

사물인터넷(IoT) 기반의 스트리밍 데이터 암호화

팀 명 : IoTCaps
지도 교수 : 양정모 교수님
팀 장 : 이기원
팀 원 : 윤대영
김혜영

목 차

1. 서론	
1-1 연구 배경	03
1-2 연구 필요성	03
1-3 주제 선정	03
2. 관련연구	
2-1 Media Source Extensions(MSE)	04
2-2 WebSocket	06
2-3 AES-CBC	07
3. 본론	
3-1 구상도 설명	08
3-2 회원가입	09
3-3 로그인	09
3-4 영상 접근 요청	12
3-5 영상 접근 수락	12
3-6 목록삭제	13
3-7 영상보기(잘못된 암호키)	13
3-8 영상보기(올바른 암호키)	13
3-9 시간동기/녹화	14
3-10 재생정보	14
3-11 리눅스에서 영상 전송	15
3-12 윈도우에서 영상 전송	16
3-13 IOTCAM에서 잘못된 아이디와 패스워드 입력 시	17
4. 결론	
4-1 결론 및 기대효과	18
5. 참고 자료	
5-1 참고자료	18
※ 첨부 자료	
프로그램 소스(IOTCAPS)	19
프로그램 소스(IOTCAM)	66
발표 자료	107

1. 서론

1-1 연구 배경

사물인터넷(IoT) 시대가 도래 하면서 모든 기기들이 인터넷과 연결하게 되었다. 이로 인해 인간 생활의 편리함과 여러 가지 응용기술들이 나오고 있다. 특히 영상 기술은 기존 아날로그 CCTV의 한정된 공간에서만 관찰 할 수 있다는 단점을 사물인터넷 시대에서는 CCTV에 네트워크 기능을 첨가하여 IP CCTV가 출시되면서 일반 사용자들도 저렴한 가격으로 IP CCTV를 접하게 되어 널리 사용할 수 있게 되었다.

1-2 연구 필요성

그러나 지금 IP CCTV들은 보안 요소를 고려하지 않아 여러 위협들이 존재하고 있다. 최근 언론에서는 IP CCTV 해킹과 관련된 기사가 나오며 해킹으로 인한 개인 정보 누출, 사생활 침해, 데이터 위조 및 변조 등의 위협에 놓이게 되었다.

또한 IP CCTV의 가장 핵심이라고 할 수 있는 스트리밍 기술이 발전하여 최근에는 유튜브 등의 동영상 서비스를 제공하는 곳에서도 DASH 기반의 영상 기술을 넣는 등 여러 IT분야에서 스트리밍 기술을 사용하고 있으나 영상 데이터 자체 암호화 기술은 아직 널리 사용되고 있지 않다. 현재 규모가 있는 동영상 서비스 업체에서는 TLS를 사용하여 영상 데이터를 전송을 하고 있지만 TLS전송망을 해킹하여 원본 데이터를 획득하게 되면 TLS전송 자체가 무력화되기 때문에 데이터 자체 암호화 기술은 매우 중요하다.

차세대 기술인 인공지능(AI) 분야에서도 스트리밍 기술이 필요하다. 이를 인공지능(AI)과 IP CCTV기술이 접목된 지능형 영상감시 시스템이라고 한다. 만약 지능형 영상감시 시스템이 악의적인 사용자에게 의해 공격을 당한다고 생각해보자. 크게는 인적 피해까지 발생할 있다. 예를 들어 교통 트래픽을 인공지능이 판단하여 제어하는 기술에서는 영상 데이터와 인공 지능 컴퓨터 간의 네트워크가 형성 되어야 하고 이 과정에서는 데이터 자체가 변조 되지 않아야 한다. 그러나 암호화 되지 않은 영상 데이터는 악의적인 사용자가 영상 데이터를 조작하게 되어 사회적으로 큰 문제를 야기할 수 있다.

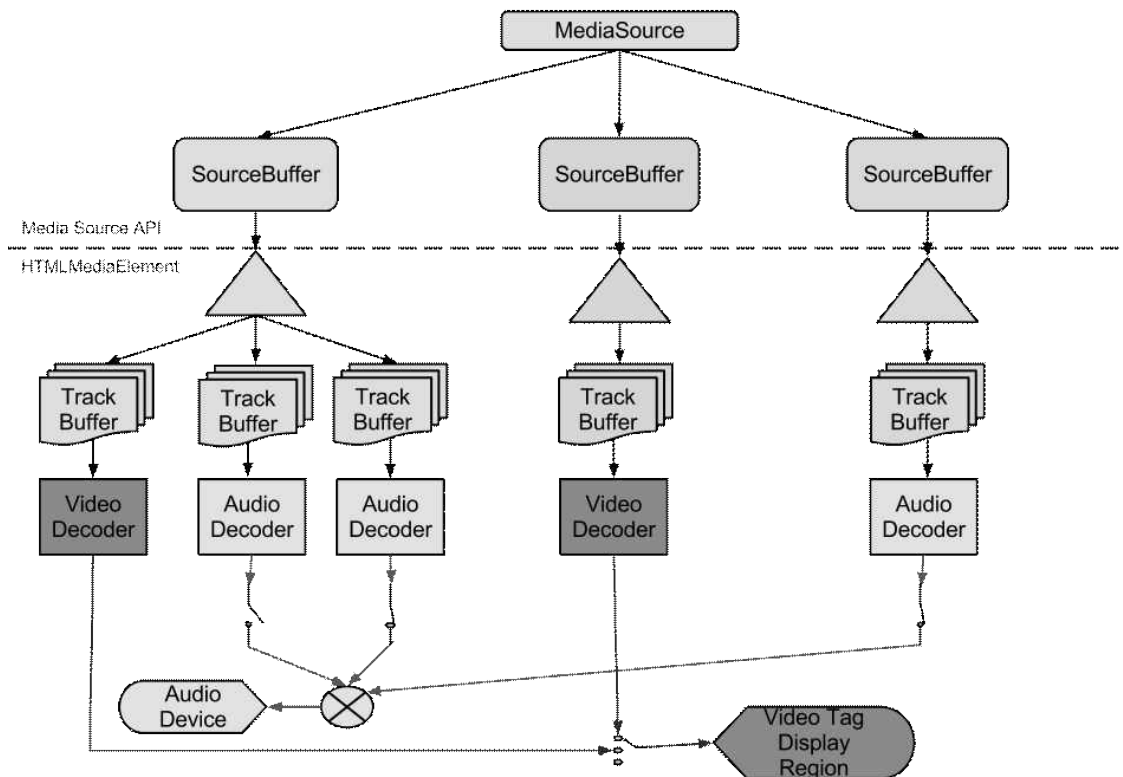
1-3 주제선정

IoTCaps는 AES-CBC 암호 알고리즘 기반으로 스트리밍의 원본 데이터인 영상 데이터를 암호화하는 기술을 개발하는 것을 주제로 선정하였고 해당 기술은 윈도우뿐만 아니라 라즈베리파이의 운영체제인 리눅스에서도 동작할 수 있도록 개발하였다.

2. 관련 연구

2-1 Media Source Extensions(MSE)

Media Source Extensions(MSE)는 자바 스크립트를 이용하여 오디오 및 비디오 스트림을 구성할 수 있게 하는 스트리밍 재생의 가장 기본이 되는 API다.



상단 그림은 MediaSource의 동작 구조를 도식화 하였다. SourceBuffer는 해당 영상이 어떤 코덱으로 구성되어있는지를 정의한다.

iotlive.js 코드 일부

```

if (playOption == "video")
sourceBuffer = mediaSource.addSourceBuffer('video/mp4; codecs="avc1.64001E"');
else
sourceBuffer = mediaSource.addSourceBuffer('video/mp4; codecs="avc1.64001E, mp4a.40.2"');
  
```

처음에는 mediaSource.addSourceBuffer를 하여 해당 영상이 어떤 MIME를 가지고 있는지 정의를 한다.

iotlive.js 코드 일부

<pre>if (playOption == "video") sourceBuffer.mode = 'sequence'; // 0초부터 시작 else sourceBuffer.mode = 'segments'; // chrome 50 이상 버전부터 필요(중간 시작)</pre>
--

또한 소스 버퍼의 모드를 설정하여 sequence(시퀀스)인지, segments(세그먼트)인지 정한다. 시퀀스 모드는 순차적으로 진행하는 모드로 처음부터 시작하고 세그먼트 모드는 영상의 중간부터 시작하게 된다. 그러나 스트리밍 데이터이므로 둘 다 영상의 중간부터 시작하나, 재생시간이 시퀀스 같은 경우 0초 부터시작, 세그먼트 같은 경우는 송출하고 있는 영상의 시간으로 재생된다.

iotlive.js 코드 일부

<pre>sourceBuffer.appendBuffer(queue.shift());</pre>
--

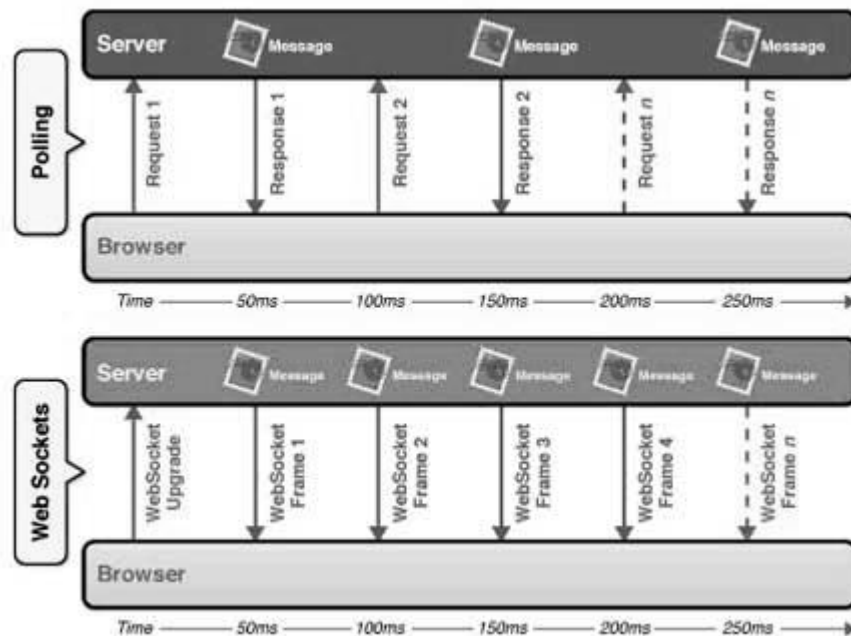
큐에 삽입된 데이터들을 FIFO(선입선출) 형태로 가져와서 appendBuffer를 하면 영상 데이터 재생이 시작된다.

iotlive.js 코드 일부

<pre>mediaSource.removeSourceBuffer(sourceBuffer);</pre>
--

소스 버퍼의 해제는 위와 같은 코드를 사용한다.

2-2 WebSocket



상단의 이미지는 WebSocket(웹소켓)과 Ajax Long Polling 방식을 비교한 이미지다. 웹 소켓은 기존의 요청-응답 관계 방식보다 쉽게 데이터를 교환할 수 있다.

웹소켓 API의 표준은 W3C에서 관장하고 프로토콜은 IETF에서 관장한다. 웹소켓은 HTTP의 요청과 마찬가지로 작동을 하지만 업그레이드 헤더를 사용하여 웹 서버에 요청한다.

