OOAD HW5

20182211 김용빈

프로젝트명 : 다양한 암호 모듈을 적용하는 시스템

프로젝트 내용 : 사용자의 데이터를 안전하게 보호하기 위하여 중요한 정보나 개인 정보 등을 암호화하는 과정이 필요합니다.
이 때 암호화 방식, 암호화를 위한 키 발급 방식 등은 다양한 분야에서 각 상황에 알맞는 알고리듬을 채용합니다.

예를 들어서 한국의 카드 혹은 계좌 확인을 위해서는 Triple seed라는 암호화 방식이 사용되고, 해외 visa카드 등에서는 Triple DES라는 암호화 방식이 사용됩니다.

또한 키 교환 방식으로 교통카드는 DH, RSA 등이 사용되며

국방급 데이터와 민간급 데이터의 암호 방식, 저전력 IoT기기 등은 안전성과 속도를 상황에 맞게 타협하여 AES, ARIA, LEA 등의 암호 알고리듬을 이용하여 적합한 암호 모듈을 사용합니다.

이렇게 상황에 맞게 다양한 암호 모듈을 적용시켜주는 시스템입니다.

프로젝트 범위 :
키 발급을 위한 시스템, 암호화 모듈을 선택하면 암호화/복호화 과정을 진행해주는 시스템을 범위로 삼았습니다.

Use Case Description

Use Case name	다양한 암호 모듈을 적용하는 시스템
Scinario	사용자가 원하는 암호 알고리듬을 사용할 수 있게끔 하는 암호 모듈 시스템
Trigging event	데이터를 안전하게 전송하기를 원한다.(기밀성)
Brief description	사용자는 키 발급 방법을 선택하여 키를 발급받고 암호 모듈을 선택하여 암호화를 진행한다. 암호화된 데이터를 가진 사용자는 알맞은 복호 모듈을 사용하여 복호화를 진행한 후 안전하게 데이터를 다룬다.
Actors	데이터를 안전하게 보호하고 싶은 사용자
Related use cases	Include : 키 발급은 키 검증을 include한다. Extend : 키 발급에 실패하거나 키를 많이 재사용하면 키를 재발급 한다.
Stakeholders	데이터를 안전하게 보호하기 위한 모든 사용자 (국가급, 민간급)
Preconditions	시스템은 항상 가동중이어야 한다. 사용하는 암호 알고리즘은 국내 혹은 해외에서 검증받은 안전한 암호 알고리즘이다.
Post conditions	발급한 키는 일정 횟수 이상 재사용하지 않는다. (본 프로그램에서는 3회)
Flow of activities	다음 장에 기술합니다.
Exception Conditions	2. 키 발급에 실패하거나 키 재사용 횟수가 일정을 넘어가면 키를 재발급한다. 6. 키 발급에 실패하거나 키 재사용 횟수가 일정을 넘어가면 키를 재발급한다.

