Portfolio

이성용
sy84.lee@gmail.com
https://github.com/sy84lee
010-6414-8305

2010. 02 ~ 2017. 12

- Developed Requirement for LG Mobile of Domestic
- Designed and Developed Internalization of FOTA Android Application
- Designed and Developed A/B Update Project
- Developed OMA-DM Android Application in Mobile of Sprint
- Developed FOTA Android Application in Mobile of Sprint
- Developed Test Application for OMA-DM, FOTA
- Developed Unit Test for LG Mobile of Domestic
- Developed Test Tracer from Req to Unit Test

2018. 01 ~ 2021. 06

- Developed The Uptane that Standard of Secure FOTA
- Designed and Developed Light Secure FOTA Module
- Designed and Developed IDPS based on the Iptables
- Designed and Developed Policy Updater of IDPS
- Designed and Developed IDPS based on the Suricata

2022. 07 ~ Now

- Developed Cryptography Algorithm Module
- Developed Secure Boot Solution
- Developed Test Suite for PK, PQC
- SDET (Software Development Engineer in Test, LG)

Activity

- SW Associate Architect Certificate(Samsung)
- JAVA Programming Certificate (LG)
- C Programming Certificate (LG)
- Software Programing Certificate, Pro (Samsung)

FOTA DM-UA Interface 설계

▪ 개요

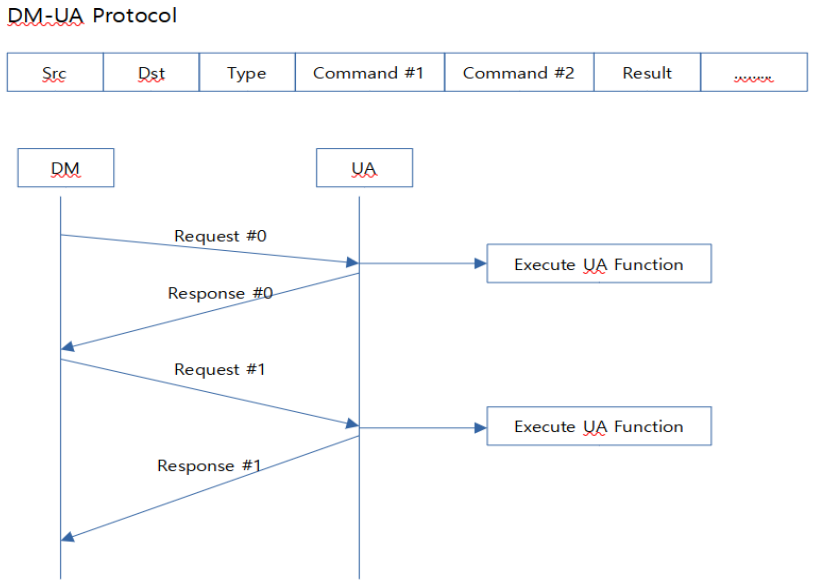
- Android의 FOTA Application은 크게 2개의 모듈로 구성되어 있음
- Android Layer의 DM(Device Management)와 Firmware Layer의 UA(Update Agent)
- DM은 UA에서 제공하는 API를 Call하여 Update 및 관련 동작을 수행함
- 따라서 DM은 UA에 Dependency가 발생하고 각 모델에 각 모듈을 Porting진행 시 UA의 Porting 진행상황에 Dependency가 발생하기 때문에 DM개발에 문제점이 발생함
- 이러한 문제를 해결하기 위해 API 대신 DM-UA간의 통신 Protocol을 제안함

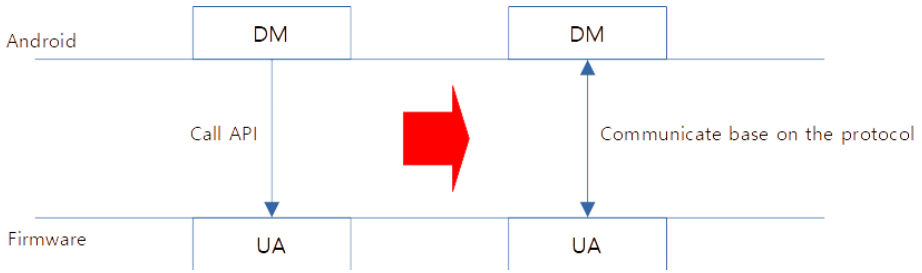
▪ 성과

- DM과 UA는 서로 간의 의존성을 제거하여 각 모듈 개발에 집중할 수 있음
- 의존성이 제거되어 각 모듈의 검증도 용이 해짐

▪ 개발

- DM-UA의 통신 Protocol을 설계
- DM-UA의 통신 방법은 Socket 또는 파일시스템 등 각 모델에서 제공하는 방법을 사용

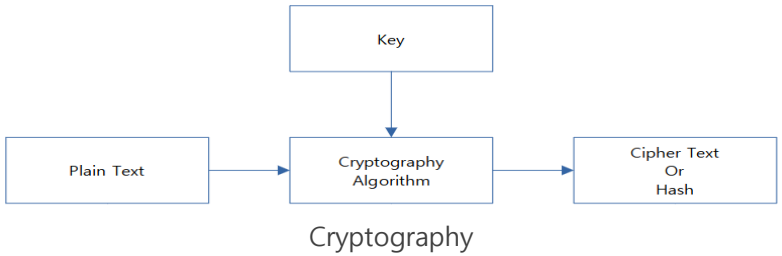




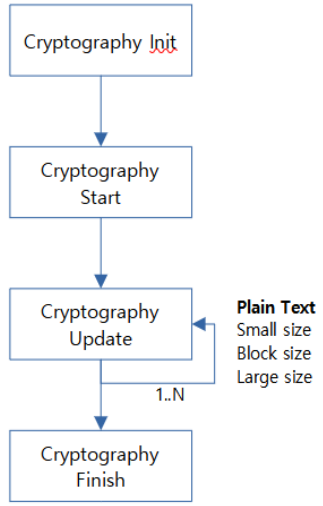
Cryptography SW Module Test design 설계 (1/2)

