciphertext, receiver.publicKey) }
};

////////////////////////////////////

#endif // CRYPTOFRAMEWORK_H

```

### 3.3 Performance requirements

### 3.4 Design constraints

### 3.5 Software system attributes

### 3.6 Other requirements