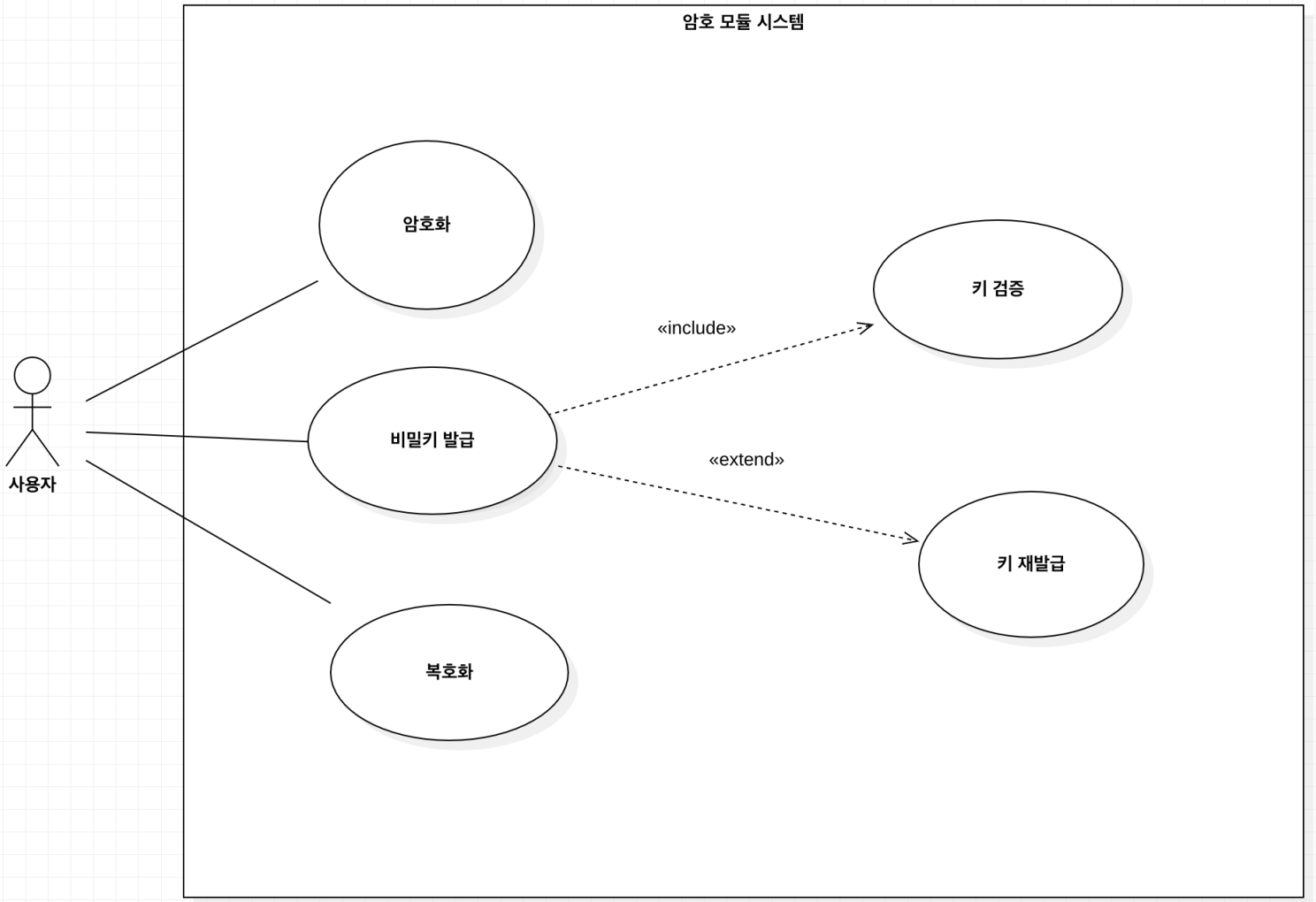
Flow of activities

Primary Actor:
사용자
- 역할 : 키 발급, 암호화 방법을 암호 모듈에서 선택

Main Success Scenario

Actor : 암호 모듈 적용 기기를 켠다.	시스템 작동
1. 대칭키 암호 모듈을 선택하고 대칭키(key) 발급을 요청한다.	
	2. 암호 모듈에 맞는 개인키를 발급하고 적절한 키인지 확인한다.
3. C : 대칭키 암호화를 진행하여 데이터를 암호화한다.	
	4. 암호화 방법에 맞게 암호화를 진행한다.
5. 암호화된 데이터를 대칭키로 복호한다.	
	6. 암호 모듈에 알맞는 복호화를 진행한다.