이번 졸업 작품에서는 웹소켓에서 보안기능을 넣은 WSS를 사용한다. WSS는 HTTPS 처럼 전송 경로가 암호화 돼서 전송한다. 즉, IoTCaps에서 제작한 모든 프로그램은 WSS를 사용하며 데이터가 지나가는 경로는 TLS로 암호화 돼서 전송한다.

iotsession.js 코드 일부

```
var sessionSocket = io.connect('wss://shana.pe.kr:1115');
```

iotlive.js 코드 일부

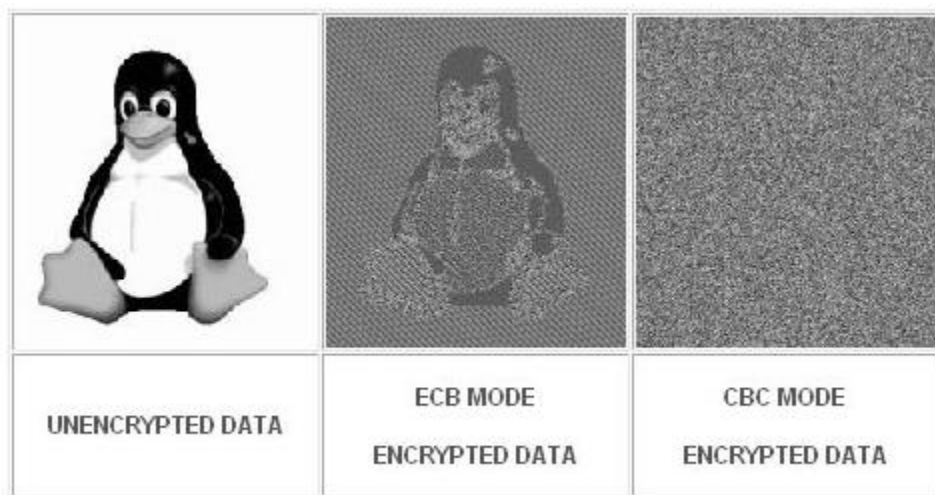
```
iotlive_conf("wss://shana.pe.kr:1120/", objvar.value);
```

위와 같이 wss를 사용하고 있다.

2-3 AES-CBC

AES암호화 기법은 미국 표준 기술 연구소 (NIST)가 5년의 표준화 과정을 거쳐 2002년 5월부터 암호화 기술 표준으로 채택한 기술이다. AES 암호의 안정성 검증 을 통해 미국, 유럽 각국의 정부 기관에서 가장 많이 사용하고 있다.

AES-CBC 모드는 가장 강력한 알고리즘으로 아래와 같이 CBC 모드는 16 바이트 단위로 암호화 한 데이터가 그 다음 16바이트 암호화에 적용되어 상호 연관 관계에 의해 원본 데이터의 중간 어느 부분만 따로 해석할 수 없는 구조이다.



AES 암호화 기술은 현재 기술로는 전문가 사이에서도 20~30년 정도는 해킹이 불가능할 것으로 예상하는 기술이다. 또한 AES로 암호화된 하드디스크는 폐기 처분이 필요한 경우 하드디스크만 빼서 버려도 안전할 정도이고, 현재 까지 나온 어떤 데이터 복구 소프트웨어 툴을 사용해도 복구가 불가능하다.

3. 본론

3-1 구상도 설명



사용자와 서버 간 기본적인 동작 구조는 다음과 같다.

- 사용자는 서버에게 인증을 요청한다.
- 서버는 사용자를 인증한다.
- 사용자는 서버에게 데이터를 요청한다.
- 서버는 사용자에게 데이터를 전송한다.

서버와 카메라 간 기본적인 동작 구조는 다음과 같다.

- 카메라는 서버에게 인증을 요청한다.
- 서버는 카메라를 인증한다.
- 카메라는 실시간으로 데이터를 서버에게 전송한다.

서버는 데이터를 중계해주는 역할을 한다.

복수의 카메라(또는 영상데이터)를 서버에 전송하는데 서버는 전송을 요청한 사용자의 아이디를 사용하여 방을 생성한다. 상단의 카메라와 서버 간 인증 과정은 아무나 영상을 전송할 수 없게 하기 위한 인증과 더불어 아이디를 기반으로 하는 방을 생성하는 과정이기도 하다. 카메라는 서버에게 영상을 전송하는데 서버에 있는 자신의 방에 영상을 전송하는 것이다.

사용자는 영상 데이터에 접근 시 서버의 방으로 접근한다.

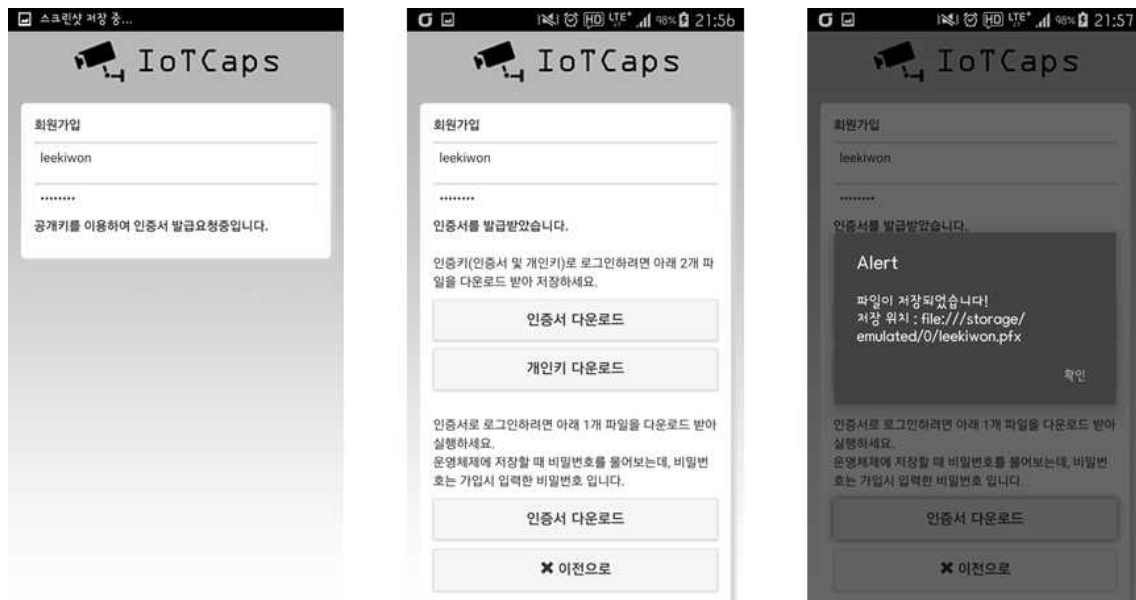
사용자를 인증하는 과정은 해당 영상을 볼 수 있는 권한을 갖고 있는지 여부를 서버의 데이터베이스에 있는 접근 제어 리스트를 통해 확인한다. 이 과정에서 접근 허가가 이루어지면 서버는 사용자를 방에 입장시킨다. 사용자가 해당 방에 들어가게 되면 데이터 요청을 할 수 있는데 데이터 요청을 하는 순간 서버가 데이터를 전송한다.

3-2부터 3-10까지는 영상을 보는 프로그램인 IOTCAPS에 대한 설명이다.

3-2 회원가입

회원가입은 앱을 통해서 할 수 있고 웹에서도 할 수 있다.

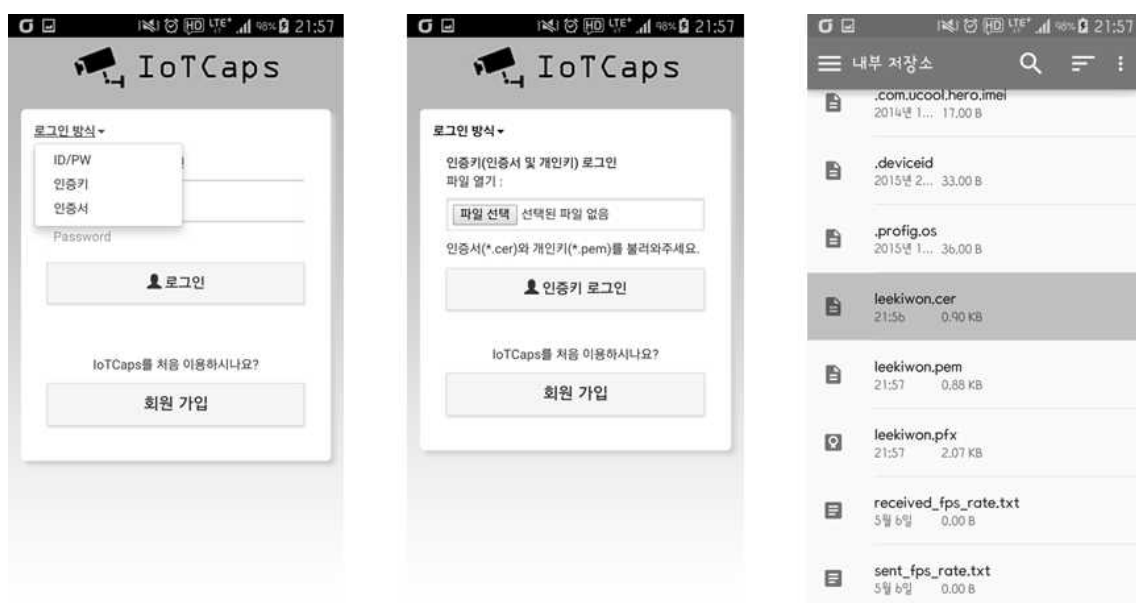
아래와 같이 회원가입을 요청하게 되면 인증서+개인키와 개인키가 들어있는 인증서를 받을 수 있게 된다.