▪ 개요

- Cryptography Algorithm SW Module에 대한 Test Case 구현
- SHA, AES는 Block 단위로 Cryptography Algorithm이 동작함
- SW로 구현한 Cryptography는 입력 Data에 대해 Cryptography Algorithm에서 사용하는 Block Size와 무관하게 입력 받도록 구현되어 있음.
- 즉, SW Cryptography는 Block Size를 기준으로, Small, Same, Large 크기의 입력 데이터를 받을 수 있음.
- 또한 1~N번의 입력을 받을 수 있음
- 따라서, Small, Same, Large에 대해 다양한 조합의 입력 순서가 발생할 수 있음
  - Init -> Start -> Update Small Size Data -> Update Small Size Data -> Finish
  - Init -> Start -> Update Same Size Data -> Update Small Size Data -> Finish
  - Init -> Start -> Update Large Size Data -> Update Large Size Data -> .....
- 각각의 Test Case에 대해 각 Step을 구현할 경우 많은 작업을 요구함



Cryptography



Function Call Process

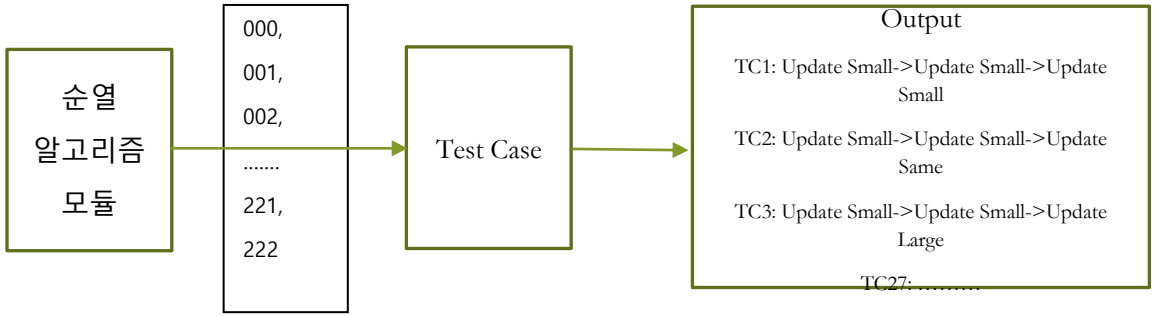
Cryptography SW Module Test design 설계 (2/2)

▪ 성과

- Test Case의 입력을 중복 순열 알고리즘을 응용하여 Test Case의 작업을 줄임.
- 즉, 3번의 Data Update를 Test Case를 작성할 때, 3 x 3 x 3 = 27개의 Test Case를 직접작성해야 하나, 중복 순열 알고리즘을 사용하여 1개의 Test Case 작업으로 27개의 Test Case를 생성할 수 있음.
- Depth를 입력 Parameter로 사용하기 때문에, 1~N개의 Depth를 조절할 수 있음.
- Array_data와 Array_data_size에 Fuzz Algorithm을 연결한다면, Random한 Data와 Random한 Size도 검증이 가능함

▪ 개발

- Data size option은 3가지가 존재. Small, Same, Large
- 기본 컨셉은 3가지 Option에 대해 중복 순열 알고리즘을 도입
- 3가지 종류의 입력할 Data를 Array로 생성.
  - Array_data[][] = {{Small Data}, { Same Data}, {Large Data}}
- 3가지 종류의 입력할 Data의 Size를 Array로 생성.
  - Array_data_size[] = {small, same, large}
- 중복순열 알고리즘은 입력받은 r에 따라서 중복순열 리스트를 출력
  - If r = 3이면 $\{0, 0, 0\}, \{0, 0, 1\}, \{0, 0, 2\}$ …….. $\{2, 2, 2\}$
  - 출력된 중복순열의 각 순열에 따라 Array_data와 Array_data_size의 Value를 가져와 Test Case의 입력으로 삽입함