Use Case Diagram



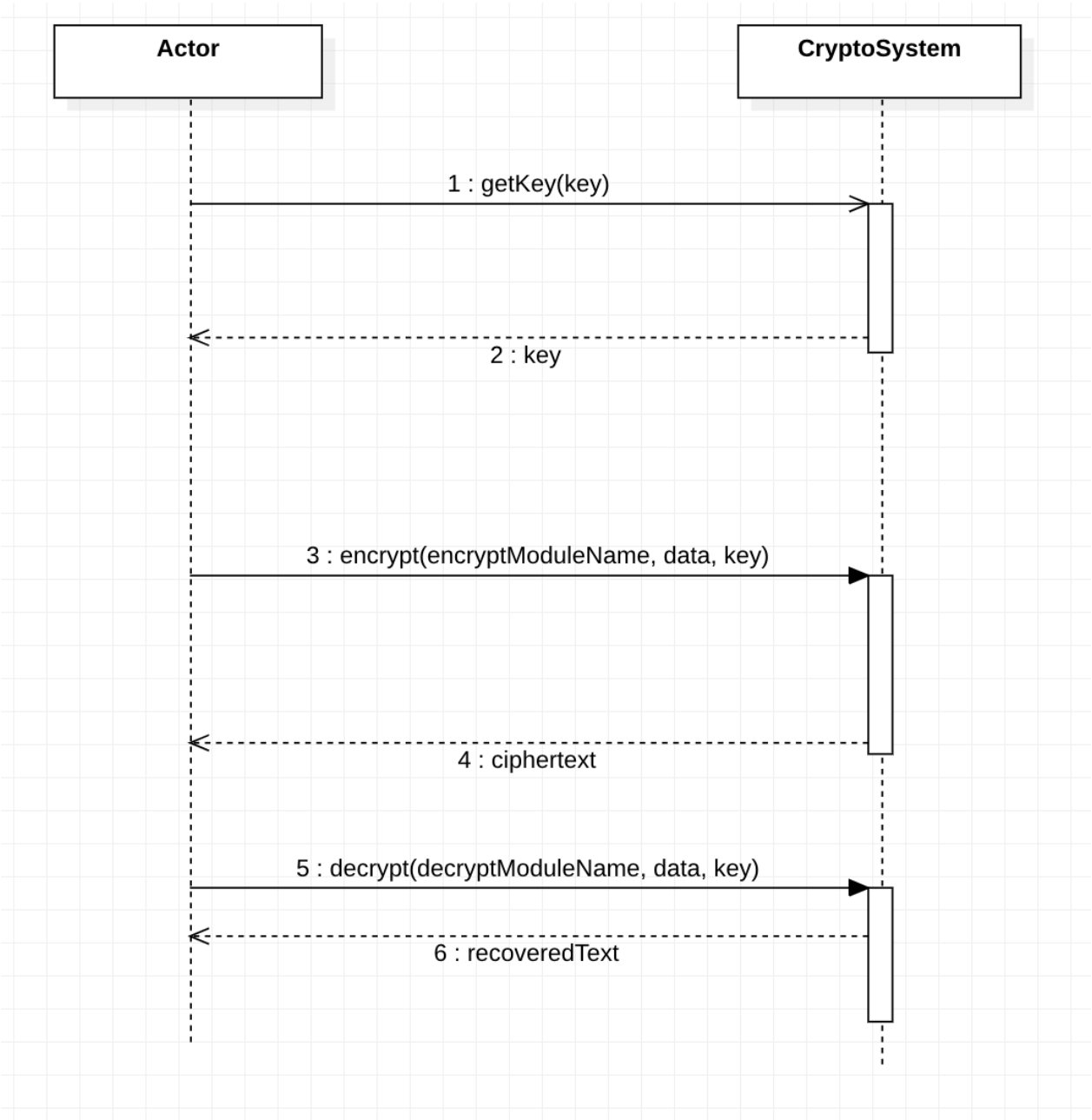
Non-Funtional Requirements

NFR	Quality	Quality Attribute
키 발급/ 키 검증	Performance	0.1초 안에 키 발급/ 키 검증 수행하기
암호화/복호화	Performance	1초 안에 암/복호화 과정 수행하기
보안 수준	Security - Confidentiality	Security level 1(128-bit) 이상의 안전성 준수하기
서버의 가용성	Reliability - Availability	99%의 가용성으로 서버 운영하기

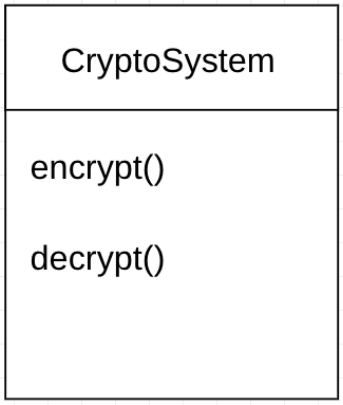
Domain Model

Class명	Concept 설명	주요 attributes
UserInterface	사용자가 CryptoSystem을 이용할 수 있게끔 도와주는 creator	query: string - 사용자가 요청하는 서비스를 저장하는 attribute cryptoModuleString - 사용자가 선택한 암호화 모듈을 저장하고 있는 attribute plaintextString - 사용자가 암호화하고자 하는 평문을 저장하는 attribute
CryptographicManager	유저 - 암호화, 복호화 모듈 혹은 키 관리 클래스간에 indirect하게 이어주는 담당하는 controller	crypto: Crypto - 암호화 instance launch:bool - 사용 가능한 모듈인지 확인하는 attribute
SingletonKeyManager	키 발급을 담당하는 Class	behavior만 존재
Crypto	대칭키 암복호화 알고리듬의 부모(추상) 클래스	key, pt, ct, recovered 등의 crypto 모듈에 필요한 속성들 (protected) behavior은 자식 클래스에서 overriding
Aes	Cryptographic algorithm인 Aes 암복호화 과정을 담당하는 알고리듬 클래스	roundKey, rc 상수 등의 Aes 암복호화 과정에 사용되는 값들
Aria	Cryptographic algorithm인 Aria 암복호화 과정을 담당하는 알고리듬 클래스	enc_w, dec_w, C1, C2, C3상수 등의 Aria 암복호화 과정에 사용되는 값들
GaloisField	Aes 암복호화 중간의 연산에 사용되는 체(Field)를 정의하는 클래스. Aes에서 체 중 특별히 갈루아 체 를 사용하기 때문에 클래스로 따로 관리한다.	matrix, row, col 등의 갈루아 체에서 정의되는 속성들

System Sequence Diagram



Operation Contract



System operations

Operation: encrypt

Responsibilities: 암호화 방식에 알맞는 비밀키(대칭키 암호에서 사용하는 키)를 이용하여 데이터(사용자에게 입력 받은)의 암호화 과정을 진행한다.

pre-conditions: 안전한 암호 모듈을 사용하며 안전한 비밀키를 발급받았다.

post-conditions: 평문과 암호문은 전혀 관련성이 보이지 않는다. (당연히 사용하는 암호 모듈에 따라서 생성된 암호문이 다르다.)

