

**IoTyzzer CAT(Cryptographic Algorithm Test
서비스 사용자 매뉴얼)**

KMU iotyzzer-cat team

iotyzzer.cat@gmail.com

<제목 차례>

제 1 장 IoTyzer CAT 환경구성	4
제 1 절 IoTyzer CAT 클라이언트, libiotyzer 다운로드	4
제 2 절 IoTyzer CAT 클라이언트 개요	7
제 3 절 libiotyzer 개요	8
제 2 장 CAT를 위한 인터페이스 구성	9
제 1 절 libiotyzer 인터페이스 API 수정 방법	9
제 2 절 libiotyzer 빌드 & 설치	15
제 3 장 IoTyzer CAT 클라이언트 사용법	19
제 1 절 Generate test request	19
제 2 절 Generate test response	24

<표 차례>

표 1 query_blockcipher 함수의 매개변수	11
--------------------------------------	----

<그림 차례>

그림 1. libiotyzer 역할	4
그림 2 CAT 클라이언트, libiotyzer 다운로드 화면	5
그림 3. IoTyzer CAT 서비스를 위한 사전 준비 항목	5
그림 4. libiotyzer와 테스트 대상 암호모듈 간 인터페이스 개념도	9
그림 5 (예시) 사용자 암호알고리즘 소스코드, 헤더파일 추가	11
그림 6 query_blockcipher 함수의 매개변수	11
그림 7 define.h의 매크로	12
그림 8 query_blockcipher 함수 내부에 API 적용 예시	13
그림 9 libiotyzer-masterWx64WDebugWlibioTyzer.dll	15
그림 10 libiotyzer-masterWx64WReleaseWlibioTyzer.dll	16
그림 11 dll 파일을 client-mainWlib로 이동	16
그림 12 Powershell 실행	17
그림 13 Powershell 창	17

그림 14 Powershell에서 IoTyzerClient.exe 실행	18
그림 15 Main Menu	19
그림 16 Cryptography Algorithm Menu	20
그림 17 Modes of Operation Menu	21
그림 18 Blockcipher Menu	22
그림 19 Key Size Menu	22
그림 20 IoTyzer CAVP Request	23
그림 21 생성된 .req 파일	23
그림 22 생성된 log 폴더	24
그림 23 생성된 .log 파일	24
그림 24 IoTyzer CAVP Response	27
그림 25 생성된 .res 파일	28

제 1 장 IoTyzer CAT 환경구성

제 1 절 IoTyzer CAT 클라이언트, libiotyzer 다운로드

IoTyzer CAT 서비스를 이용하기 위해서는 CAT 서버와 통신할 수 있는 클라이언트, 사용자의 암호모듈과 상호작용하여 암호알고리즘에 대한 challenge와 response를 전달받을 수 있는 사용자 인터페이스가 필요하다.

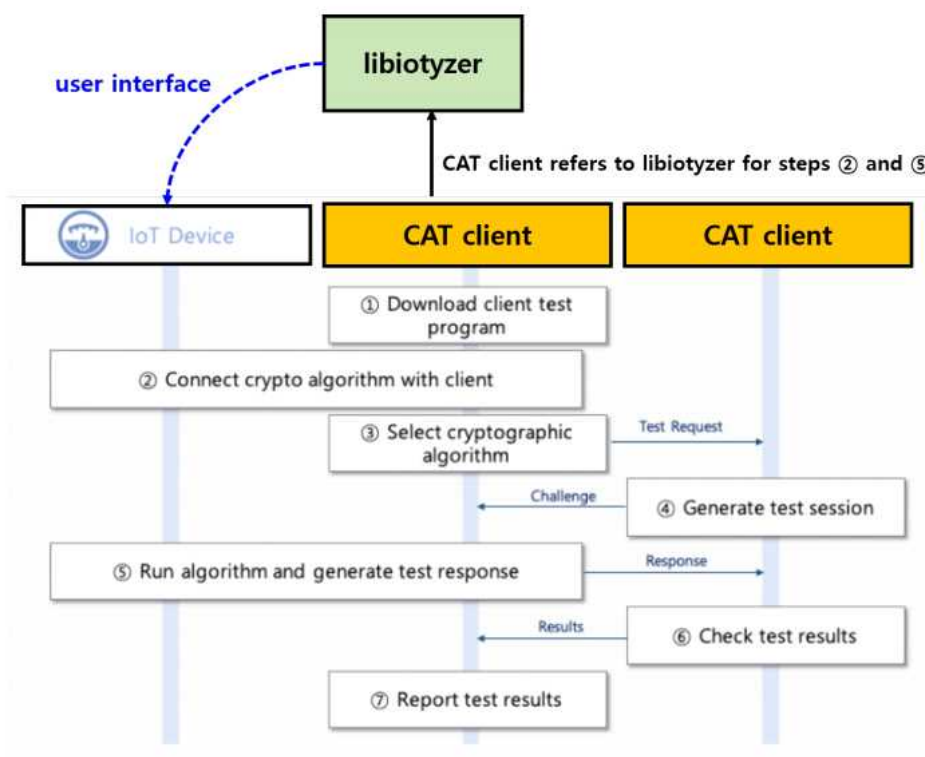


그림 1. libiotyzer 역할

위 그림은 libiotyzer가 CAT 서비스에서 수행하는 역할을 나타냅니다. libiotyzer CAT 서비스는 사용자에게 CAT 클라이언트와 사용자가 테스트

하고자 하는 암호알고리즘들이 포함된 구현물과의 인터페이스를 제공한다.