OMADM/FOTA Android Application 개발

▪ 개요

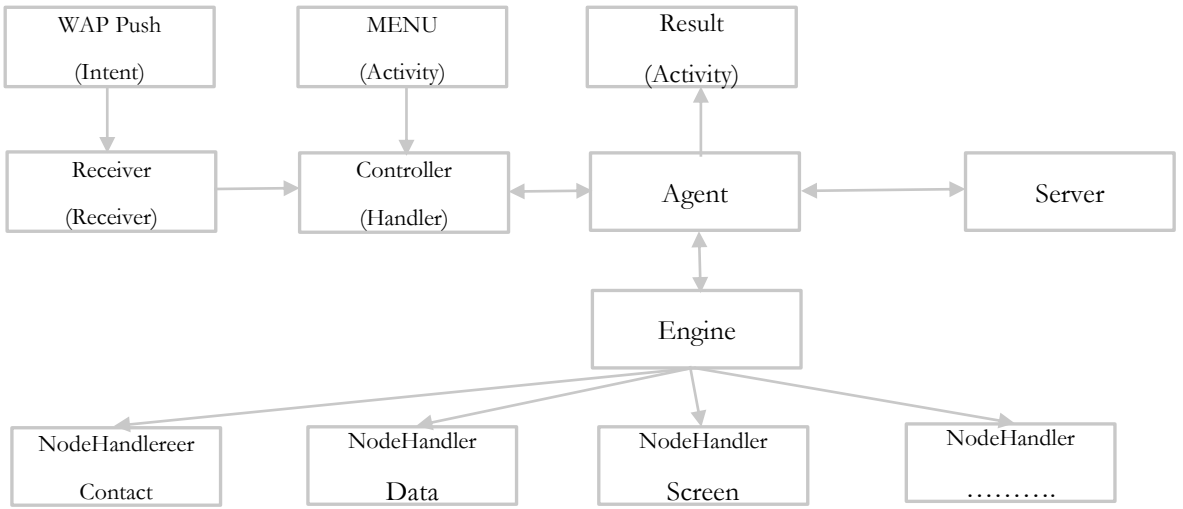
- OMA-DM Protocol에 따라 서버와 통신하여 Device를 제어하는 Android Application
- Http 또는 Https를 통해 서버와 통신
- OMADM Android Application은 타 기능과의 Communication 수행
- Server Initiation, Client Initiation을 지원

▪ 성과

- NodeHandler Interface를 통해 다양한 유관부서 담당자와의 역할 분배 확립
- NodeHandler Interface를 통해 OMA-DM을 통한 각 유관 기능의 검증이 용이하게 됨
- 타 북미 사업자 Android FOTA Application의 선행 구조가 됨
- 북미 Sprint 사업자의 Android Device 모델에 탑재되어 출시됨
- 기존 Android OMA-DM Application 대비 Issue 발생율이 현저히 감소함

▪ 개발

- Controller, Agent, Engine의 구조를 가짐.
- 내/외부로부터의 입력은 Controller가 받아 순차적으로 수행
- 서버와의 통신 및 데이터 처리, UI 구성은 Agent에서 수행
- OMA-DM Protocol Message Parsing은 Engine에서 수행
- NodeHandler Interface를 통해 타 기능을 제어



Test Case Tracer Tool 개발

▪ 개요

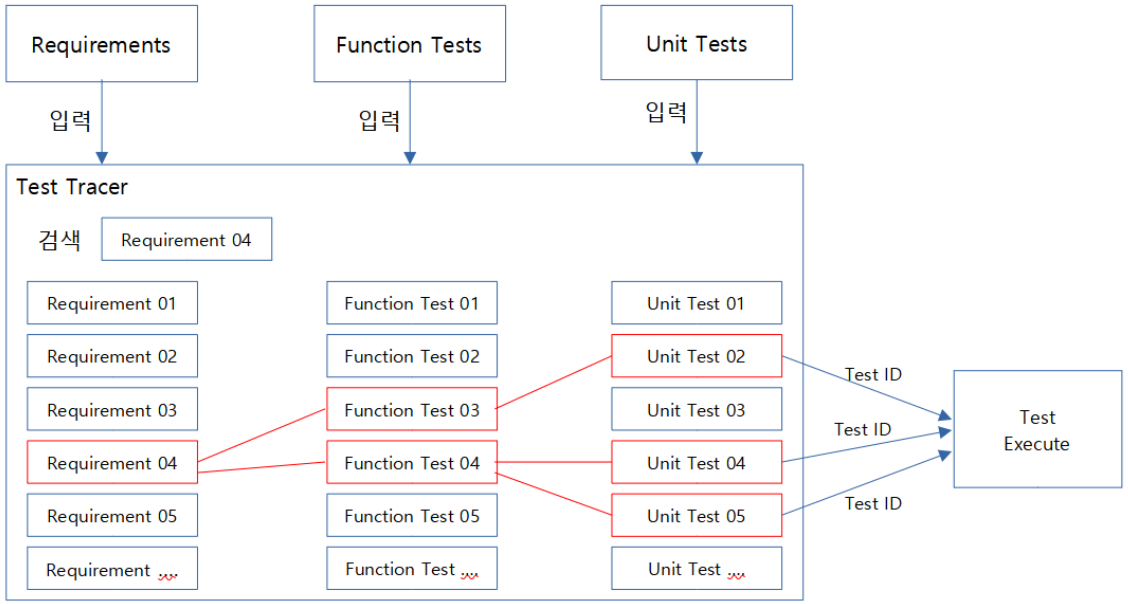
- 신규 요구사항의 발생 또는 요구사항의 변경 시 검증 누락 발생
- 변경된 사항에 대해서만 집중 검증하고, 변경된 사항에 대해 영향을 받는 기능은 검증이 부재 되는 상황이 많음
- 이러한 검증의 부재는 이슈 발생의 원인이 됨
- 변경점 또는 요구사항으로의 테스트 케이스를 추적함으로 변경점에 따라 필수 검증되어야 하는 테스트 케이스를 수행