Operation: decrypt

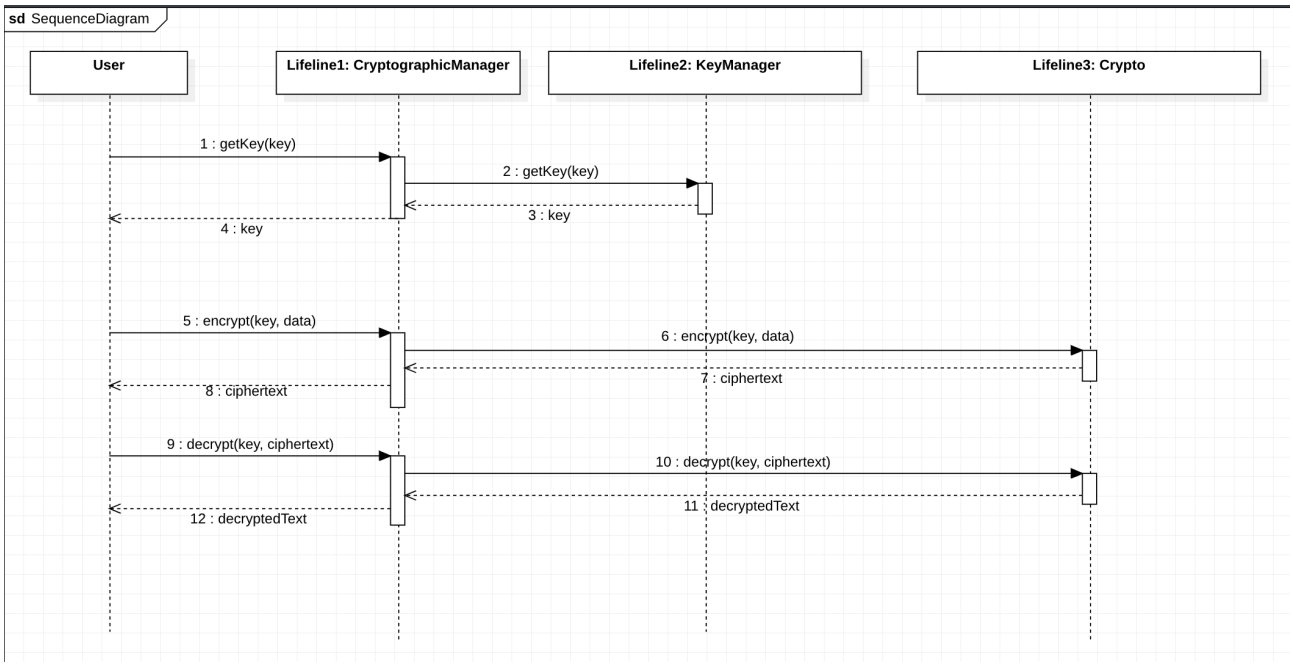
Responsibilities: 복호화 방식에 알맞는 비밀키(대칭키 암호에서 사용하는 키)를 이용하여 데이터의 복호화 과정을 진행한다.

pre-conditions: 안전한 암호 모듈을 사용하며 안전한 비밀키를 발급받았다.

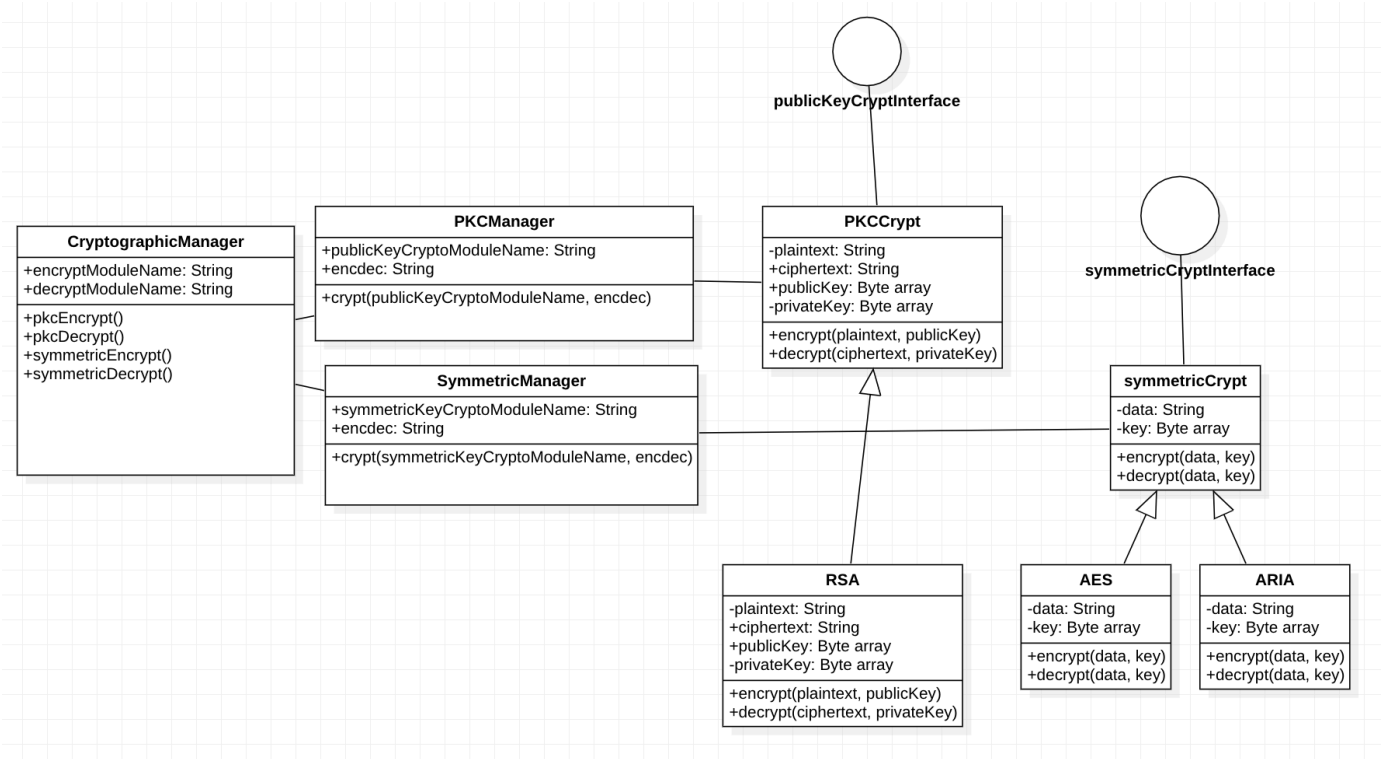
post-conditions: 암호문을 복호화한 복호문이 평문과 일치한다.