서비스를 이용하시는 사용자는 다음 [링크](#)로 접속하여 Step1. Download test client에 명시된 링크를 통해 아래와 같이 CAT 클라이언트, libiotyzer를 다운받을 수 있는 창으로 이동할 수 있다.

Download			
Client & Library			
Download the test client from the link according to your operating environment.			
Link	Type	OS	Description
Client	.zip	Windows & Linux	IoTyzer CAT client binary - IoTyzerClient.exe(Windows) - IoTyzerClient(Linux)
Library	build system	Windows & Linux	IoTyzer CAT Library build system - Visual studio solution(Windows) - Makefile(Linux)
Documentations			
Link	Type	Description	
Download	pdf	User Manual	

그림 2 CAT 클라이언트, libiotyzer 다운로드 화면

현재 배포되는 CAT 클라이언트, libiotyzer 버전은 Windows 10, Ubuntu 에서 동작할 수 있다.

환경	다운로드 항목
Windows 10	- libiotyzer 빌드 프로젝트(visual studio) - CAT 클라이언트 실행파일(.exe)
Ubuntu	- libiotyzer 빌드 프로젝트(makefile) - CAT 클라이언트 실행파일(elf)

그림 3. IoTyzer CAT 서비스를 위한 사전 준비 항목

CAT 서비스를 사용하길 원하는 사용자는 다운로드한 libiotyzer, CAT클라이언트를 적절히 빌드, 실행할 수 있는 환경을 구성하고난 후 테스트

하고자 하는 암호모듈의 형태에 따라 CAT 클라이언트와의 인터페이스를 구성할 수 있다.

제 2 절 IoTyzer CAT 클라이언트 개요

IoTyzer CAT 클라이언트는 사용자가 개발한 암호모듈, IoTyzer CAT 서버와 상호작용하여 암호알고리즘들의 구현정확성 테스트를 가능하게 하는 응용프로그램입니다. 사용자가 IoTyzer CAT 클라이언트를 통해 암호알고리즘의 구현정확성을 테스트하는 과정은 다음과 같다.

1. libiotyzer를 수정, 빌드하여 암호모듈과의 인터페이스 구성
2. 테스트하고자 하는 암호알고리즘을 선택하여 IoTyzer CAT 서버에 해당 암호알고리즘들에 대한 테스트 벡터
3. 수신한 테스트 벡터를 암호모듈에 입력하여 얻은 출력들을 파일로 저장(response 파일)
4. response 파일을 IoTyzer CAT 서버에 제출
5. 암호알고리즘 구현정확성 테스트 성공/실패 결과 확인

제 3 절 libiotyzer 개요

libiotyzer는 IoTyzer CAT 클라이언트 동작 시 함께 로드되어 사용자가 구현정확성 테스트를 할 수 있게 하는 동적 라이브러리이다.

본 문서에서는 예제를 통해 사용자가 암호모듈과 IoTyzer CAT 클라이언트 사이에 인터페이스를 구성하는 방법을 설명한다.

제 2 장 CAT를 위한 인터페이스 구성

제 1 절 libiotyzer 인터페이스 API 수정 방법

본 장에서는 libiotyzer를 사용하여 IoTyzer CAT 클라이언트와 테스트 대상이 되는 암호모듈 간에 인터페이스를 구성하는 방법을 설명한다. 다음 그림은 IoTyzer CAT 클라이언트, libiotyzer, 테스트 대상 암호모듈 간에 상호작용 방법을 나타낸다.

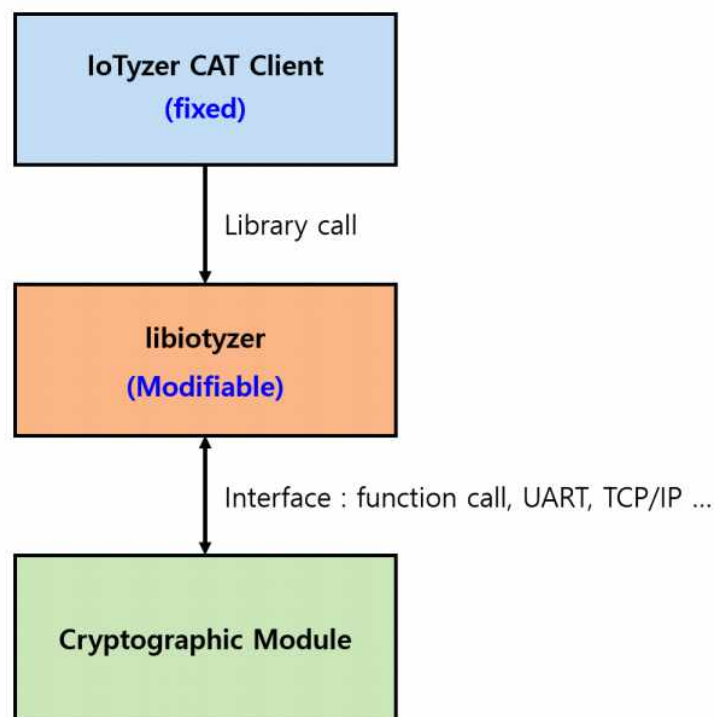


그림 4. libiotyzer와 테스트 대상과 암호모듈 간 인터페이스 개념도

먼저, 위 그림에서 libiotyzer는 사용자가 수정, 빌드를 자유롭게 할 수 있는 소프트웨어 라이브러리 소스코드 형태로 배포됩니다. 사용자는 loTyzer CAT 클라이언트가 제공하는 여러가지 암호알고리즘들에 대한 테스트를 사용하기 위해, libiotyzer를 수정해야 한다.