▪ 성과

- 요구사항과 Test case의 추적성을 확보를 통해 요구사항 변경 또는 신규 요구사항에 대한 검증 능력이 강화됨
- 요구사항 변경 등의 이벤트 발생 시 Tool을 활용한 변경점 검증 프로세스를 추가함으로 Issue 발생율이 낮아짐
- UI를 통한 구현을 통해 Test Case 관리가 용이해짐
- 개발자들의 검증에 대한 인식 변화

▪ 개발

- 요구사항과 테스트 케이스의 입력을 받아 추적성을 확보할 수 있는 Tool을 개발
- 요구사항 선택 했을 경우 관련된 Function Test와 Unit Test를 Display
- Unit Test를 선택 했을 경우 관련된 Function Test와 요구사항을 Display



Uptane (Secure FOTA Framework) 개발

▪ 개요

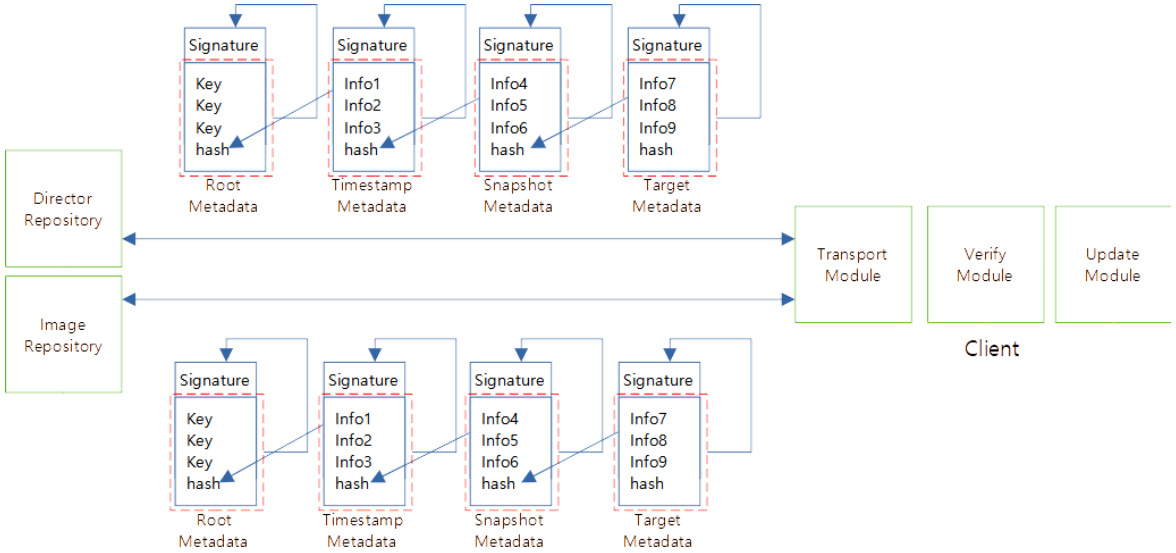
- Cyber 공격의 증가로 인해 복원 방법의 중요성이 대두됨
- Network를 통한 복원 방법의 하나로 FOTA가 주목받게 됨
- FOTA의 중요성이 커지고 FOTA 자체에 대한 보안성도 필요하게 됨
- FOTA를 안전하게 수행하기 위한 Requirement가 만들어지게 됨
- https://uptane.github.io/

▪ 성과

- FOTA 기능의 보안성 향상
- 다양한 응용을 통해 자사 제품 탑재
  - Light Version 개발을 통한 저사양 장치 탑재
  - 기능 모듈화를 통해 선택적 기능 탑재

▪ 개발

- Server (Director Repository, Image Repository)
  - 각 Metadata 생성
  - Private Key를 통해 Signature 생성
- Client (담당 역할)
  - Server로부터 4개 Metadata X 2 = 8개의 Metadata를 Download 받음
  - 각 Metadata의 Signature Verify 진행
  - 각 Metadata의 Information Verify 진행
  - Verify 된 정보에 따라 Update 수행



IDS(Intrusion Detection System) 개발

▪ 개요

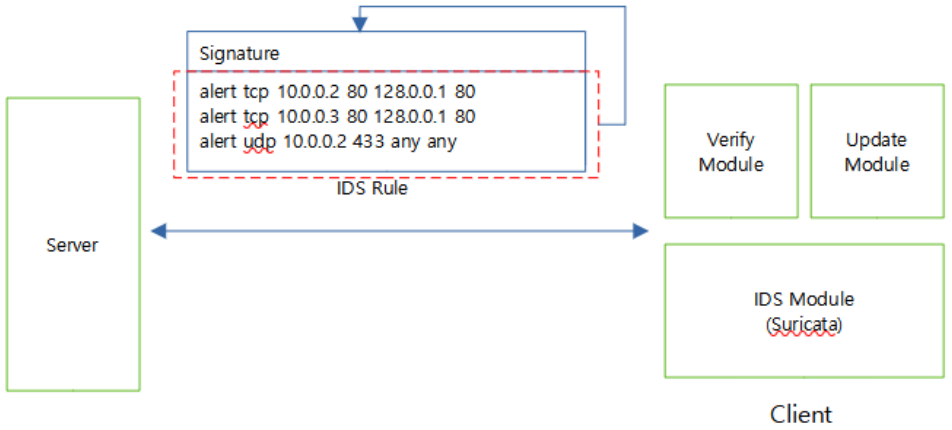
- Opensource(Suricata)를 활용한 Network Packet 이상탐지 기능 개발 탑재
- Network를 통한 IDS Policy 및 Rule Update 기능 탑재
- Update 기능은 무결성 및 기밀성 제공