Contracts

Sequence Diagram

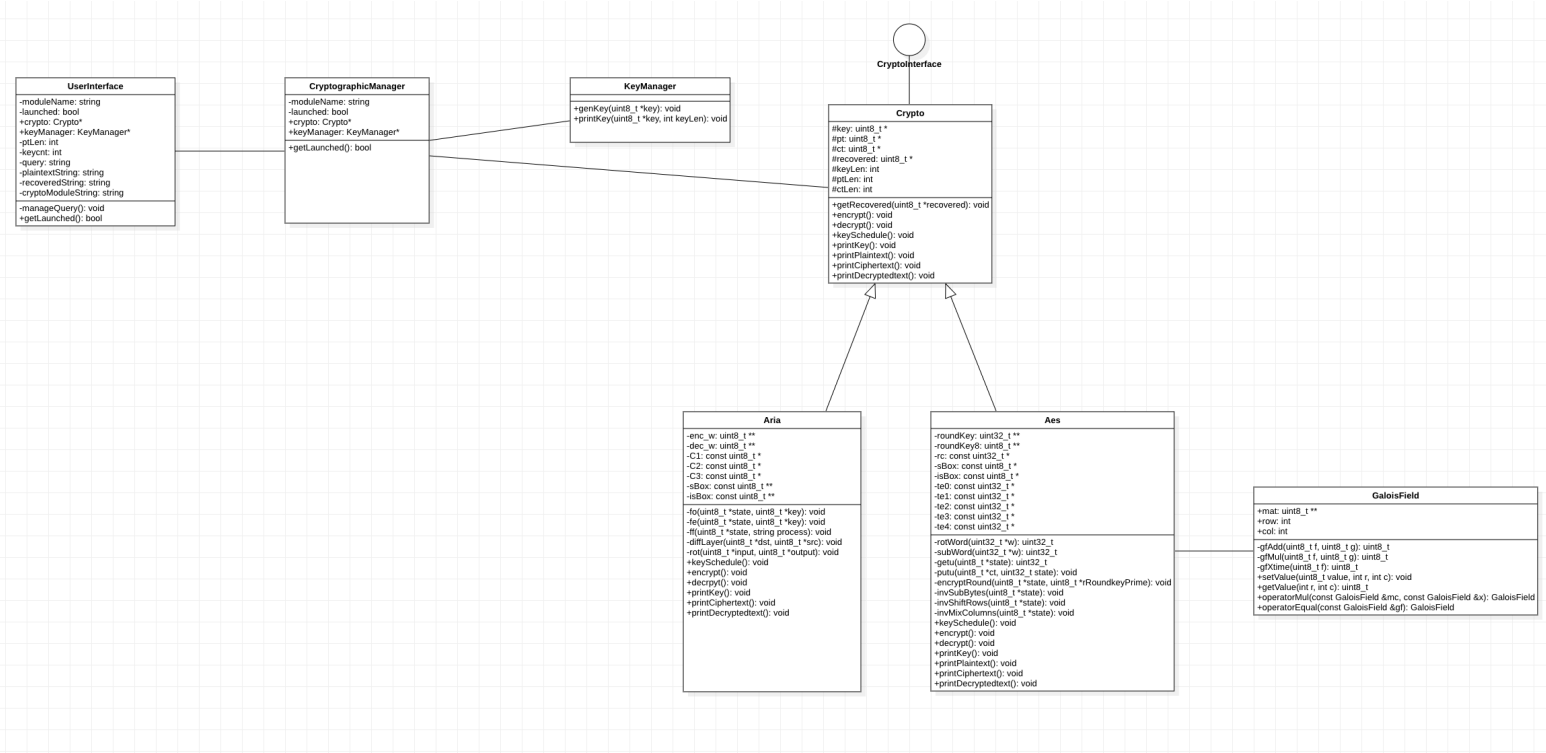


(HW3의) Class Diagram



(HW4의) Class Diagram

GRASP에 해당되는 설계 부분에 대하여 해당 class에 표시하기
ex. controller, creator, expert, low coupling, high cohesion, don't talk to strangers