사용자에게 최초로 배포되는 libiotyzer 소스코드에는 각 암호알고리즘들에 대한 시험을 수행하는 함수들이 선언되어 있으며, 로직은 작성되어 있지 않습니다. 사용자는 테스트하고자 하는 암호모듈의 형태에 따라 해당 함수들의 로직을 작성하여 CAT 서비스를 사용할 수 있다. 자세한 사용방법은 "1.Example. Blockcipher ECB interface API"을 참고한다.

*** 주의사항 : query blockcipher modes와 같은 인터페이스 API는 배포시 선언된 상태로 사용자에게 제공되므로 API의 로직 이외에 파라미터, 인터페이스 API 명을 절대 수정해서는 안된다.**

1. Example. Blockcipher ECB interface API

libiotyzer 인터페이스 API를 수정하여 구현한 암호화 알고리즘을 테스트해 보는 순서는 다음과 같다.

가. 제 1장 1절을 통해 설치한 libiotyzer-master 디렉토리의 libioTyzer.sln을 실행시킨다.

나. libioTyzer.sln(솔루션 파일) 내부의 소스 파일 libiotyzer.c에 구현된 query_blockcipher 함수에 구현한 암호화 알고리즘을 적용시킨다.

다. 테스트하고자 하는 암호화 알고리즘을 query_blockcipher 함수 내부에 적용하기 위해 사전 구현한 암호화 알고리즘의 소스 파일과 헤더 파일을 솔루션 파일에 추가한다. (그림 5 (예시) 사용자 암호 알고리즘 소스코드, 헤더 추가 예시)

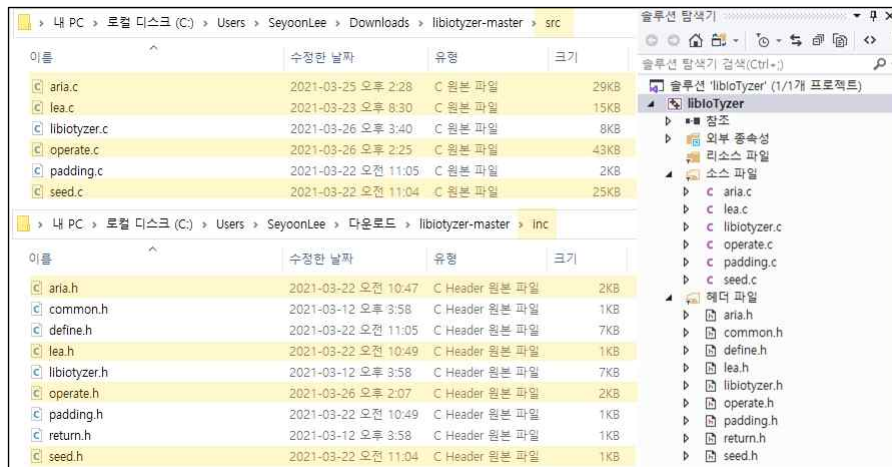


그림 5 (예시) 사용자 암호알고리즘 소스코드, 헤더파일 추가

라. 구현한 알고리즘을 테스트해보기 위해서는 query_blockcipher 함수의 매개변수에 맞게 구현한 암호 알고리즘의 매개변수에 적용시켜야 한다. 먼저, query_blockcipher 함수의 매개변수가 어떻게 (표 1 query_blockcipher 함수의 매개변수)를 참조하여 정의되어 있는지 살펴보자. (그림 6 query_blockcipher 함수의 매개변수 참조)

```

IOTZ_RETURN query_blockcipher(
    IOTZ_UBYTE* out,           // Byte array(Ciphertext)
    IOTZ_INT* outLen,          // Length is byte length
    const IOTZ_UBYTE* in,      // Byte array(Plaintext)
    const IOTZ_INT inLen,      // Length is byte length
    const IOTZ_UBYTE* iv,      // Byte array(IV or Nonce for CBC, CTR, OFB, if not NULL)
    const IOTZ_INT ivLen,      // Length is byte length
    const IOTZ_UBYTE* key,      // Byte array(Key)
    const IOTZ_INT keyLen,     // Length is byte length, ex) 16, 24, 32
    const IOTZ_BC_MODE mode,   // use define.h "IOTZ_BC_MODE" variables
    // IOTZ_ECB : ECB
    // IOTZ_CBC : CBC
    // IOTZ_CTR : CTR
    // IOTZ_OFB : OFB
    const IOTZ_BC_ALG alg,     // use define.h "IOTZ_BC_ALG" variables
    // IOTZ_ARIA : ARIA
    // IOTZ_SEED : SEED
    // IOTZ_LEA : LEA
    const IOTZ_BC_ENC_DEC oper // Encrypt(IOTZ_ENC), Decrypt(IOTZ_DEC)
)

```

그림 6 query_blockcipher 함수의 매개변수

자료형	변수명	설명
IOTZ_UBYTE*	out	출력값.
IOTZ_INT	outlen	출력값의 길이
IOTZ_UBYTE*	in	입력값
IOTZ_INT	inLen	입력값의 길이
IOTZ_UBYTE*	iv	초기벡터(Initializtion Vector)
IOTZ_INT	ivLen	초기벡터의 길이
IOTZ_UBYTE*	key	암 · 복호화 키
IOTZ_INT	keyLen	키의 길이
IOTZ_BC_MODE	mode	블록암호 운용모드
IOTZ_BC_ALG	alg	블록암호 알고리즘
IOTZ_BC_ENC_DEC	oper	암 · 복호화 동작