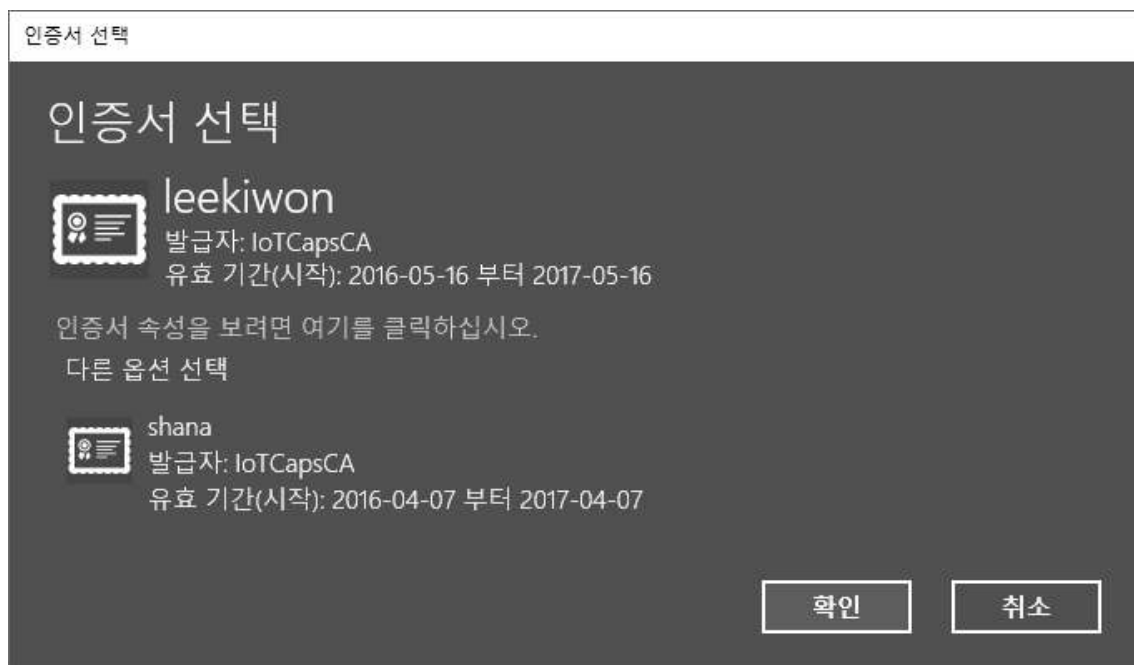
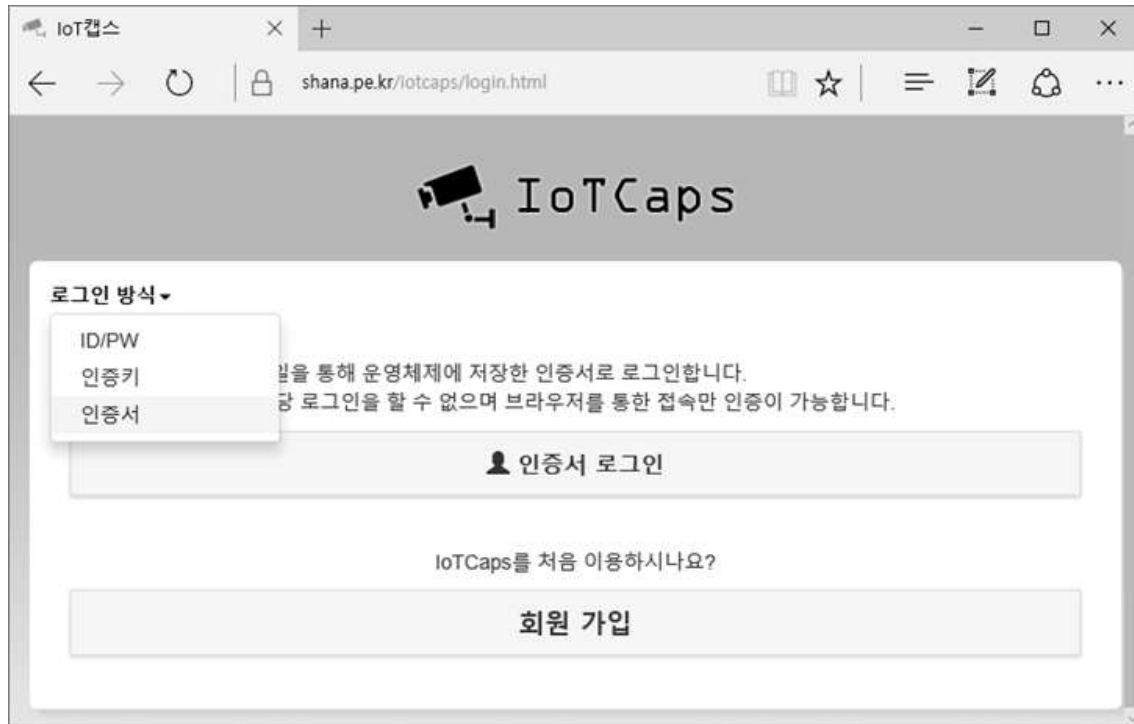
3-3 로그인

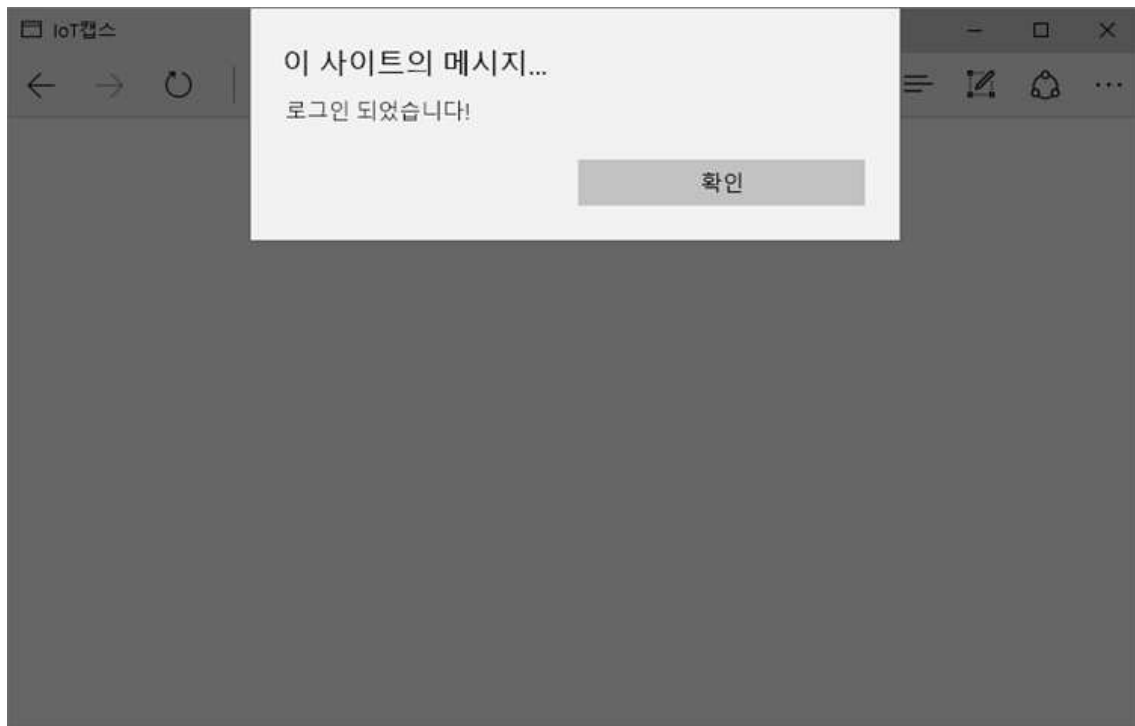
로그인 방식은 3가지가 있으며 ID/PW, 인증키, 인증서가 있다.

아래 스크린 샷은 인증키를 이용하여 로그인하는 스크린 샷이다. 인증키를 이용하여 로그인 하려면 pem과 cer파일을 불러와야 한다.

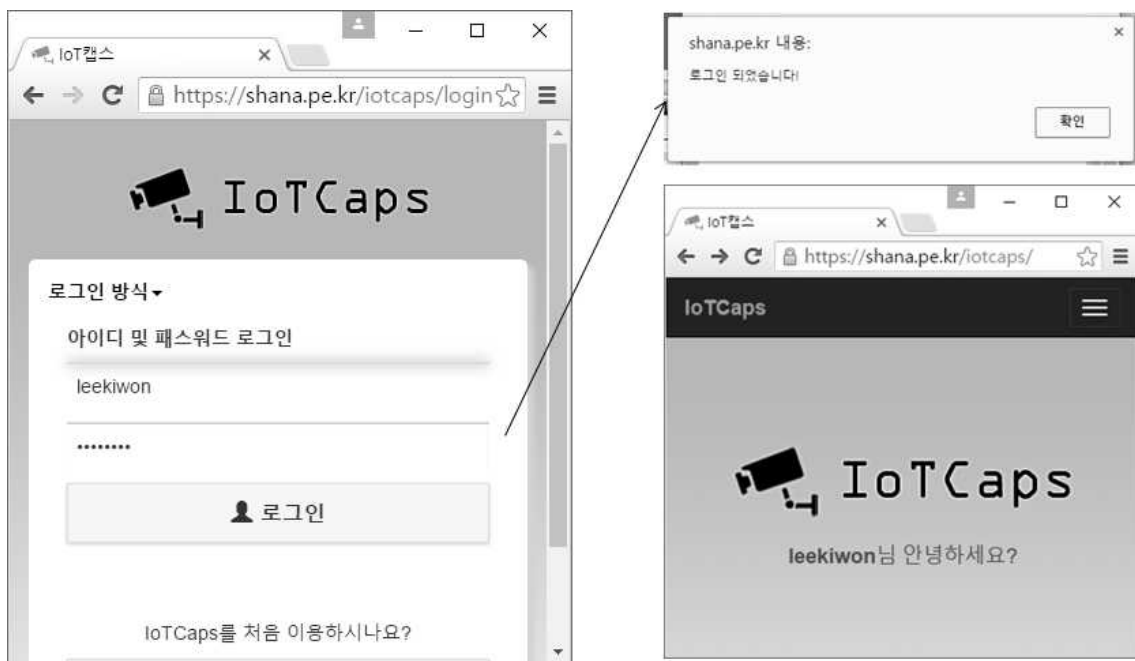


아래는 인증서를 통한 로그인 방법이다. 인증서 로그인 버튼을 누르면 아래와 같이 인증서 선택창이 나오게 된다. 인증서 설치 시 발급받은 개인키가 들어있는 인증서(pfx)파일을 실행하면 된다.





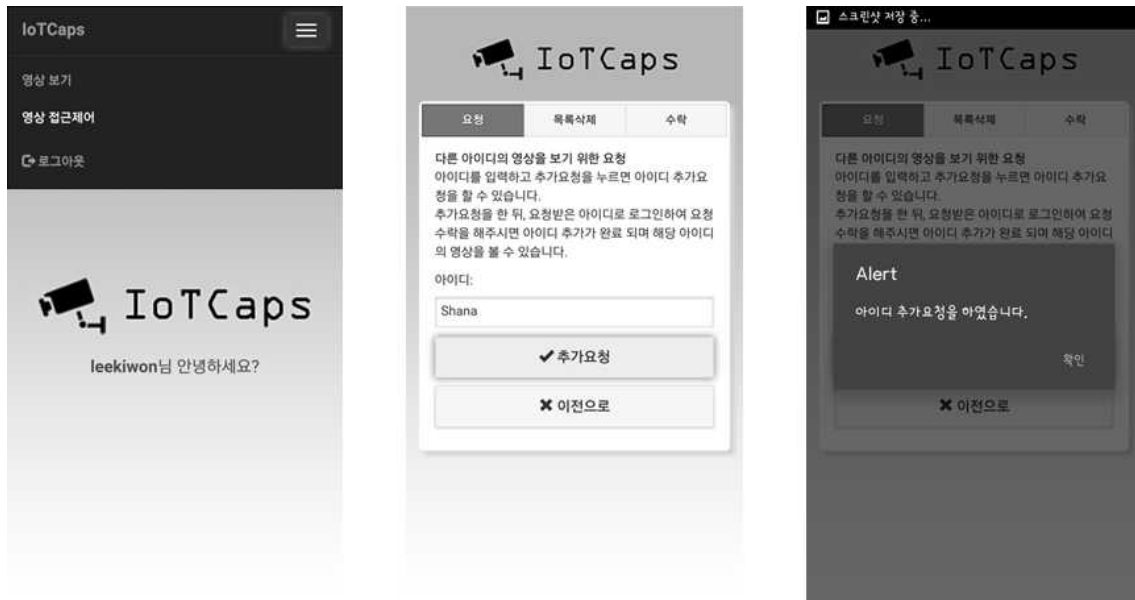
아래는 아이디 패스워드를 이용한 로그인 방법이다.