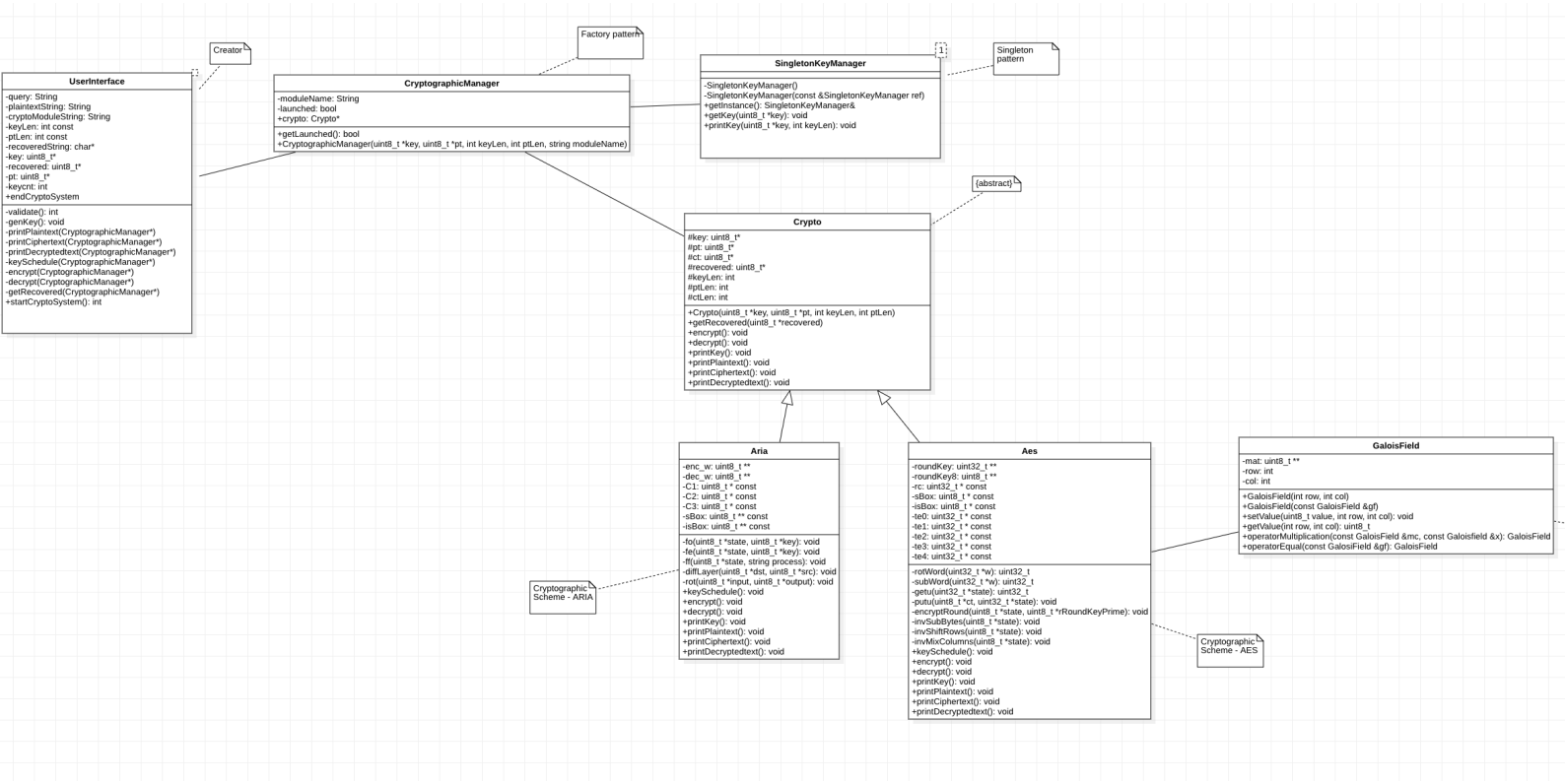
CryptographicManager 클래스가 Controller의 역할을 합니다.
Actor가 UserInterface 클래스를 사용하며 KeyManager 클래스, Crypto 클래스의 기능을 다루고 싶어하는 상황을 가정해보면
우선 UserInterface 클래스에서 CryptoManager 클래스의 객체를 생성하고,
키 발급, 검증과 관련된 기능을 사용하고 싶으면 CryptoManager 클래스의 객체를 이용하여 KeyManager 클래스를 이용하게끔 사용하고,
암호화, 복호화 등 암호 알고리즘과 관련된 가능을 사용하고 싶으면
CryptoManager 클래스의 객체를 이용하여 Crypto 클래스를 이용하게끔 사용합니다.

이 때 CryptographicManager 클래스가 KeyManager, Crypto 클래스를 attribute로 가지고 있기 때문에 UserInterface 클래스와 KeyManager, Crypto 두 클래스를 연결해주는 역할을 합니다.
또한 CryptographicManager을 사이에 끼고 있는 UserInterface, KeyManager, Crypto 클래스 끼리는 서로 알 필요도, 알지도 못합니다. 이는 Low Coupling에 해당하며, 서로 상호작용하지 않는다는 점에서 Don’t Talk to Strangers에 해당합니다.