표 1 query_blockcipher 함수의 매개변수

추가적으로 자료형의 IOTZ_UBYTE는 unsigned char이며 IOTZ_INT는 unsigned int 그리고 IOTZ_BC_MODE, IOTZ_BC_ALG, IOTZ_BC_ENC_DEC는 enum 형식으로 사용하고자하는 운용모드, 알고리즘, 동작에 따라 define.h에 구현되어 있는 매크로를 사용하면 된다. (그림 7 define.h의 매크로) 예를 들어 ARIA 블록 암호 알고리즘을 ECB 운용모드를 이용하여 암호화 동작에 사용하고 싶다면, IOTZ_BC_MODE : IOTZ_ECB, IOTZ_BC_ALG : IOTZ_ARIA, IOTZ_BC_ENC_DEC : IOTZ_ENC 매크로를사용

```

typedef enum _IOTZ_BC_MODE
{
    IOTZ_BC_MODE_NONE = IOTZ_NONE,
    IOTZ_ECB,
    IOTZ_CBC,
    IOTZ_CTR,
    IOTZ_OFB,
} IOTZ_BC_MODE;

typedef enum _IOTZ_BC_ALG
{
    IOTZ_BC_ALG_NONE = IOTZ_NONE,
    IOTZ_ARIA,
    IOTZ_SEED,
    IOTZ_LEA,
    IOTZ_AES,
} IOTZ_BC_ALG;

typedef enum _IOTZ_BC_KEY_SIZE
{
    IOTZ_BC_KEY_SIZE_NONE = IOTZ_NONE,
    IOTZ_BC_128_BIT_KEY = IOTZ_SYMMETRIC_KEY_128_BIT_SIZE,
    IOTZ_BC_192_BIT_KEY = IOTZ_SYMMETRIC_KEY_192_BIT_SIZE,
    IOTZ_BC_256_BIT_KEY = IOTZ_SYMMETRIC_KEY_256_BIT_SIZE,
} IOTZ_BC_KEY_SIZE;

typedef enum _IOTZ_BC_ENC_DEC
{
    IOTZ_ENC = 0,
    IOTZ_DEC,
} IOTZ_BC_ENC_DEC;

```

그림 7 define.h의 매크로

하면 된다.

2.3. query_blockcipher 함수의 매개변수에 맞게 구현한 암호 알고리즘의 매개변수에 적용시킨다. (그림 8. query_blockcipher 함수 내부에 API 적용 예시 참조)

```
IOTZ_RETURN query_blockcipher(
    IOTZ_UBYTE* out,           // Byte array(Ciphertext)
    IOTZ_INT* outLen,          // Length is byte length
    const IOTZ_UBYTE* in,      // Byte array(Plaintext)
    const IOTZ_INT inLen,      // Length is byte length
    const IOTZ_UBYTE* iv,      // Byte array(IV or Nonce for CBC, CTR, OFB, if not NULL)
    const IOTZ_INT ivLen,      // Length is byte length
    const IOTZ_UBYTE* key,      // Byte array(Key)
    const IOTZ_INT keyLen,      // Length is byte length, ex) 16, 24, 32
    const IOTZ_BC_MODE mode,    // use define.h "IOTZ_BC_MODE" variables
                                // IOTZ_ECB : ECB
                                // IOTZ_CBC : CBC
                                // IOTZ_CTR : CTR
                                // IOTZ_OFB : OFB
    const IOTZ_BC_ALG alg,      // use define.h "IOTZ_BC_ALG" variables
                                // IOTZ_ARIA : ARIA
                                // IOTZ_SEED : SEED
                                // IOTZ_LEA : LEA
    const IOTZ_BC_ENC_DEC oper // Encrypt(IOTZ_ENC), Decrypt(IOTZ_DEC)
)
{
    if (alg == IOTZ_ARIA) {
        if (mode == IOTZ_ECB) {
            if (oper == IOTZ_ENC)
                ECB_ENC(in, out, inLen, outLen, IOTZ_ARIA, key, keyLen);
        }
        else if (mode == IOTZ_CBC) {
            if (oper == IOTZ_ENC)
                CBC_ENC(in, out, inLen, outLen, IOTZ_ARIA, key, keyLen, iv);
        }
    }
}
```

그림 8 query_blockcipher 함수 내부에 API 적용 예시

query_blockcipher 함수의 매개변수 중 const 자료형 매개변수들은 libiotyzer에서 실험해 보고자 하는 값들을 제시해준다. 따라서 사용자는 제시되는 값들이 구현한 암호화 알고리즘에 적용될 수 있도록 자료형과 값을 정확하게 맞춰 줘야 한다. 예를 들어, 키의 길이를 bit로 맞춰 주는 게 아닌 byte로 맞춰줘야 한다. 128-bit 암호화 키를 사용한다면, bit 형식인 128을 keyLen에 대입시키는 게 아닌 byte 형식인 16으로 대입시켜야 한다.

(그림 8. query_blockcipher 함수 내부에 API 적용 예시)에서 API 적용 명세는 다음과 같다.