▪ 성과

- IDS 개발을 통해 침입 탐지 기술 마련
- IDS 기술 제품 적용을 통해 제품의 보안성 향상
- Module 교체를 통해 동일한 기능의 선택적 모듈 적용 가능
- Ex) Suricata -> Snort

▪ 개발

- Server
  - IDS Policy 생성 및 UX 제공
- Client (담당 역할)
  - 이상탐지 기능 수행
  - IDS Policy 또는 Rule Update 기능 수행
  - Update Data는 암호화 및 Signature를 통해 무결성, 기밀성 제공
  - Update Data가 전송된 후 Client는 Signature 검증 및 복호화 후 Update 수행



Cryptography Algorithm API 개발

▪ 개요

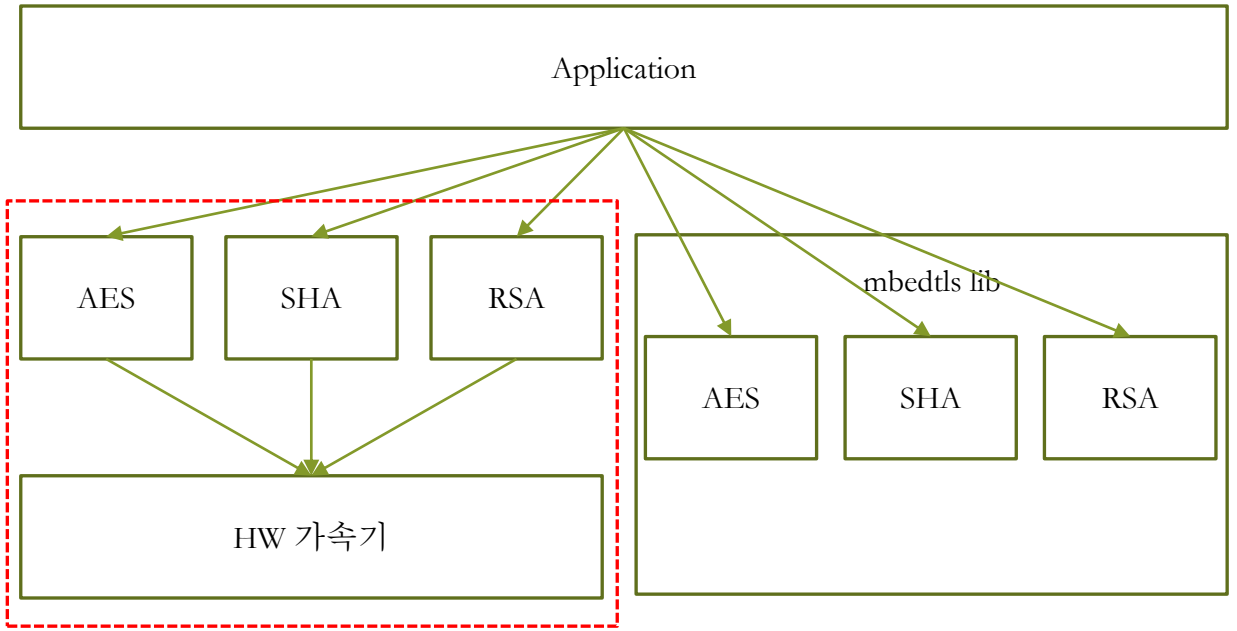
- Cryptography HW 가속기에 대한 어려운 사용접근성을 해결하기 위한 방안
- 대부분의 사용자는 Cryptography에 대한 알고리즘 지식이 없음
- HW 가속기를 사용하여 Cryptography API를 제공
- Opensource와 유사형태로 제공하여 사용 접근성 및 편의성 제공

▪ 성과

- Cryptography HW 가속기에 대한 사용법 예제로 사용 가능
- Mbedtls에서 제공하는 API와 같은 형태로 개발하여 사용편의성 제공
- SW Code로만 작성된 Cryptography보다 빠른 속도 제공

▪ 개발

- mbedtls를 benchmark 하여 개발
- mbedtls를 사용하는 코드에서 mbedtls API에서 신규 개발 API로 교체 가능하도록 개발
- HW Exception Case를 SW에서 제어하여 사용편의제공



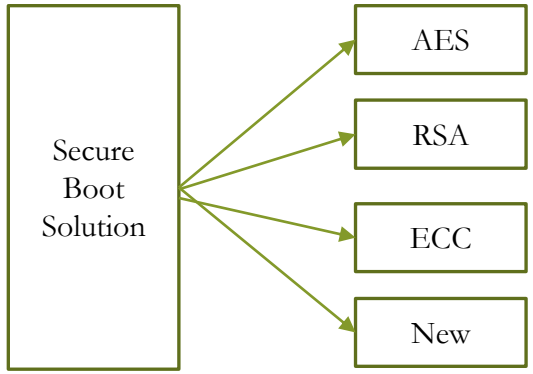
Secure boot Solution 개발

▪ 개요

- Image 및 Data를 Verify할 수 있는 Solution을 제공함으로 제품의 Security 향상
- 다양한 Cryptography Algorithm을 사용할 수 있는 Secure Boot Solution 개발