UserInterface 클래스가 Creator의 역할을 합니다.
사용자는 프로그램을 시작하면 UserInterface 클래스의 객체를 이용하게 되는데, UserInterface 클래스가 사용자에게 쿼리를 입력받아 키 발급, 키 검증, 암호화, 복호화 등의 기능을 해당 쿼리에 알맞는 기능을 하는 클래스의 객체를 불러서 사용자가 이용할 수 있게끔 도와줍니다.

암호화, 복호화의 기능을 할 수 있는 Crypto 클래스의 서브클래스로 Aes, Aria가 있습니다.
Aes 클래스는 Aes 암호 알고리듬의 expert 역할을 합니다. 이 때 현대대수의 개념인 체(Field) 중에서 Galois Field GF(2^8)을 Aes의 체로 활용하는데, 이 체를 다루는 클래스는 GaloisField 클래스입니다. GaloisField 클래스는 Galois Field의 expert 역할을 합니다. 즉 Aes 클래스는 Galois Field를 필요로 하지만 그 내부 동작은 GaloisField 클래스에서 전담합니다. 따라서 Aes에서 Galois Field 대신에 다른 체를 사용하고 싶다면 GaloisField 클래스 대신에 다른 체 클래스로 변경하기에 용이합니다. 이는 Aes는 Aes알고리즘만, GaloisField는 체의 역할로만 사용되므로 각각 그 Expert에 해당하며 이로 인해 Low Coupling의 효과를 봅니다. 또한 GaloisField 클래스를 대체하기 용이하다는 점에서 High Cohesion에 해당합니다.

(HW5의) Class Diagram



HW5에서 Design Pattern으로 Singleton Patterm, Factory Pattern을 추가하였습니다. 이에 대하여 HW 5 Class Diagram에 표시하였고, 아래의 Design Refinement by GRASP Principle and Design Patterns에 부연설명 하겠습니다.

Design Refinement by GRASP Principle and Design Patterns

Before 단계의 class 등 설계 classifier	After 단계의 Class 등 설계 classifier	Architecture Design Ratiomale	Architecture Design Ratiomale	NFR와 QA에 대한 영향 분석
KeyManager	**SingletonKeyManager**	Singleton	기존 KeyManager을 CryptographicManager의 attribute으로 가지고 있었으나 삭제하고, KeyManager 클래스를 SingletonKeyManager Class으로 변경하여 사용했습니다. SingletonKeyManager은 Singleton이므로 Constructor (Defalut, parameterized, copy 모두)의 visibility을 private으로 하였고, getInstance() method으로 SingletonKeyManager instance을 가져와서 사용할 수 있습니다. 사용하는 시점은 암호화가 되기 이전에 사용자가 키 발급을 요청하면 CryptographicManager가 SingletonKeyManager::getInstance()으로 인스턴스를 가져오면 SingletonKeyManager가 lazy initialize으로 반환합니다.	instance가 필요할 때 똑같은 instance를 만들지 않고 딱 한 번만 만들게 되므로, 불필요한 자원을 줄이고 속도가 더 빨라졌다. NFR 키 발급 - QA에서 0.1초 안에 키 발급에 성공하였습니다.

Before 단계의 class 등 설계 classifier	After 단계의 Class 등 설계 classifier	Architecture Design Ratiomale	Architecture Design Ratiomale	NFR와 QA에 대한 영향 분석
CryptoGraphicManager	CryptographicManager	GRASP - Controller, Design Pattern - Factory Pattern	Crypto class가 Product라고 하면 Aes, Aria class가 ConcreteProduct에 해당하고 Creator, ConcreteCreator에 해당하는 부분이 CryptographicManager이라고 할 수 있습니다. CryptographicManager은 factoryMethod와 operation을 가지고 있습니다. 사용자가 원하는 암호 모듈이 Aes인지 Aria인지 프로그램 시작 전에는 모릅니다. 따라서 사용자가 입력하여 UserInterface 로 부터 전달받은 암호 모듈에 따라 CryptographicManager가 동적으로 결정해야할 필요가 있습니다. CryptographicManager는 사용자가 Aes 암호 모듈을 사용하고 싶다면 Aes로, Aria 암호 모듈을 사용하고 싶다면 Aria를 이용할 수 있게끔 instant를 생성합니다. 한편 Crypto라는 superclass는 Aes, Aria를 서브클래스고 가지고 있는데, Aes, Aria는 Crypto의 encrypt(), decrypt() 등의 behavior을 overriding하고 있어서 각자 상황에 알맞게 끔 이용이 가능하고, 따라서 CrytographicManager가 Factory pattern으로 적합하게 설계되었습니다.	Controller의 역할을 함으로써 암호 모듈 scheme와 key management, user 가 전부 분리되고 암호 공격에 대하여 좀 더 안전해져서 서버의 가용성이 99% 이상이 되었다.