- 암호 알고리즘 : ARIA
- 운용모드 : ECB, CBC

- 구현 함수 : ECB_ENC, CBC_ENC

ECB_ENC 함수에는 query_blockcipher로부터 입력값(in), 입력값의 길이(inLen), 사용 암호알고리즘(oper), 키(key), 키 길이(keyLen), 출력값(out), 출력값의 길이(outLen)를 함수에 대입한다. 이 후 ECB_ENC 함수 내부 동작을 통해 출력값에는 입력한 정보들을 통해 입력값(in)의 암호문으로, 출력값의 길이(outLen)에는 출력값의 byte 길이를 대입시켜 준다. CBC_ENC 함수에서는 ECB_ENC 함수와 동일하게 query_blockcipher로부터 입력값(in), 입력값의 길이(inLen), 사용 암호알고리즘(oper), 키(key), 키 길이(keyLen), 출력값(out), 출력값의 길이(outLen)를 함수에 대입한다. 추가적으로 초기벡터(iv)도 대입시킨다. 이 후 CBC_ENC 함수 내부 동작을 통해 출력값에는 입력한 정보들을 통해 입력값(in)의 암호문으로, 출력값의 길이(outLen)에는 출력값의 byte 길이를 대입시켜준다. (그림 2. query_blockcipher 함수 내부에 API 적용 예시)에는 예시가 없지만, 다른 블록암호나 운용모드를 사용시 이와 같이 query_blockcipher 함수 내부에 구현한 암호알고리즘 API를 알맞게 적용시켜 주면 된다.

제 2 절 libiotyzer 빌드 & 설치

1. libioTyzer build

- 가. 2장 1절에서 API를 수정한 뒤, 솔루션을 빌드(F7)한다. (빌드가 실패할 시, 출력된 오류 목록에 따라 오류를 수정하여 성공하도록 한다.)
- 나. Debug 모드로 빌드가 성공할 시, libiotyzer-master\wx64\Debug 디렉토리에 성공한 시간에 맞춰 libioTyzer.dll 라이브러리가 생성된다. (그림 9. libiotyzer-master\wx64\Debug\libioTyzer.dll 참조)

이름	수정된 날짜	유형	크기
libioTyzer.tlog	2021-03-25 오후 2:59	파일 폴더	
aria.obj	2021-03-25 오후 2:28	3D Object	62KB
lea.obj	2021-03-23 오후 8:30	3D Object	54KB
libioTyzer.dll	2021-03-25 오후 2:59	응용 프로그램 확장	191KB
libioTyzer.exp	2021-03-25 오후 2:41	Exports Library File	2KB
libioTyzer.lib	2021-03-25 오후 2:59	Incremental Linke...	457KB
libioTyzer.log	2021-03-22 오전 11:05	Object File Library	3KB
libioTyzer.obj	2021-03-25 오후 2:59	텍스트 문서	4KB
libioTyzer.pdb	2021-03-25 오후 2:59	3D Object	17KB
libioTyzer.vcxproj.FileListAbsolute.txt	2021-03-22 오전 11:05	Program Debug ...	540KB
operate.obj	2021-03-25 오후 2:59	텍스트 문서	1KB
padding.obj	2021-03-22 오전 11:05	3D Object	91KB
seed.obj	2021-03-22 오전 11:05	3D Object	11KB
vc142.idb	2021-03-25 오후 2:59	3D Object	38KB
vc142.pdb	2021-03-25 오후 2:59	VC++ Minimum ...	107KB
		Program Debug ...	84KB

그림

9

libiotyzer-master\wx64\Debug\libioTyzer.dll

2. Release 모드로 빌드가 성공할 시, libiotyzer-master\wx64\Release 디렉토리에 성공한 시간에 맞춰 libioTyzer.dll 라이브러리가 생성된다. (참조 :그림 10. libiotyzer-master\wx64\Release\libioTyzer.dll)

이름	수정된 날짜	유형	크기
libioTyzer.tlog	2021-03-26 오전 11:49	파일 폴더	
aria.obj	2021-03-26 오전 11:49	3D Object	127KB
lea.obj	2021-03-26 오전 11:49	3D Object	118KB
libioTyzer.dll	2021-03-26 오전 11:49	응용 프로그램 확장	33KB
libioTyzer.exp	2021-03-26 오전 11:49	Exports Library File	2KB
libioTyzer.iobj	2021-03-26 오전 11:49	IOBJ 파일	98KB
libioTyzer.ipdb	2021-03-26 오전 11:49	IPDB 파일	34KB
libioTyzer.lib	2021-03-26 오전 11:49	Object File Library	3KB
libioTyzer.log	2021-03-26 오전 11:49	텍스트 문서	5KB
libioTyzer.obj	2021-03-26 오전 11:49	3D Object	76KB
libioTyzer.pdb	2021-03-26 오전 11:49	Program Debug ...	380KB
libioTyzer.vcxproj.FileListAbsolute.txt	2021-03-26 오전 11:49	텍스트 문서	1KB
operate.obj	2021-03-26 오전 11:49	3D Object	172KB
padding.obj	2021-03-26 오전 11:49	3D Object	83KB
seed.obj	2021-03-26 오전 11:49	3D Object	103KB
vc142.pdb	2021-03-26 오전 11:49	Program Debug ...	84KB

그림 10 libiotyzer-master\wx64\Release\libioTyzer.dll

3. 생성된 libioTyzer.dll을 client-main\lib 디렉토리로 이동시킨다. (그림 11 dll 파일을 client-main\lib로 이동 참조)



그림 11 dll 파일을 client-main\lib로 이동

4. Powershell을 실행시키기 위해 실행창(윈도우 키 + R)을 실행시키고 powershell을 입력하여 Powershell을 실행시킨다. (그림 12. Powershell 실행, 그림 13. Powershell 창 참조)

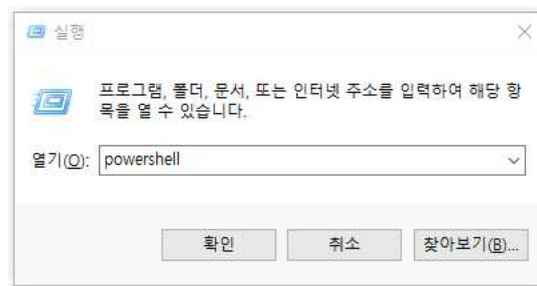


그림 12 Powershell 실행

5. Powershell 창을 실행시킨 뒤, 설치한 client-main의 디렉토리로 이동한다. 이 후, IoTyzerClient.exe를 실행시킨다. (그림 14. Powershell에서 IoTyzerClient.exe 실행 참조)