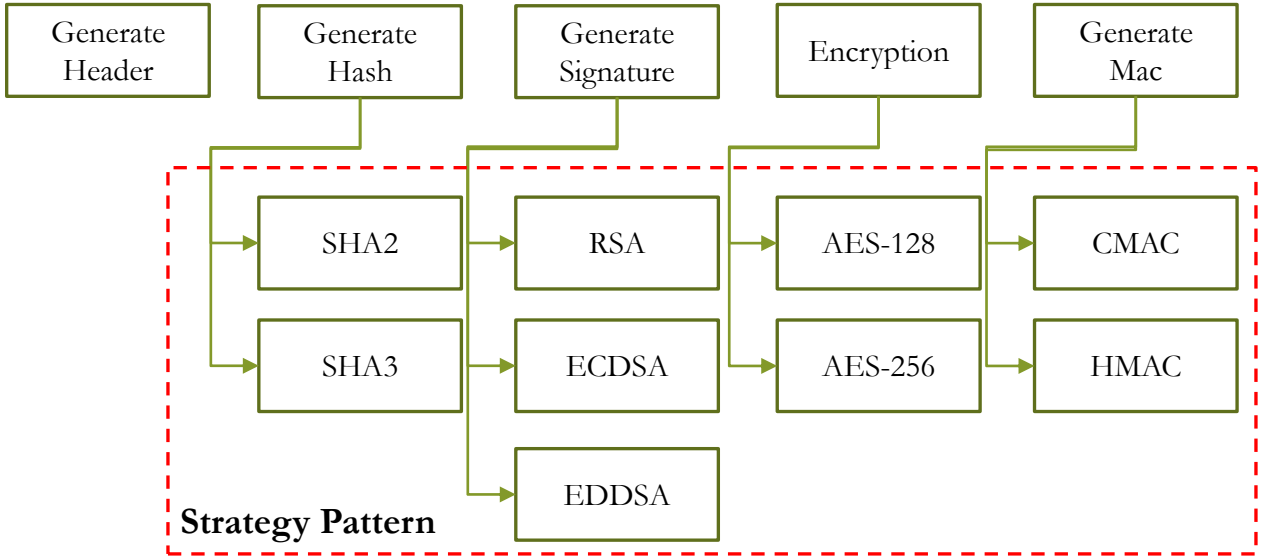
▪ 성과

- Secure Boot Solution에서 사용하는 다양한 Cryptography Algorithm을 Secure Boot Solution 코드의 수정 없이 추가 및 변경 가능
- Strategy Pattern 적용으로 기존 Solution 대비 소비되는 메모리 감소



▪ 개발

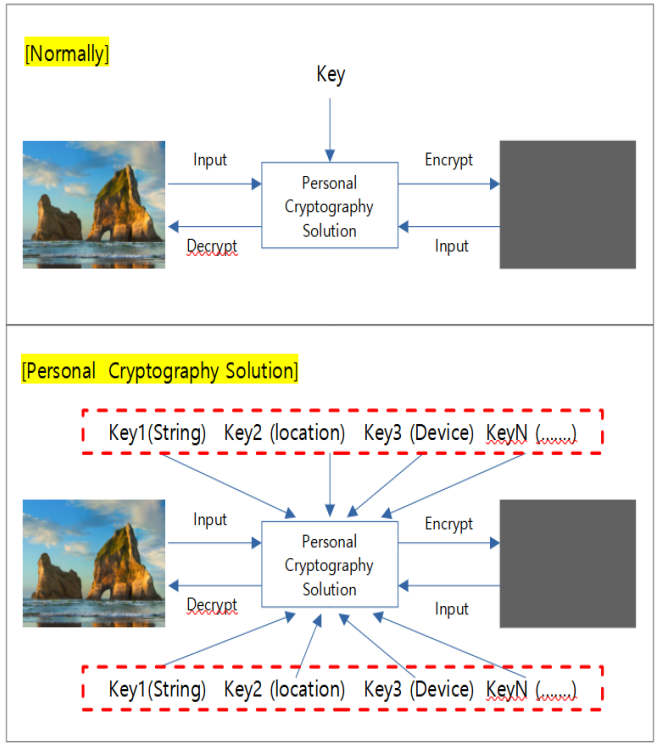
- Secure Boot를 위한 Secure Image 생성 과정은 Header 생성, Signature 생성, Encryption, MAC 생성 등으로 이루어짐
- 각 단계에서는 적용할 수 있는 알고리즘 군이 있고, 사용자에 따라 변경 및 신규추가가 가능
- 각 단계에서 사용하는 알고리즘을 SW Design 패턴 중 Strategy 패턴을 적용하여 알고리즘 변경 및 신규 알고리즘 추가 용이하게 개발



Personal Cryptography Solution Android Application – Side Project (1/2)