Before 단계의 class 등 설계 classifier	After 단계의 Class 등 설계 classifier	Architecture Design Ratiomale	Architecture Design Ratiomale	NFR와 QA에 대한 영향 분석
Crypto	Crypto	Polymorphism	Crypto가 subclass로 Aes, Aria을 가지고 있는데 이 둘은 Crypto의 encrypt(), decrypt() 등의 method를 overriding하고 있다. 따라서 다형성이 잘 지켜지므로 확장성이 용이합니다.	암호 모듈의 추가가 용이해진다.
Aes, Aria	Aes, Aria	(Cryptographic algorithm)	Aes와 Aria는 암호 모듈입니다. 전부 직접 구현하였으며, 그 안에서 , 수학적, 소프트웨어적 수준애서 할 수 있는 최적화까지 최대한 진행하였습니다. Openssl에서 가져올 수도 있지만 최적화의 관점과 최신 암호들은 Openssl에 없다는 점을 고려하여 암호 모듈의 확장성과 재사용성이 매우 용이하게 설계하였습니다.	암/복호화 전부 1초 안에 수행 가능하며 본 프로그램에서 사용하는 암호 모듈 모두 Security level(암호에서 사용하는 안전성 평가, 128-bit 안전성을 의미합니다.) 을 준수하였습니다.

결과 Snapshot
ooad 수업과 프로젝트를 마무리 지으며..

처음에 상당히 막막했습니다. 전공이 아닐 뿐더러 oop 수업을 들은 적도 없었지만 객체지향에 대하여 느끼고 싶어서 소프트웨어 방법론을 하나도 모른 채 수업을 듣기 시작했습니다. 하지만 수업을 들으면서 프로젝트와 병행하여 조금씩 해나가니 와닿기 시작했습니다. 또 프로젝트가 한 학기동안 점점 쌓여가는 형식이다 보니 이 프로젝트가 한 학기 수업을 자신만의 언어로 정리해가는 느낌이었습니다. 제가 구현한 암호 primitive는 전부 c나 assembler 혹은 hardware 수준에서 구현이 되어있는데, c++을 이용하여 수업에서 배운 내용들을 기반으로 분석하고 설계하다 보니 우여곡절 끝에 암호라는 저만의 언어로 잘 소화해낸 결과인 것 같습니다. 처음 프로젝트 주제 선정할 때 고민이 많았는데, 잘 하는 분야를 프로젝트로 도전해 보라는 조언 정말 감사했습니다.
그리고 역시 설계 과정에서 중요한 것들이 매우 많지만 class diagram의 중요성을 뼈저리게 깨달았습니다. class diagram을 아는 사람들에게는 프로그램을 공유하는 데에도 앞으로도 매우 유용하게 사용하게 될 것입니다.

결과도 매우 만족스럽습니다. 암호 분야에서는 양자컴퓨터에 대비해 요즘 양자컴퓨터에 내성이 있는 양자내성암호 공모전이 국제적으로 이루어지고 있는데, 그런 것들에 대해서는 암호 모듈이 잘 정립되어 있지 않습니다. 심지어 공식 사이트에서 까지도요. 그런 최신 암호 모듈들을 여기에 추가로 삽입하기 쉽게 구현해놨기 때문에 용이하게 붙일 수 있을 것 같아서 구현 결과에 대해서도 만족스러운 프로젝트였습니다.

Use Case 구동 시연 영상 설명

환경 : Mac os
```
Apple clang version 13.0.0 (clang-1300.0.29.3)
Target: x86_64-apple-darwin20.6.0
```

c++버전은 11을 사용하게끔 Makefile 컴파일 옵션에 넣었습니다.

실행 방법 : HW_CryptoSystem 디렉토리 내부에서 make 명령어로 컴파일 후, ./cryptosystem으로 run합니다.

Use Case 중 키 발급, 키 검증, 암호화, 복호화 과정에 대하여 시연하였습니다.

사용자가 프로그램을 실행하고 키 발급(1)을 선택합니다.
그러면 128-bit의 랜덤한 키가 생성됩니다. 또한 키를 3번까지 재사용 가능하게끔 구현하여서 현재 키 사용 횟수는 0입니다.

키를 발급받은 후 암호화&복호화(3)을 이용하여 데이터를 입력하고, 평문과 암호 모듈 이름을 입력하게 됩니다.
Hello World! 를 ARIA 암호 알고리듬으로 암호/복호가 정상적으로 진행되었음을 확인할 수 있습니다.
이후 키 검증을 하면 발급받은 키의 사용 횟수는 1입니다.

그 다음 동일한 키를 이용하여 암호화&복호화(3)을 이용하여 동일한 데이터 Hello World!를 입력한 후 AES 암호 알고리듬으로 암호/복호가 정상적으로 진행되었음을 확인할 수 있습니다. 이 때 앞서 동일한 평문과 동일한 키로 ARIA 암호모듈을 이용한 암호문과는 다름을 확인할 수 있습니다.

이후 키 검증을 하면 발급받은 키의 사용 횟수는 2입니다.
그 뒤 암호화&복호화(3)를 한 번 더 하고 키 검증을 하게 되면
키의 사용 횟수가 3이 되고 이 때 암호화&복호화(3)을 시도하게 되면 키를 재발급 받으라는 문구가 나오게 되고 암호화&복호화 과정을 진행하지 않습니다.

이러한 예외 사항에 사용자는 키 재발급(1)을 다시 받고 새로운 키를 이용하여 암호화&복호화(3)의 과정을 진행할 수 있습니다.