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6

PS C:\Users\GeyoonLee> cd C:\Users\GeyoonLee\Downloads\client-main\
PS C:\Users\GeyoonLee\Downloads\client-main> ls

디렉터리: C:\Users\GeyoonLee\Downloads\client-main

Mode                LastWriteTime         Length Name
----                -
d-----        2021-03-26 오후 2:22           .cavo
d-----        2021-03-26 오후 2:34           lib
-a-----        2021-02-26 오후 8:36              9 .gitignore.txt
-a-----        2021-02-26 오후 8:36          75762 IoTyzerClient
-a-----        2021-02-26 오후 8:36        201216 IoTyzerClient.exe
-a-----        2021-02-26 오후 8:36           55 README.md

PS C:\Users\GeyoonLee\Downloads\client-main>
PS C:\Users\GeyoonLee\Downloads\client-main> .\IoTyzerClient.exe -f

=====
Start IoTyzer Client!
IoTyzer Client v0.4.17 (2021.02.25)
Copyright (C) 2021 kookmin University & EIL. All rights reserved.
=====

Main Menu
=====
Select menu.
[1] Load IoT target(Initial Process)
[2] Request CAVP test
[3] Response CAVP test
[4] Release IoT target(Terminate Process)
[5] Exit
=====
Choice: [1-5]
```

그림 13 Powershell에서 IoTyzerClient.exe 실행

제 3 장 IoTyzer CAT 클라이언트 사용법

제 1 절 Generate test request

1. 클라이언트 메뉴 선택

가. Main Menu

IoTyzerClient.exe를 실행하면 다음과 같은 메뉴 선택창이 나타난다.
(그림 15 참조) CAVP 검증을 위한 테스트 벡터 요청을 하고 싶은 경우
2번을 입력한다.

```
-----  
Select menu.  
[1] Load IoT target(Initial Process)  
[2] Request CAVP test  
[3] Response CAVP test  
[4] Release IoT target(Terminate Process)  
[5] Exit  
-----
```

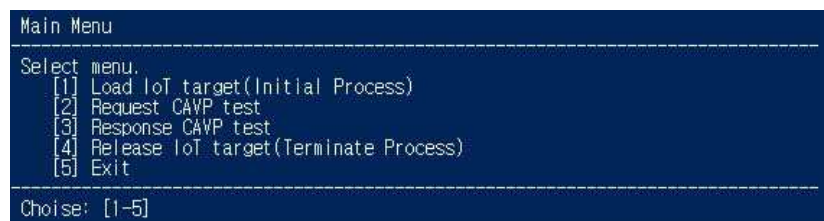



그림 14 Main Menu

나. Cryptography Algorithm Menu

Main Menu에서 2번을 입력한 경우, 다음과 같이 블록암호, 해시함수, 메시지 인증, 전자 서명 등 암호 알고리즘을 선택할 수 있는 입력창이 나타난다. (그림 16 참조)

Select menu.

- [0] Back to Previous Menu
 - [1] Blockcipher Cipher Modes
 - [2] Secure Hashing
 - [3] Message Authentication (Limited Service support)
 - [4] Random Number Generation (Limited Service support)
 - [5] Digital Signature
 - [6] Key Agreement Schemes
 - [7] Key Derivation Functions (Not Service)
-



```
Select menu.  
[0] Back to Previous Menu  
[1] Blockcipher Cipher Modes  
[2] Secure Hashing  
[3] Message Authentication (Limited Service support)  
[4] Random Number Generation (Limited Service support)  
[5] Digital Signature  
[6] Key Agreement Schemes  
[7] Key Derivation Functions (Not Service)  
Choise: [0-7]
```

그림 15 Cryptography Algorithm Menu

블록암호에 대한 검증 테스트 벡터를 요청하고 싶은 경우에는 1번을 입력한다. 이전 단계로 돌아가고 싶은 경우에는 0번을 입력한다.

다. Modes of Operation Menu

1번을 입력한 경우 다음과 같이 운용모드를 선택할 수 있는 입력창이 나타난다. (그림 17 참조)

Select menu.

[0] Back to Previous Menu

[1] ECB mode

[2] CBC mode

[3] CTR mode

[4] OFB mode

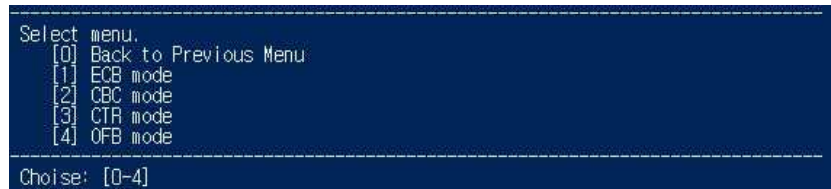


그림 16 Modes of Operation Menu

ECB mode의 테스트 벡터를 요청하고 싶은 경우에는 1번, CBC mode는 2번, CTR mode는 3번, OFB mode는 4번을 입력한다. CFB mode는 지원하지 않는다. 이전 단계로 돌아가고 싶은 경우에는 0번을 입력한다.

라. Blcokcipher Menu

운용모드에 대한 선택이 끝났다면 다음과 같이 블록암호에 대해 선택할 수 있는 입력창이 나타난다. (그림 18 참조)

Select menu.

- [0] Back to Previous Menu
 - [1] ARIA
 - [2] SEED
 - [3] LEA
-

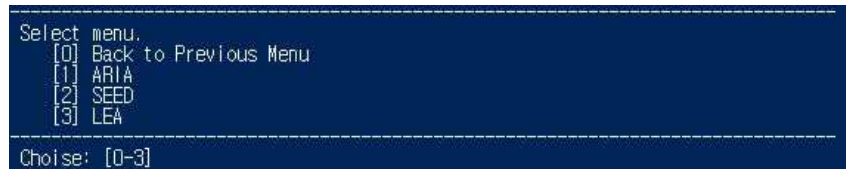


그림 17 Blockcipher Menu

ARIA에 대한 테스트 벡터를 요청하고 싶은 경우에는 1번, SEED는 2번, LEA는 3번을 입력한다. 이전 단계로 돌아가고 싶은 경우에는 0번을 입력한다.

마. Key Size Menu

블록암호까지 선택을 마쳤다면 다음과 같이 키 사이즈에 대해 선택할 수 있는 입력창이 나타난다. (그림 19 참조) SEED의 경우 키 사이즈가 128-bit로 고정이므로 이 과정 없이 바로 테스트 벡터를 요청하게 된다.

Select menu.

- [0] Back to Previous Menu
 - [1] 128-bit Key size
 - [2] 192-bit Key size
 - [3] 256-bit Key size
-