3-4 영상 접근 요청

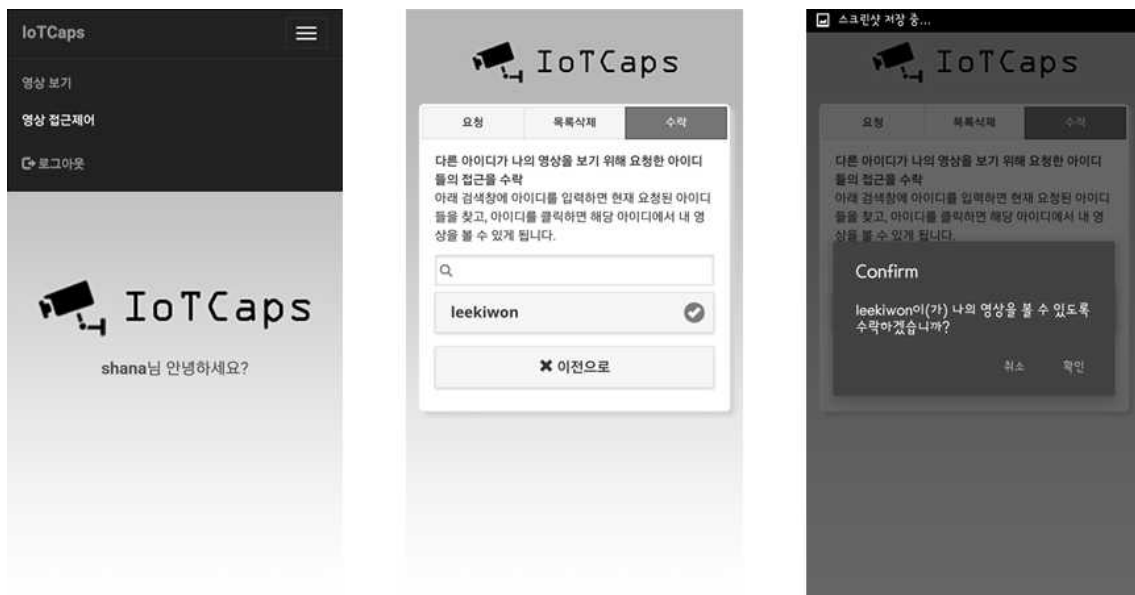
A라는 사용자가 B라는 사용자의 영상을 보려면 A사용자가 B사용자에게 요청을 하고 B가 수락해야 A가 B사용자의 영상을 볼 수 있다.

아래와 같이 영상 접근제어 -> 요청 -> 아이디 입력 후 추가요청을 해야 한다.



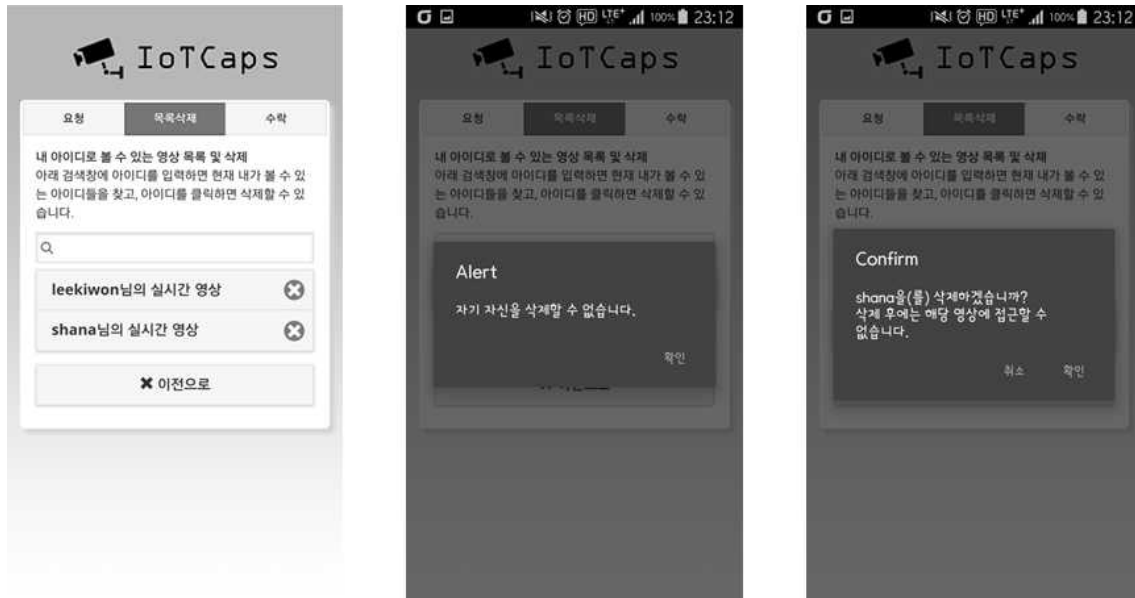
3-5 영상 접근 수락

영상 접근 수락은 접근을 요청 받은 사용자의 아이디로 로그인하여 요청한 사용자를 수락해주어야 한다. 영상 접근제어 -> 수락 -> 요청한 사용자 아이디를 클릭하여 수락할 수 있다.



3-6 목록삭제

아래와 같이 추가된 영상 목록을 삭제할 수 있다.



3-7 영상보기(잘못된 암호키)



- 영상을 선택하고 암호키 입력창에 잘못된 암호키 입력
- 채널에 접근 권한이 있는지 확인(인증 요청)
- 잘못된 암호키를 입력하면 바로 영상이 나오지 않고 10초 타임아웃 시작
- 10초 후에도 올바른 영상 데이터가 아니면 암호키 입력창 다시 출력

3-8 영상보기(올바른 암호키)



server영상 선택 후 올바른 암호키 입력

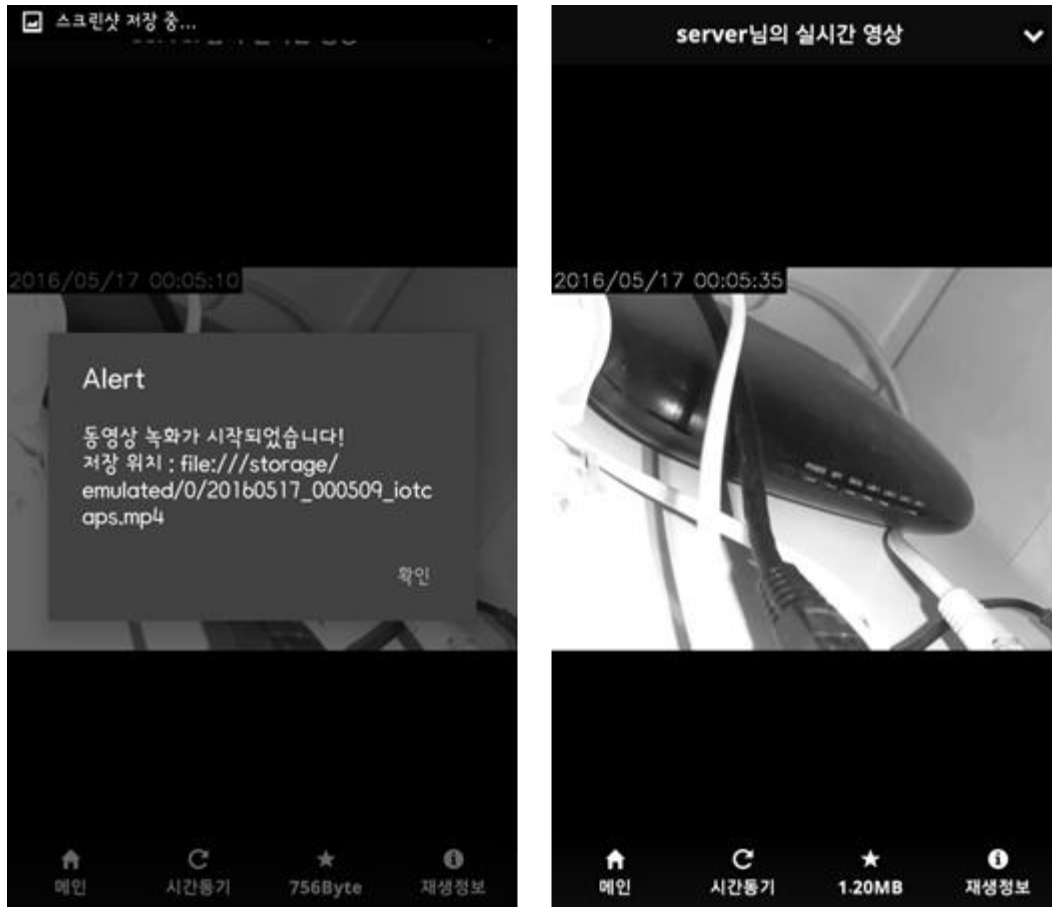
채널에 접근 권한이 있는지 확인

영상에 접속하는 중

영상 출력

3-9 시간동기/녹화

시간동기 기능은 영상의 지연이 발생할 경우 지연을 최소화 해주는 기능이다.
녹화 기능은 현재 영상을 녹화해주는 기능이다.



3-10 재생정보

재생정보 기능을 이용하여 현재 재생중인 영상의 재생정보를 볼 수 있다.



3-11 부터는 영상 전송 프로그램인 IOTCAM을 소개한다.



윈도우용 IOTCAM(x86)



라즈베리파이
리눅스용
IOTCAM(ARM)