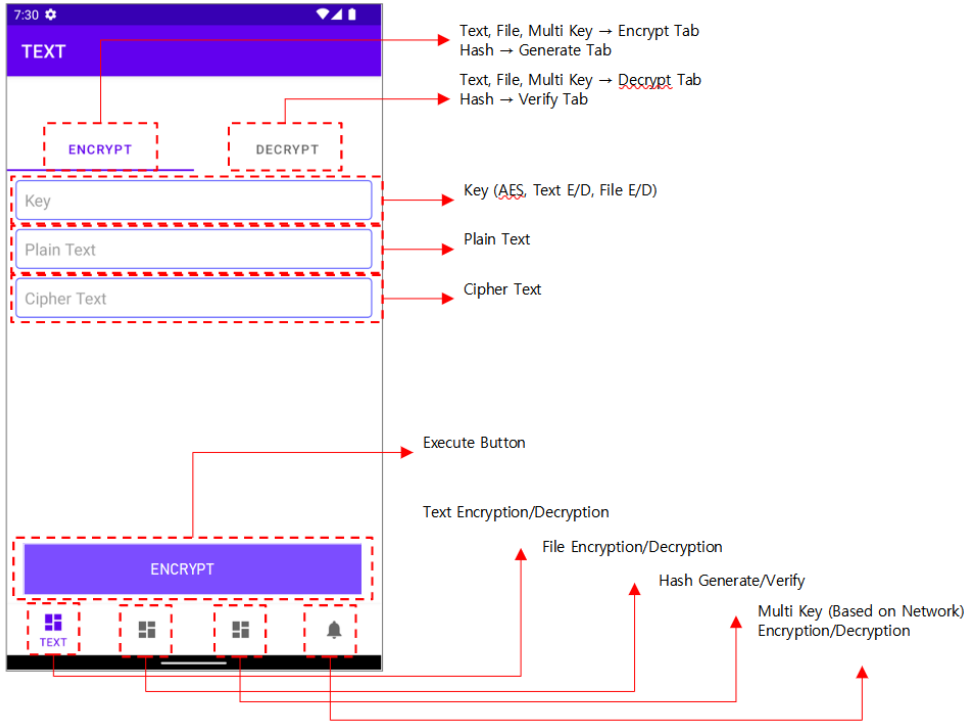
▪ 개요

- Text Encrypt/Decrypt 기능을 제공.
- File Encrypt/Decrypt 기능을 제공.
- 다중 키를 활용한 Encrypt/Decrypt 기능 제공.
- Hash 기능을 제공.
- 다중 키는 문자열, 위치, 디바이스가 될 수 있음.



▪ 개발

- 개발 언어는 Kotlin
- 각 Navi는 Fragment로 구성
- 각 Fragment는 Tab으로 구성
- 각 Fragment는 기능 Class를 통해 기능 구현



Personal Cryptography Solution Android Application – Side Project (2/2)

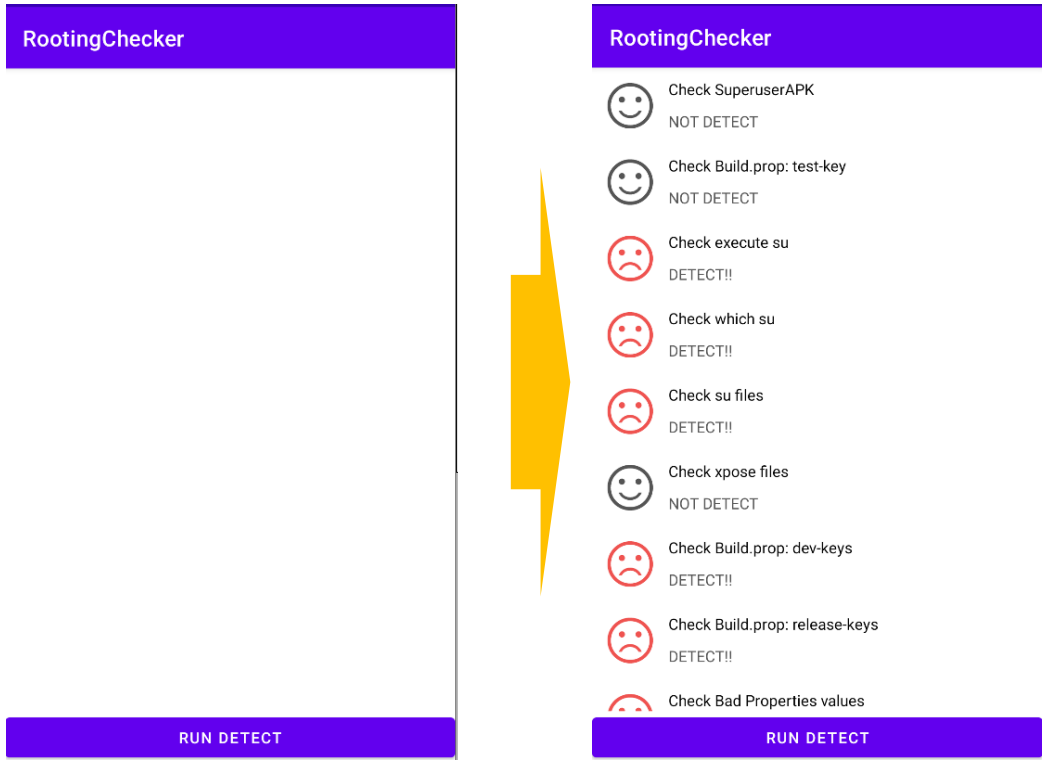
▪ 성과

- 기본적인 데이터, 파일의 암호화, 복호화, 해시 기능을 제공함으로 사용자는 쉽게 데이터의 기밀성 및 무결성을 강화 시킬 수 있음.
- 다중 키를 활용한 암호화, 복호화 기능을 통해 개인 자료의 기밀성이 강화될 것으로 예상됨
- 다수의 동의 후에 열람되어야 할 중요한 데이터의 경우 다중 키는 다수(여러 사람)의 키, 또는 위치, 근접한 장치 정보로 대체되어 사용할 수 있으므로 데이터의 기밀성이 강화될 수 있기 때문에, 비즈니스적으로 활용될 수 있을 것으로 예상됨