```
Select menu.
[0] Back to Previous Menu
[1] 128-bit Key size
[2] 192-bit Key size
[3] 256-bit Key size
-----
Choise: [0-3]
```

그림 18 Key Size Menu

키 사이즈가 128-bit인 경우에는 1번, 192-bit는 2번, 256-bit는 3번을 입력한다. 이전 단계로 돌아가고 싶은 경우에는 0번을 입력한다.

바. IoTyzer CAVP Request

모든 선택 과정을 마친 뒤 검증을 위한 테스트 벡터 요청이 시작된다. IoTyzer server와 연결하여 KAT, MMT, MCT 검증을 위한 테스트 벡터 파일을 요청한다. Request 성공 여부는 [OK], [Fail]로 판단한다. (그림 20 참조)

```
IoTyzer CAVP Request
IoTyzer CAVP Request connect server
IoTyzer CAVP Request connect server ..... [OK]
IoTyzer CAVP Request ARIA128(ECB) start
  ARIA128(ECB) req start
    ARIA128(ECB)KAT recv start
    ARIA128(ECB)KAT recv finish
    ARIA128(ECB)MMT recv start
    ARIA128(ECB)MMT recv finish
    ARIA128(ECB)MCT recv start
    ARIA128(ECB)MCT recv finish
  ARIA128(ECB) req finish
IoTyzer CAVP Request ARIA128(ECB) finish
IoTyzer CAVP Request ..... [OK]
```

그림 19 IoTyzer CAVP Request

IoTyzer CAVP Request가 성공했을 경우, client-main 폴더 아래에 생성해 주었던 .cavp 폴더에 .req 형식의 파일이 생성된다. (그림 21 참조)

ARIA128(ECB)KAT.req	2021-03-26 오후 1:46	REQ 파일	21KB
ARIA128(ECB)MCT.req	2021-03-26 오후 1:46	REQ 파일	1KB
ARIA128(ECB)MMT.req	2021-03-26 오후 1:46	REQ 파일	3KB

그림 20 생성된 .req 파일

Request CAVP test 과정이 끝난 후 다시 [] 선택창이 나타난다. 이후 Request CAVP test, Response CAVP test 등 원하는 메뉴를 선택하여 테스트를 진행한다.

2. 클라이언트 로그 확인

CAVP 검증용 테스트 벡터를 요청하는 과정의 로그를 기록하고 싶다면 실행 명령어에 -f [로그 파일이 저장될 폴더명]을 추가해야 한다.

ex) ./IoTyzerClient.exe -f log

-f: <filepath> Print log to file

이후 과정은 [Main Menu]과 동일하다. 로그 파일은 .log 형식으로 client-main 폴더 아래의 지정한 폴더에 저장된다. (그림 22, 그림 23 참조)

.cavp	2021-03-26 오후 1:57	파일 폴더	
lib	2021-03-22 오전 11:38	파일 폴더	
log	2021-03-26 오후 2:52	파일 폴더	
.gitignore.txt	2021-03-15 오후 6:45	텍스트 문서	1KB
IoTyzerClient	2021-03-15 오후 6:45	파일	74KB
IoTyzerClient.exe	2021-03-15 오후 6:45	응용 프로그램	72KB
README.md	2021-03-15 오후 6:45	Markdown 원본 ...	1KB

그림 21 생성된 log 폴더

IoTyzer.log	2021-03-26 오후 2:53	텍스트 문서	9KB
-------------	--------------------	--------	-----

그림 22 생성된 .log 파일

log 파일이 저장될 폴더는 client-main 폴더 아래에 있어야 하며, 만약 log 파일이 저장될 폴더가 없을 경우에는, 지정한 폴더명으로 폴더가 새

롭게 생성되어 그 안에 저장된다. 이미 .log 파일이 있는 경우, 그 아래에 추가적으로 log가 기록되는 방식으로 저장된다.

제 2 절 Generate test response

1. 클라이언트 메뉴 선택

가. Main Menu

IoTyzerClient.exe 실행 후 Request CAVP test가 끝나면 다음과 같은 메뉴 선택창이 나타난다. (그림 15 참조) Request CAVP test를 통해 받아 온 테스트 벡터로 검증을 하고 싶은 경우, 3번을 입력한다.

Select menu.

- [1] Load IoT target(Initial Process)
 - [2] Request CAVP test
 - [3] Response CAVP test
 - [4] Release IoT target(Terminate Process)
 - [5] Exit
-

나. Cryptography Algorithm Menu

Main Menu에서 3번을 입력한 경우, 다음과 같이 블록암호, 해시함수, 전자 서명 등 암호 알고리즘을 선택할 수 있는 입력창이 나타난다. (그림 11 참조)

Select menu.

- [0] Back to Previous Menu
 - [1] Blockcipher Cipher Modes
 - [2] Secure Hashing
 - [3] Message Authentication (Limited Service support)
 - [4] Random Number Generation (Limited Service support)
 - [5] Digital Signature
 - [6] Key Agreement Schemes
 - [7] Key Derivation Functions (Not Service)
-

블록암호에 대한 검증을 진행하고 싶은 경우, 1번을 입력한다. 이전 단계로 돌아가고 싶은 경우에는 0번을 입력한다.

다. Modes of Operation Menu

1번을 입력한 경우 다음과 같이 운용모드를 선택할 수 있는 입력창이 나타난다. (그림 17 참조)

Select menu.

- [0] Back to Previous Menu
 - [1] ECB mode
 - [2] CBC mode
 - [3] CTR mode
 - [4] OFB mode
-

ECB mode에 대한 검증을 요청하고 싶은 경우에는 1번, CBC mode는 경우에는 2번, CTR mode는 3번, OFB mode는 4번을 입력한다. CFB mode는 지원하지 않는다. 이전 단계로 돌아가고 싶은 경우에는 0번을

입력한다.

라. Blockcipher Menu

운용모드에 대한 선택이 끝났다면 다음과 같이 블록암호에 대해 선택할 수 있는 입력창이 나타난다. (그림 18 참조)

Select menu.

[0] Back to Previous Menu

[1] ARIA

[2] SEED

[3] LEA

ARIA에 대해 검증을 요청하고 싶은 경우에는 1번, SEED는 2번, LEA는 3번을 입력한다. 이전 단계로 돌아가고 싶은 경우에는 0번을 입력한다.

마. Key Size Menu

블록암호까지 선택을 마쳤다면 다음과 같이 키 사이즈에 대해 선택할 수 있는 입력창이 나타난다. (그림 19 참조) SEED의 경우 키 사이즈가 128-bit로 고정이므로 이 과정 없이 바로 검증을 진행하게 된다.

Select menu.

[0] Back to Previous Menu

[1] 128-bit Key size

[2] 192-bit Key size

[3] 256-bit Key size

키 사이즈가 128-bit인 경우에는 1번, 192-bit는 2번, 256-bit는 3번을

입력한다. 이전 단계로 돌아가고 싶은 경우에는 0번을 입력한다

바. IoTyzer CAVP Response

모든 선택 과정을 마친 뒤 검증이 시작된다. 단, CAVP Response 과정은 CAVP Request 과정에서 수신한 테스트 벡터 파일이 .cavp 폴더 내에 존재할 경우에만 가능하다.

수신한 테스트 벡터의 입력값을 이용하여 사용자의 알고리즘을 통해 출력값을 생성한 뒤, IoTyzer server와 연결하여 출력값이 일치하는지 비교하여 검증을 진행한다. KAT, MMT, MCT 각각의 검증 결과가 [OK] 또는 [Fail]로 나타난다. (그림 24 참조)