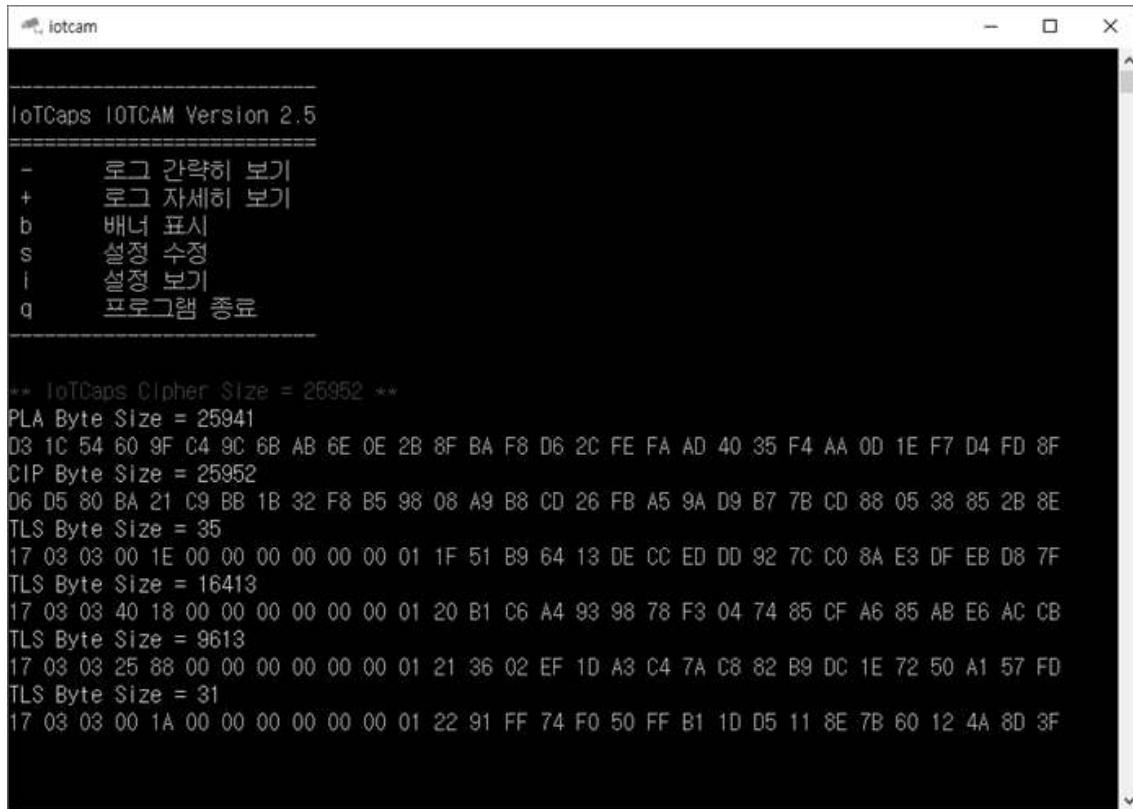
3-11 리눅스에서 영상 전송

```
IoTCaps IOTCAM Version 2.5
=====
-      로그 간략히 보기
+      로그 자세히 보기
b      배너 표시
s      설정 수정
i      설정 보기
q      프로그램 종료
=====

** IoTCaps Cipher Size = 32784 **
PLA Byte Size = 32768
00 00 00 34 73 69 64 78 01 00 00 00 00 00 01 00 0F 42 40 00 00 00 00 01 58 EA 7D 00 00
CIP Byte Size = 32784
AF F5 51 45 27 B7 2E 97 27 E6 2D 29 3F 51 9B 3A 2C 10 48 80 10 4A 88 82 EF 31 14 E9 33 3B
TLS Byte Size = 69
17 03 03 00 40 16 6B 2C 5D D8 C0 3E EA 8B 9B 42 D4 1F 2E 7A C0 03 E7 15 B4 8C 56 65 12 7A
TLS Byte Size = 16485
17 03 03 40 60 AA 51 9B 95 3A FF E2 FF 94 D7 60 28 21 BE F6 85 13 87 60 0A 4B D1 B1 80 24
TLS Byte Size = 16565
17 03 03 40 B0 20 AE 9C 39 33 B7 BC 8A 06 8E D8 77 40 BA 0C E9 E9 BD 37 05 31 40 4A EF 15
TLS Byte Size = 117
17 03 03 00 70 6D 8F FD BB 68 71 7F E0 24 5B A4 95 61 63 81 56 17 11 1B E6 DF A2 52 D4 75
TLS Byte Size = 149
17 03 03 00 90 FB 43 42 18 D6 49 91 9A 07 48 B8 21 43 CD AC 2F 3E 98 13 96 2D 7E 26 E7 C2
```

- 위의 스크린 샷은 영상 전송을 하는 장면을 자세하게 보여주고 있다.
- 리눅스에서는 v4l2를 이용하여 비디오 장치와 연결하여 영상을 전송한다.
- PLA는 평문, CIP는 AES-CBC로 암호화된 암호문, TLS는 실제 사용자가 받을 데이터이다.

3-12 윈도우에서 영상 전송

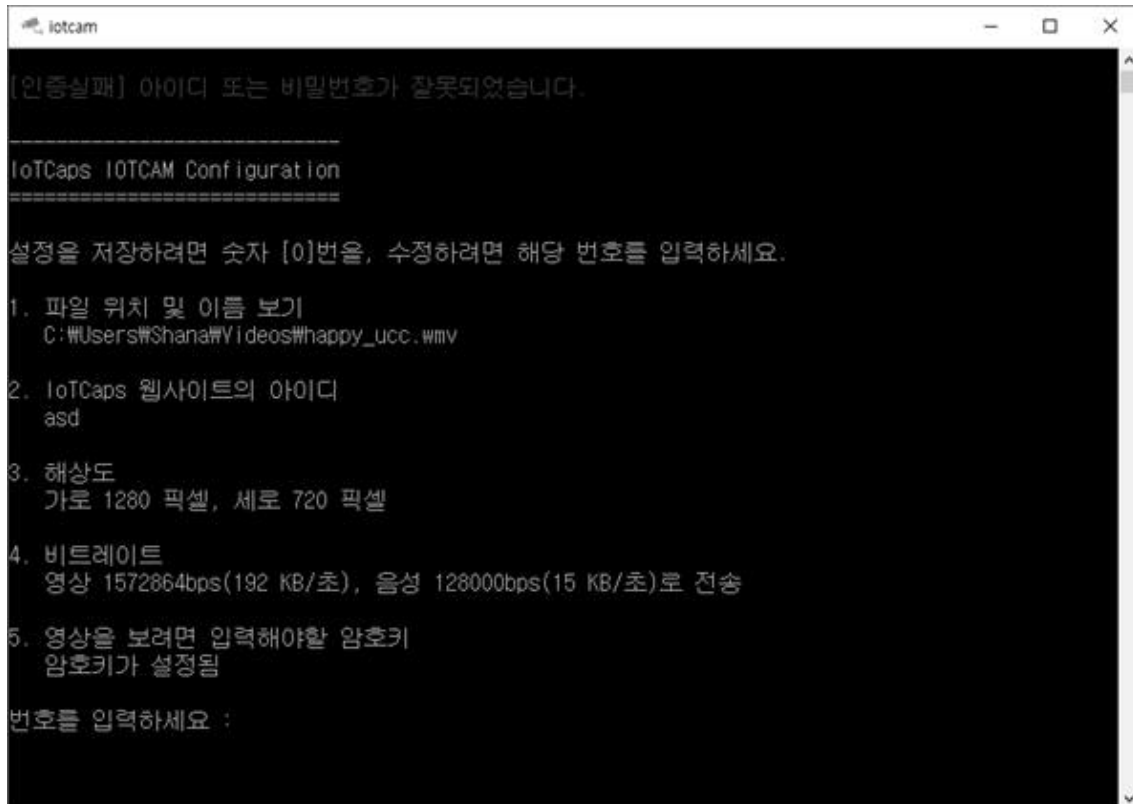


```
IoTcaps IoTcam Version 2.5
=====
-      로그 간략히 보기
+      로그 자세히 보기
b      배너 표시
s      설정 수정
i      설정 보기
q      프로그램 종료
=====

** IoTcaps Cipher Size = 25952 **
PLA Byte Size = 25941
03 1C 54 60 9F C4 9C 6B AB 6E 0E 2B 8F BA F8 D6 2C FE FA AD 40 35 F4 AA 0D 1E F7 D4 FD 8F
CIP Byte Size = 25952
06 D5 80 BA 21 C9 BB 1B 32 F8 B5 98 08 A9 B8 CD 26 FB A5 9A D9 B7 7B CD 88 05 38 85 2B 8E
TLS Byte Size = 35
17 03 03 00 1E 00 00 00 00 00 01 1F 51 B9 64 13 DE CC ED DD 92 7C C0 8A E3 DF EB D8 7F
TLS Byte Size = 16413
17 03 03 40 18 00 00 00 00 00 01 20 B1 C6 A4 93 98 78 F3 04 74 85 CF A6 85 AB E6 AC CB
TLS Byte Size = 9613
17 03 03 25 88 00 00 00 00 00 01 21 36 02 EF 1D A3 C4 7A C8 82 B9 DC 1E 72 50 A1 57 FD
TLS Byte Size = 31
17 03 03 00 1A 00 00 00 00 00 01 22 91 FF 74 F0 50 FF B1 1D D5 11 8E 7B 60 12 4A 8D 3F
```

- 윈도우 IOTCAM은 리눅스 IOTCAM과 같은 구조를 지니고 있다.
- 다만 장치들은 DirectShow를 사용하여 영상 장치 및 오디오 장치에 접근한다.
- TLS전송은 데이터의 크기가 2^{14} (16384) 바이트를 초과할 수 없기 때문에 TLS전송 시 쪼개서 전송한다.
- 그러므로 IOTCAM은 암호문 앞에 암호문의 크기를 같이 전송한다. 쪼개진 데이터를 다시 조립하려면 암호문의 크기를 알아야하기 때문이다.

3-13 IOTCAM에서 잘못된 아이디와 패스워드 입력 시



위와 같이 잘못된 아이디와 패스워드를 입력하여 인증 요청을 하면 상단에 인증실패 메시지가 나온다.

4-1. 결론 및 기대효과

스트리밍 데이터를 4단계로 보호

- 1단계: 다른 곳에서 동일한 아이디로 로그인시 이전에 접속했던 아이디는 자동 접속 해제 및 비인가자 접근 시 차단(인증 및 세션 관리)
- 2단계: 아무나 영상을 볼 수 없게 접근 제어(접근 제어)
- 3단계: 암호화 알고리즘인 AES-CBC를 사용하여 실시간 스트리밍 데이터 암호화(암호화)
- 4단계: TLS 전송(암호화)

기대 효과

사물인터넷(IoT)의 주요 서비스인 IP CCTV 해킹위협은 데이터 유출과 위조 및 변조, 사생활 침해 등이 있으며 이를 방지하기 위해 스트리밍 데이터 암호화가 필요하다.

차세대 IT기술인 인공지능(AI)과 IP CCTV기술이 접목된 지능형 영상감시 시스템의 관심이 높아지면서 영상 기술의 보안이 중요해지고 있으며 이를 위해 스트리밍 데이터의 암호화가 필요하다.

※ 지능형 영상감시 시스템은 영상을 인공지능으로 판단하여 물체 인식, 상황 인식, 교통관제 등을 할 수 있다.