Rooting Checker – Side Project

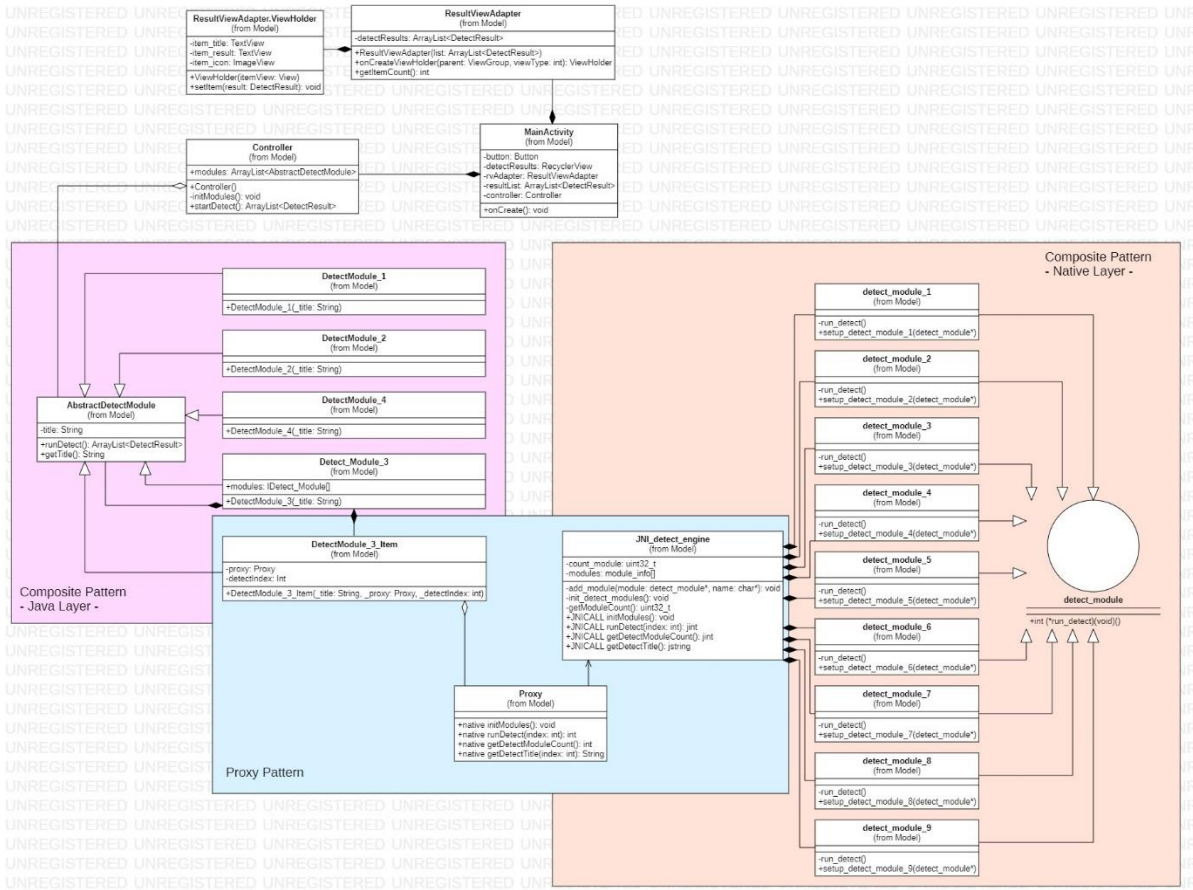
▪ 개요

- Android Rooting 여부를 확인
- 10여가지 Detect 방법을 통해 Rooting 여부를 확인



▪ 개발

- 개발 언어는 Java
- Java Layer와 Native Layer에 Composite Pattern을 적용하여 Java를 통한 Detect Module, C를 통한 Detect Module를 추가 가능



Fuzzing Test를 활용한 검증 강화

▪ 개요

- Random Test인 Fuzzing Test 도입을 통해 검증 능력을 높이고자 도입
- Fuzzing Test를 통해 예상하지 못한 값에 대한 예외처리 추가하고 SW 품질 향상을 목표로 함

▪ 성과

- Radamsa를 이용하여 개발된 "Random Input Generator"은 Option에 따라 길이, 특수문자 포함 여부, 문자, 숫자 등 다양한 입력을 생성할 수 있도록 개발되어, 향후 다양한 개발과제에서 사용 가능함.
- Random Input Generator와 Test Tool의 연동을 통해 다양한 Exception Case를 발견하였고, 이에 대한 예외처리를 통해 SW 품질을 향상시킴.
- Random Input Generator를 통한 Fuzzing Test를 개발자 검증 프로세스에 추가하여 신규 요구사항 개발 및 변경점에 대해 지속적인 SW 품질을 관리할 수 있게 됨.

▪ 개발

- 다양한 Fuzzing Test opensource 중에서 radamsa를 이용.
- Radamsa를 통해 다양한 입력을 생성함.
- 생성된 입력은 Test application의 입력이 됨