```
IoTyzer CAVP Response
IoTyzer CAVP Response connect server
IoTyzer CAVP Response connect server ..... [OK]
IoTyzer CAVP Response ARIA128(ECB) start
  ARIA128(ECB) rsp start
    Compare ARIA128(ECB)KAT rsp/fax ..... [OK]
    Compare ARIA128(ECB)MMT rsp/fax ..... [OK]
    Compare ARIA128(ECB)MCT rsp/fax ..... [OK]
  ARIA128(ECB) rsp finish
  IoTyzer CAVP Response ARIA128(ECB) finish
IoTyzer CAVP Response ..... [OK]
```

그림 23 IoTyzer CAVP Response

검증 성공 여부와는 별개로 IoTyzer CAVP Response가 성공했을 경우, client-main 폴더 아래에 생성해 주었던 .cavp 폴더에 .rsp 형식의 파일이 생성된다. 이 파일은 검증에 사용한 rsp 파일인 테스트 벡터에 사용자의 알고리즘을 통해 출력한 출력값이 추가되어 작성된 파일이다. (그림 25 참조)

ARIA128(ECB)KAT.req	2021-03-26 오후 2:35	REQ 파일	21KB
ARIA128(ECB)KAT.rsp	2021-03-26 오후 2:35	RSP 파일	32KB
ARIA128(ECB)MCT.req	2021-03-26 오후 2:35	REQ 파일	1KB
ARIA128(ECB)MCT.rsp	2021-03-26 오후 2:35	RSP 파일	13KB
ARIA128(ECB)MMT.req	2021-03-26 오후 2:35	REQ 파일	3KB
ARIA128(ECB)MMT.rsp	2021-03-26 오후 2:35	RSP 파일	4KB

그림 24 생성된 .res 파일

2. 클라이언트 로그 확인

CAVP Response 과정의 로그를 기록하고 싶다면 실행 명령어에 `-f` [log 파일이 저장될 폴더명]을 추가해야 한다.

ex) `./IoTyzerClient.exe -f log`
`-f: <filepath> Print log to file`

저장 방식은 [클라이언트 메뉴 선택]과 동일하며, 이후 진행 과정은 [클라이언트 로그 확인]과 동일하다.