5-1. 참고자료

Media Source Extensions(MSE)

<https://www.w3.org/TR/media-source/>

WebSocket

<http://d2.naver.com/helloworld/1336>

AES-CBC

<http://linuxforge.tistory.com/191>

6-1. 프로그램 소스

IoTCaps

config.html
<pre><div data-role="page" class="ui-page-theme-a bgimg" id="config"> <div data-role="content"> <div id="banner">
 <center></center>
 </div> <div data-role="tabs" id="tabs" style="background-color:white; box-shadow: 5px 5px 5px 0px lightgray; border-radius: 5px;"> <div data-role="navbar"> 요청 목록삭제 수락 </div> <div id="configtabone" class="ui-body-d ui-content"> 다른 아이디의 영상을 보기 위한 요청
 <p> 아이디를 입력하고 추가요청을 누르면 아이디 추가요청을 할 수 있습니다.
 추가요청을 한 뒤, 요청받은 아이디로 로그인 하여 요청수락을 해주시면 아이디 추가가 완료 되며 해당 아이디의 영상을 볼 수 있습니 다. </p> <p>아이디: <input type="text" value="" maxlength="20" id="configaddid" /></p> <button id="configaddbtn" onclick="chAddReq();" >&nbsp;추가요청</button></pre>

<pre> <button id="configprevbtn" onclick="history.back();" >&nbsp;이전으로</button> </div> <div id="configtabtwo" class="ui-body-d ui-content" style="display:none;"> 내 아이디로 볼 수 있는 영상 목록 및 삭제
 <p>아래 검색창에 아이디를 입력하면 현재 내가 볼 수 있는 아이디들을 찾고, 아이디를 클릭하면 삭제할 수 있습니다.</p> <form class="ui-filterable"><input id="idFilter" data-type="search"></form> <ul id="config-id" data-role="listview" data-filter="true" data-inset="true" data-input="#idFilter"> <button id="configprevbtn" onclick="history.back();" >&nbsp;이전으로</button> </div> <div id="configtabthree" class="ui-body-d ui-content" style="display:none;"> 다른 아이디가 나의 영상을 보기 위해 요청한 아이 디들의 접근을 수락
 <p>아래 검색창에 아이디를 입력하면 현재 요청된 아 이디들을 찾고, 아이디를 클릭하면 해당 아이디에서 내 영상을 볼 수 있게 됩니다.</p> <form class="ui-filterable"><input id="idREQFilter" data-type="search"></form> <ul id="config-req-id" data-role="listview" data-filter="true" data-inset="true" data-input="#idREQFilter"> <button id="configprevbtn" onclick="history.back();" >&nbsp;이전으로</button> </div> </div> </div> </div> </pre>
<pre> index.html <!DOCTYPE html> <html lang="ko"> <head> </pre>

```

<meta charset="utf-8" name="viewport" content="width=device-width" />
<title>IoT캡스</title>
<script type="text/javascript" src="js/jquery-1.11.1.min.js"></script>
<script type="text/javascript" src="js/jquery.mobile-1.4.5.min.js"></script>
<script type="text/javascript" src="js/socket.io.min.js"></script>
<script type="text/javascript" src="js/forge.min.0.6.12.js"></script>
<script type="text/javascript" src="js/iotapp.js"></script>
<script type="text/javascript" src="js/iotsession.js"></script>
<script type="text/javascript" src="js/iotcert.js"></script>
<script type="text/javascript" src="js/iotlive.js"></script>
<script type="text/javascript" src="js/bootstrap.min.js"></script>
<link rel="stylesheet" type="text/css" href="css/iotcaps.min.css" />
<link
            rel="stylesheet"
            type="text/css"
href="css/jquery.mobile.icons.min.css" />
<link
            rel="stylesheet"
            type="text/css"
href="css/jquery.mobile.structure-1.4.5.min.css">
<link rel="stylesheet" type="text/css" href="css/bootstrap.min.css" />
<link rel="stylesheet" type="text/css" href="css/bootstrap-glyphicons.css"
/>
<link rel="stylesheet" type="text/css" href="css/iotcaps-new.css" />
<link
            rel="shortcut
            icon"
            type="image/x-icon"
href="css/images/favicon.ico" />
</head>
<body>
<div data-role="page" class="ui-page-theme-b bgimg" id="index">

    <nav class="navbar navbar-inverse">
        <div class="container-fluid">
            <div class="navbar-header">
                <a class="navbar-brand" href="#">IoTCaps</a>
                <button
                    style="width:45px"
                    type="button"
class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar"
aria-expanded="false" aria-controls="navbar">
                    <span class="sr-only">토글 메뉴바</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
            </div>
            <div id="navbar" class="navbar-collapse collapse">

```

```

        <ul id="menulayout" class="nav
navbar-nav"></ul>

        <ul id="memberlayout" class="nav navbar-nav
navbar-right"></ul>

    </div>

</div>

</nav>

<div data-role="content">
    <div id="banner">
        <br /><br /><br />
        <center></center>
        <br />
    </div>
    <center><p style="color:#4c4c4c; font-size:18px; text-shadow: 0 0
0 darkgray;" id="iot_logininfo"></p></center>
</div>

</div>
</body>
</html>

```

join.html

```

<div data-role="page" class="ui-page-theme-a bgimg" id="join">
    <div data-role="content">
        <div id="banner">
            <br />
            <center></center>
            <br />
        </div>

        <div class="ui-body-d ui-content" style="background-color:white;
box-shadow: 5px 5px 5px 0px lightgray; border-radius: 5px;">

            <b>회원가입</b><br />

            <form class="form-inline" role="form">
                <div class="form-group" style="width:100%">
                    <label class="sr-only" for="InputID">아이디

```



```

        <strong><h5 style="color: white;text-shadow: 1px 1px 1px
#000000;">
            <p id="debugbox_msg"></p>
            <p id="debugbox_msg_cip"></p>
            <p id="debugbox_msg_dat"></p>
            <p id="debugbox_msg_rec"></p>
        </h4></strong>
    </div>

    <div data-role="live-header" id="headerPanel" style="position:absolute;
width:100%; margin-top:-7px; z-index:1;">
        <select id="iotlive-id" data-native-menu="false"
data-placeholder="true" onChange="iotlive_selectCH(this)"></select>
    </div>

    <video id="vtag" style="width:100%; display:block;" autoplay></video>

    <div data-role="navbar" id="bottomPanel" style="position:absolute;
width:100%; bottom:0px; left:0px; z-index:1;">
        <ul>
            <li><button style="border:0; background:linear-gradient(to
bottom, rgba(255,255,255,0), rgba(34,34,34,0.4));" class="ui-btn ui-icon-home
ui-btn-icon-top" onclick="history.back();">메인</button></li>
            <li><button style="border:0; background:linear-gradient(to
bottom, rgba(255,255,255,0), rgba(34,34,34,0.4));" class="ui-btn ui-icon-refresh
ui-btn-icon-top" onclick="iotlive_resync();">시간동기</button></li>
            <li><button style="border:0; background:linear-gradient(to
bottom, rgba(255,255,255,0), rgba(34,34,34,0.4));" class="ui-btn ui-icon-star
ui-btn-icon-top" id="recbtn" onclick="iotlive_rec(0);">녹화시작</button>
                <a style="border:0; background:linear-gradient(to
bottom, rgba(255,255,255,0), rgba(34,34,34,0.4));" class="ui-btn ui-icon-star
ui-btn-icon-top" id="downloadLink" onclick="iotlive_save(0);" style="display:none">
저장하기</a></li>
            <li><button style="border:0; background:linear-gradient(to
bottom, rgba(255,255,255,0), rgba(34,34,34,0.4));" class="ui-btn ui-icon-info
ui-btn-icon-top" onclick="iotlive_info();">재생정보</button></li>
        </ul>
    </div>
</div>

```

login.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="utf-8" name="viewport" content="width=device-width" />
  <title>IoT캡스</title>
  <script type="text/javascript" src="js/jquery-1.11.1.min.js"></script>
  <script type="text/javascript" src="js/jquery.mobile-1.4.5.min.js"></script>
  <script type="text/javascript" src="js/socket.io.min.js"></script>
  <script type="text/javascript" src="js/forge.min.0.6.12.js"></script>
  <script type="text/javascript" src="js/iotapp.js"></script>
  <script type="text/javascript" src="js/iotsession.js"></script>
  <script type="text/javascript" src="js/iotcert.js"></script>
  <script type="text/javascript" src="js/iotlive.js"></script>
  <script type="text/javascript" src="js/bootstrap.min.js"></script>
  <link rel="stylesheet" type="text/css" href="css/iotcaps.min.css" />
  <link
    rel="stylesheet"
    type="text/css"
href="css/jquery.mobile.icons.min.css" />
  <link
    rel="stylesheet"
    type="text/css"
href="css/jquery.mobile.structure-1.4.5.min.css">
  <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css" />
  <link rel="stylesheet" type="text/css" href="css/bootstrap-glyphicons.css"
/>
  <link rel="stylesheet" type="text/css" href="css/iotcaps-new.css" />
  <link
    rel="shortcut
    icon"
    type="image/x-icon"
href="css/images/favicon.ico" />
</head>
<body>
<div data-role="page" class="ui-page-theme-a bgimg" id="login">
  <div data-role="content">
    <div id="banner">
      <br />
      <center></center>
      <br />
    </div>
    <div class="ui-body-d ui-content" style="background-color:white;
box-shadow: 5px 5px 5px 0px lightgray; border-radius: 5px;">

      <div class="dropdown nav">
        <a
          id="dLabel"
          data-toggle="dropdown"
```

```

aria-haspopup="true"    role="button"    aria-expanded="false">로그인    방식<span
class="caret"></span></a>

        <ul        class="dropdown-menu"        role="menu"
aria-labelledby="dLabel">

                <li><a                                href="#"
onclick="$('#logintabone').show();$('#logintabtwo').hide();$('#logintabthree').hide()">I
D/PW</a></li>

                <li><a                                href="#"
onclick="$('#logintabtwo').show();$('#logintabone').hide();$('#logintabthree').hide()">
인증키</a></li>

                <li><a                                href="#"
onclick="$('#logintabthree').show();$('#logintabone').hide();$('#logintabtwo').hide()">
인증서</a></li>

        </ul>

</div>

        <div id="logintabone" class="ui-body-d ui-content">
                <b>아이디 및 패스워드 로그인</b>

                <form class="form-inline" role="form">
                        <div                                class="form-group"
style="width:100%">
                                <label                                class="sr-only"
for="InputID">아이디</label>
                                <input type="text" value=""
maxlength="20"    id="loginid"    class="form-control    col-xs-4"    placeholder="ID"
style="width:100%">
                                </div>
                                <br>
                                <div                                class="form-group"
style="width:100%">
                                        <label                                class="sr-only"
for="InputPassword">Password</label>
                                        <input    type="password"
value=""    maxlength="20"    id="loginpassword"    class="form-control    col-xs-4"
placeholder="Password" style="width:100%">
                                        </div>
                                </form>
                                <br />

```

```

                                <button                                id="idpwlgbtn"
onclick="login();"><span    class='glyphicon    glyphicon-user'></span>&nbsp;로그인
</button>

                                <br />
                                </div>

                                <div    id="logintabtwo"    class="ui-body-d    ui-content"
style="display:none;">

                                <b>인증키(인증서 및 개인키) 로그인</b>
                                <p>파일    열기    :    <input    type="file"
id="inputfile" /></p>

                                <p    id="filestatus">인증서(*.cer)와    개인키
(*.pem)를 불러와주세요.<br /></p>

                                <button                                id="certlgbtn"
onclick="verifyCert();"><span    class='glyphicon    glyphicon-user'></span>&nbsp;인증
키 로그인</button>

                                <p    id="verifystatus"></p>

                                </div>
                                <div    id="logintabthree"    class="ui-body-d    ui-content"
style="display:none;">

                                <b>인증서 로그인</b>
                                <p>가입시 발급받은 pfx파일을 통해 운영체제에 저장
한 인증서로 로그인합니다.<br />

                                다만, 앱을 통해서도 해당 로그인을 할 수 있으며 브라
우저를 통한 접속만 인증이 가능합니다.</p>

                                <    b    u    t    t    o    n
onclick="location.href='https://shana.pe.kr/iotcaps/auth/';"><span
class='glyphicon    glyphicon-user'></span>&nbsp;인증서 로그인</button>

                                </div>

                                <div id="logintabone" class="ui-body-d ui-content">
                                <center>IoTcaps를 처음 이용하시나요?</center>
                                <a            href="join.html"            data-role="button"
class="btn-lg">회원 가입</a>
                                </div>

                                </div>

                                </div>
</div>
</div>

```

```
</body>
```

```
</html>
```

logout.html

```
<div data-role="page" class="ui-page-theme-a bgimg" id="logout">
```

```
</div>
```

iotcaps-new.css

```
.bgimg {  
    background-image:url("images/bg.png");  
    background-repeat: repeat-x;  
    background-color: white;  
}  
.ui-page-theme-a a {  
    color: black;  
}
```

iotapp.js

```
/**  
 * Created by LEEKIWON on 2015-11-15.  
 * Last modification date 2016-05-15.  
 */  
var runiotapp = false; // 앱으로 실행하면 true가 된다.  
function dynamicLoadJSCSS(filename, filetype)  
{  
    if (filetype == "js")  
    {  
        var fileref=document.createElement('script');  
        fileref.setAttribute("type","text/javascript");  
        fileref.setAttribute("src", filename);  
    }  
    else if (filetype == "css")  
    {  
        var fileref=document.createElement("link");  
        fileref.setAttribute("rel", "stylesheet");  
        fileref.setAttribute("type", "text/css");  
        fileref.setAttribute("href", filename);  
    }  
    if (typeof fileref !== undefined)  
        document.getElementsByTagName("head")[0].appendChild(fileref);  
}  
function init()  
{  
    runiotapp = true;  
}  
if (navigator.userAgent.match(/(iPhone|iPod|iPad|Android|BlackBerry)/))  
{  
    // cordova 자바스크립트 불러오기
```

```

dynamicLoadJSCSS("cordova.js", "js");
    // 장치 레디
document.addEventListener("deviceready", init, false);
}else
{
    var agt = navigator.userAgent.toLowerCase();
    if(agt.indexOf("chrome") == -1)
    {
        alert('본 브라우저로는 접속할 수 없습니다.\nPC에서는 크롬으로, 모바일에서는 IoT캡스 앱으로 접속하시길 바랍니다.');
```

일에서는 IoT캡스 앱으로 접속하시길 바랍니다.');

```

        window.open("about:blank", "_self").close();
    }
}function fail(error)
{
    var msg = error;
    switch(error.code)
    {
        case FileError.NOT_FOUND_ERR:
            msg = "File Not Found";
            break;
        case FileError.SECURITY_ERR:
            msg = "Security Error";
            break;
        case FileError.ABORT_ERR:
            msg = "Abort error";
            break;
        case FileError.NOT_READABLE_ERR:
            msg = "Not Readable";
            break;
        case FileError.ENCODING_ERR:
            msg = "Encoding Error";
            break;
        case FileError.NO_MODIFICATION_ALLOWED_ERR:
            msg = "No Modification Allowed";
            break;
        case FileError.INVALID_STATE_ERR:
            msg = "Invalid State";
            break;
        case FileError.SYNTAX_ERR:
            msg = "Syntax Error";
            break;
        case FileError.INVALID_MODIFICATION_ERR:

```

```

        msg = "Invalid Modification Error";
        break;
    case FileError.QUOTA_EXCEEDED_ERR:
        msg = "Quota Exceeded";
        break;
    case FileError.TYPE_MISMATCH_ERR:
        msg = "Type Mismatch Error";
        break;
    case FileError.PATH_EXISTS_ERR:
        msg = "Path Already Exists Error";
        break;
    }
    alert("fail: "+ msg);
}
// type 0 = 동영상 녹화 저장용(헤더 저장, 메시지 출력)
// type 1 = 동영상 녹화 저장용(헤더 저장, 메시지 노출력)
// type 2 = 동영상 녹화 이어 붙이기 용(이어 저장)
// type 9 = 인증서, 개인키 저장용(decodeURI 작업 함)
function createFile(filename, data, type)
{
    try
    {
        if (type=='9') data = decodeURIComponent(data); // 인증서 저장용

window.resolveLocalFileSystemURL(cordova.file.externalRootDirectory,
function(dir)
    {
        dir.getFile(filename, {create:true}, function(file)
        {
            file.createWriter(function(writer)
            {
                writer.onwriteend = function(evt)
                {
                    if (type=='9')
                        alert("파일이 저장되었습니다!\n저장 위치 : " + cordova.file.externalRootDirectory + filename);
                    else if (type=='0' || type=='1')
                    {
                        if (type=='0') alert("동영상 녹화가 시작되었습니다!\n저장 위치 : " + cordova.file.externalRootDirectory + filename);

recnow = false; // 작업

```

완료	getHEADrec = true;
	}
	else if (type=='2')
	{
	recnow = false; // 작업
완료	if (recsize >= (recSizeApp
* 1024 * 1024)) // 분할 저장 부분	{
	iotlive_save(1); //
저장	iotlive_rec(1); //
다시 기록 시작	}
	}
	};
	if (type=='2') writer.seek(writer.length); //
이어 붙이기	writer.write(data); // 파일 기록 작업
	}, fail);
	});
	});
	catch (e)
	{
	alert(e.message);
	}
	}

iotcert.js
<pre> /** * Created by LEEKIWON on 2015-10-26. * Last modification date 2016-05-16. */ function login() { // 비활성화 \$('#idpwlgbtn').addClass('ui-disabled'); var md = forge.md.sha256.create(); md.update(\$('#loginpassword')[0].value); // 정보 보내기 </pre>

```

        sessionSocket.emit('LOGIN_REQ',      [$('#loginid')[0].value.toLowerCase(),
md.digest().toHex()));
        sessionSocket.on('LOGIN_OKAY', function(data)
        {
            sessionSocket.removeAllListeners();
            sessionStorage.setItem('id', data[0]);
            sessionStorage.setItem('uid', data[1]);
            alert('로그인 되었습니다!');
            // 활성화
            $('#idpwlgbtn').removeClass('ui-disabled');
            history.back();
        });
        sessionSocket.on('LOGIN_FAIL', function(data)
        {
            sessionSocket.removeAllListeners();
            alert('아이디 또는 패스워드가 잘못되었습니다!');
            // 활성화
            $('#idpwlgbtn').removeClass('ui-disabled');
        });
    }var logincertpem, loginprikeypem;
    function readFile(e)
    {
        function infolabel(str, append)
        {
            if (append) $('#filestatus')[0].innerHTML += str;
            else $('#filestatus')[0].innerHTML = str;
        }
        var file = e.target.files[0];
        if (!file) return;
        var reader = new FileReader();
        reader.onload = function(e)
        {
            var contents = e.target.result;
            // 파일 첨부한 내용에따라 개인키 인증서 구분
            if(contents.indexOf('-----BEGIN CERTIFICATE-----') != -1) // 인
증서
            {
                logincertpem = contents;
                infolabel('인증서 파일' + file.name + ' 불러왔습니다.<br />',
1);
            }
        }
    }

```

```

else if(contents.indexOf('-----BEGIN RSA PRIVATE KEY-----') !=
-1) // 개인키
{
    loginprikeypem = contents;
    infolabel('개인키 파일' + file.name + ' 불러왔습니다.<br />',
1);
}
else
    alert('올바른 인증서 또는 개인키 파일이 아닙니다!');
};
reader.readAsText(file);
}function verifyCert()
{
    // 비활성화
    $('#certlgncbtn').addClass('ui-disabled');
    function infolabel(str)
    {
        $('#verifystatus')[0].innerHTML = str;
    }
    if (loginprikeypem === undefined || logincertpem === undefined)
    {
        alert('인증서와 개인키 파일을 불러와주세요.');
```

인키

```

        // 활성화
        $('#certlgncbtn').removeClass('ui-disabled');
        return;
    }
    var prikey = forge.pki.privateKeyFromPem(loginprikeypem); // 개
    var cert = forge.pki.certificateFromPem(logincertpem); // 인증서
    var dt = new Date();
    // 아이디 값을 서명
    var msg = cert.subject.attributes[3].value + ' ' + dt;
    var md = forge.md.sha1.create();
    md.update(msg, 'utf8');
    var sign = prikey.sign(md);
    // 서명 확인
    try
    {
        var check = cert.publicKey.verify(md.digest().bytes(), sign);

        // 정보 보내기

```

```

        sessionSocket.emit('VERIFY_CERT_REQ',
[forge.util.bytesToHex(sign), logincertpem, msg]);

        sessionSocket.on('LOGIN_OKAY', function(data)
        {
            sessionSocket.removeAllListeners();
            sessionStorage.setItem('id', data[0]);
            sessionStorage.setItem('uid', data[1]);
            alert('로그인 되었습니다!');
            // 활성화
            $('#certlgnbtn').removeClass('ui-disabled');
            history.back();
        });
        sessionSocket.on('LOGIN_FAIL', function(data)
        {
            sessionSocket.removeAllListeners();
            alert('등록되어있지 않은 인증서입니다!');
            // 활성화
            $('#certlgnbtn').removeClass('ui-disabled');
        });
    }
    catch(e)
    {
        alert('인증서가 잘못되었습니다!');
        // 활성화
        $('#certlgnbtn').removeClass('ui-disabled');
        return;
    }
}function createCert()
{
    function infolabel(str, append)
    {
        {
            if (append) $('#certstatus')[0].innerHTML += str;
            else $('#certstatus')[0].innerHTML = str;
        }

        var idvar = $('#certid')[0];
        var passwordvar = $('#certpassword')[0];
        var btnvar = $('#certbtn');
        var prevbtn = $('#certprevbtn');
        // 빈 아이디 검사
        if(idvar.value == "")

```

```

{
    alert("아이디를 입력하세요.");
    idvar.focus();
    return;
}

// 영문소문자, 숫자만 입력받기
for (i=0; i<idvar.value.length; i++)
{
    ch=idvar.value.charAt(i);
    if (!(ch >= '0' && ch <= '9') && !(ch >= 'a' && ch <= 'z'))
    {
        alert ("아이디에는 영문 소문자와 숫자만 입력할 수 있습니다.");

        idvar.focus();
        idvar.select();
        return
    }
}

// 아이디 공백 검사
if (idvar.value.indexOf(" ")>=0)
{
    alert("아이디에는 공백을 사용할 수 없습니다.");
    idvar.focus();
    idvar.select();
    return;
}

// 아이디 길이 (2~20)자 까지
if (idvar.value.length<2 || idvar.value.length>20)
{
    alert ("아이디를 2에서 20자 사이로 입력하세요.");
    idvar.focus();
    idvar.select();
    return;
}

// 빈 비밀번호 체크
if (passwordvar.value=="")
{
    alert("비밀번호를 입력하세요.");
    passwordvar.focus();
    return;
}

```

```

    }

    // 비밀번호 길이 (4~20)자 까지
    if (passwordvar.value.length<4 || passwordvar.value.length>20)
    {
        alert ("비밀번호를 4에서 20자 사이로 입력하세요");
        passwordvar.focus();
        passwordvar.select();
        return;
    }

    // 버튼 감추기
    btnvar.closest('.ui-btn').hide();
    prevbtn.closest('.ui-btn').hide();

    // 키생성
    var keypair = forge.pki.rsa.generateKeyPair(1024);
    var publicKey = keypair.publicKey;
    var privateKey = keypair.privateKey;
    // pem포맷으로 변환
    var pemPub = forge.pki.publicKeyToPem(publicKey);
    var pemPriv = forge.pki.privateKeyToPem(privateKey);
    // 정보 보내기
    sessionSocket.emit('CERT_REQ', [idvar.value, passwordvar.value, pemPub,
pemPriv]);
    infolabel('<b>공개키를 이용하여 인증서 발급요청중입니다.</b><br />', 0);
    // 인증서를 받았으면 표시
    sessionSocket.on('CERT_OK', function(data)
    {
        sessionSocket.removeAllListeners();
        var uripkcs12data = 'data:application/x-pkcs12;base64,' +
encodeURIComponent(data[1]); // 개인키 + 인증서(PKCS#12)
        var uricerdata = 'data:application/text;charset=utf-8,' +
encodeURIComponent(data[0]); // 인증서(CER)
        var uriprdata = 'data:application/text;charset=utf-8,' +
encodeURIComponent(pemPriv); // 개인키(PEM)
        infolabel('<b>인증서를 발급받았습니다.</b><br /><br />', 0);

        //*****
        infolabel('인증키(인증서 및 개인키)로 로그인하려면 아래 2개 파일을 다
운로드 받아 저장하세요.<br />', 1);
    });

```

```

        // 앱에서 실행시
        if (runiotapp)
        {
            infolabel('<a onclick=\"createFile(\"' + idvar.value +
'.cer\\', \"' + encodeURIComponent(data[0]) + '\\', \'9\');\" href=#>인증서 다운로드
</a>', 1);

            infolabel('<a onclick=\"createFile(\"' + idvar.value +
'.pem\\', \"' + encodeURIComponent(pemPriv) + '\\', \'9\');\" href=#>개인키 다운로
드</a>', 1);

        }
        else // 웹에서 실행시
        {
            infolabel('<a href=' + uricerdata + ' download=\"' +
idvar.value + '.cer\">인증서 다운로드</a>', 1);
            infolabel('<a href=' + uripridata + ' download=\"' +
idvar.value + '.pem\">개인키 다운로드</a>', 1);
        }
        //*****

        infolabel('<br />', 1);

        //*****
        infolabel('인증서로 로그인하려면 아래 1개 파일을 다운로드 받아 실행
하세요.<br />', 1);
        infolabel('운영체제에 저장할 때 비밀번호를 물어보는데, 비밀번호는 가
입시 입력한 비밀번호 입니다.<br />', 1);

        // 앱에서 실행시
        if (runiotapp)
            infolabel('<a onclick=\"createFile(\"' + idvar.value +
'.pfx\\', \"' + encodeURIComponent(data[1]) + '\\', \'9\');\" href=#>인증서 다운로드
</a>', 1);

        else // 웹에서 실행시
            infolabel('<a href=' + uripkcs12data + ' download=\"' +
idvar.value + '.pfx\">인증서 다운로드</a>', 1);
        //*****

```

```

        $('#certstatus').children('a').buttonMarkup();
        prevbtn.closest('.ui-btn').show();
    });
    // 인증서를 못 받았으면 표시
    sessionSocket.on('CERT_FAIL', function(data)
    {
        sessionSocket.removeAllListeners();
        infolabel('<b>' + data + '</b></br>', 0);
        btnvar.closest('.ui-btn').show();
        prevbtn.closest('.ui-btn').show();
    });
}

```

iotlive.js

```

/**
 * Created by LEEKIWON on 2015-09-15.
 * Last modification date 2016-05-15.
 */
//////////
// 초기화
var socketok = undefined; // 웹소켓 데이터
var mediaSource = undefined; // 미디어 소스 API
var sourceBuffer = undefined; // 미디어 소스 API
var videoBufferHead = []; // 데이터 헤더
var videoBuffer = []; // 데이터
var getSIDXrec = false; // 처음 기록이 SIDX으로 했는지 확인
var appendBuf = false; // 소스버퍼에 어팬드 여부
var getHEADrec = false; // 헤더 데이터가 들어갔는지 확인
var recnow = false; // 쓰기 작업여부
var recfilename = ""; // 기록할 때 파일 이름
var recsize = ""; // 기록 파일 크기 측정
var recbloburl = ""; // blob기록용 주소
var queue = []; // 큐 데이터
var streamQueue = []; // 영상데이터 큐
var reaminQueue = []; // 남은 큐
var streamDecProcess = false; // 임계구역 설정(병행수행시 충돌 방지)
var getHeaderData = false; // 헤더 데이터 삽입 여부

```

```

var getSIDXpla = false; // 처음 시작이 SIDX으로 했는지 확인
var firstsidx = false; // 시작 인덱스인지 확인(채널을 변경하고 시작할 때 설정)
var authloopcnt = 10; // 인증 루프 카운트(타임아웃 체크)
var datasize = 0; // 비트율 사이즈
var datapos = 0; // 비트율 계산용 시간
var datasizeprt = 0; // 디버그 프린트용 비트율 사이즈
//////////
// 기타
var syncTimer = null; // 싱크 맞출때 쓰는 변수
var authTimer = null; // 인증요청 타이머
var setTOffset = 86400;
var decipheriv = "";
//////////
// 환경검사
var isMSEdge = (navigator.appName == "Netscape") &&
(navigator.appVersion.indexOf('Edge') != -1);
//////////
// 환경설정
var streamURL = "";
var streamID = "";
var playOption = "";
var recSizeWeb = 230; // 웹에서 녹화할 때 분할 MB 단위 용량( 230MB 추천 249MB
최대)-라즈베리 카메라로 20분
if (isMSEdge) recSizeWeb = 10; // 엣지는 10MB 단위로 저장
var recSizeApp = 1024; // 앱에서 녹화할 때 분할 MB 단위 용량(1024MB 추천
1024MB 최대)-라즈베리 카메라로 1시간 30분
var selectCH = false; // 수동으로 채널을 선택한 여부
//////////
// 디버그
var iotliveDebug = false;
function iotlive_info()
{
    if (iotliveDebug) iotliveDebug = false;
    else iotliveDebug = true;
    iotlive_info_init();
}function iotlive_info_init()
{
    if (!iotliveDebug)
    {
        debugbox_msg.innerHTML = "";
        debugbox_msg.style.display = "none";
        debugbox_msg_dat.innerHTML = "";
    }
}

```

```

        debugbox_msg_dat.style.display = "none";
        debugbox_msg_cip.innerHTML = "";
        debugbox_msg_cip.style.display = "none";
        debugbox_msg_rec.innerHTML = "";
        debugbox_msg_rec.style.display = "none";
    }
    else
    {
        debugbox_msg.innerHTML = "IoT Caps Playback Information";
        debugbox_msg.style.display = "block";
        debugbox_msg_dat.innerHTML = "";
        debugbox_msg_dat.style.display = "block";
        debugbox_msg_cip.innerHTML = "";
        debugbox_msg_cip.style.display = "block";
        debugbox_msg_rec.innerHTML = "";
        debugbox_msg_rec.style.display = "block";
    }
}function iotlive_conf(addr, id)
{
    streamURL = addr;
    streamID = id;
}
// 시작 데이터 체크
function iotlive_chk4sidx(data)
{
    try
    {
        var dataView = new DataView(data);
        var dest_array = ['0', '0', '0', '34', '73', '69', '64', '78']; // 4 s i
        for (var i=0; i<8; i++)
        {
            if (dest_array[i] != dataView.getUint8(i).toString(16))
            {
                dataView = NULL;
                break;
            }
        }
        if (i==8) return 'eq';
        else return 'ne';
    }
    catch (e)
    {
        return 'ne';
    }
}

```

```

} // 싱크를 맞춰주는 함수
function iotlive_resync()
{
    try
    {
        if (!isMSEdge) // 엣지브라우저가 아니면 시간 맞춤 수행
        {
            $('#vtag')[0].pause();
            $('#vtag')[0].currentTime = setTOffset;
        }
        clearTimeout(syncTimer);
        syncTimer = setTimeout(function()
        {
            $('#vtag')[0].play();
            // 재생 성공시 메시지 제거
            if (getSIDXpla == true && $('#vtag')[0].currentTime > 0
            && $('#vtag')[0].currentTime != setTOffset)
            {
                alertbox_msg.innerHTML = "";
                alertbox_msg.style.display = "none";
            }

        }, 333);
    }
    catch (e) {}
} // 영상 기록
// 0 - 앱에서 처음 녹화(메시지 띄우기)
// 1 - 분할 저장시 이어 기록(메시지 안 띄움)
function iotlive_rec(msg)
{
    if (videoBufferHead.length != 2)
    {
        alert('헤더 데이터가 없어서 아직 녹화할 수 없습니다.');
```

return;

```

    }
    else
    {
        // 데이터 및 링크
        var a = $('#downloadLink')[0];
        recfilename = iotlive_filename(); // 기록 시작한 파일이름
(날짜 및 시간)

        // 앱에서 실행시(상시 저장)

```

```

if (runiotapp)
{
    var recBuffer = [];
    recBuffer.push(videoBufferHead[0]); // 헤더 넣기
    recBuffer.push(videoBufferHead[1]); // 헤더 넣기
    recsize = recBuffer[0].byteLength; // 파일 크기
    recsize += recBuffer[1].byteLength; // 파일 크기
    a.text = iotlive_calsize(recsize); // 저장하기 버튼에 파일 사
    이즈 표시

    recnow = true;
    var blob = new Blob(recBuffer, {type: 'video/mp4'});
    var fileReader = new FileReader();
    fileReader.readAsArrayBuffer(blob);
    fileReader.onload = function()
    {
        createFile(recfilename, this.result, msg); // 영상
        저장용(헤더)

    };
}
else // 웹에서 실행시(모아서 저장)
{
    videoBuffer.push(videoBufferHead[0]); // 헤더 넣기
    videoBuffer.push(videoBufferHead[1]); // 헤더 넣기
    recsize = videoBuffer[0].byteLength; // 파일 크기
    recsize += videoBuffer[1].byteLength; // 파일 크기
    a.text = iotlive_calsize(recsize); // 저장하기 버튼에 파일 사
    이즈 표시

    getHEADrec = true;

}
// 표시
$('#recbtn').closest('.ui-btn').hide();
$('#downloadLink').show();
$('#downloadLink').addClass('ui-btn-active'); // 활성 상태로 보이게
하기

}
}function iotlive_savefin()
{
    getSIDXrec = false;
    videoBuffer = [];
    getHEADrec = false; // 헤더 데이터가 들어갔는지 확인
    recnow = false; // 쓰기 작업여부

```

```

recfilename = ""; // 기록할 때 파일 이름
recsize = ""; // 기록 파일 크기 측정
// 표시
$('#recbtn').closest('.ui-btn').show();
$('#downloadLink').hide();
} // 파일이름 반환
function iotlive_filename()
{
    // 날짜 및 시간
    var dt = new Date();
    var year = dt.getFullYear();
    var month = dt.getMonth()+1;
    if (month < 10) month = '0' + month;
    var day = dt.getDate();
    if (day < 10) day = '0' + day;
    var hour = dt.getHours();
    if (hour < 10) hour = '0' + hour;
    var min = dt.getMinutes();
    if (min < 10) min = '0' + min;
    var sec = dt.getSeconds();
    if (sec < 10) sec = '0' + sec;
    return year + '-' + month + '-' + day + '_' + hour + ':' + min + ':' + sec +
'_iotcaps.mp4';
}
function iotlive_recwebblob(click)
{
    // 이전 blob주소 지우기
    if (recbloburl != "")
    {
        window.URL.revokeObjectURL(recbloburl);
        recbloburl = "";
    }
    if (videoBuffer.length > 0)
    {
        var a = $('#downloadLink')[0];
        var blob = new Blob(videoBuffer, {type: 'video/mp4'});
        a.download = recfilename;
        recbloburl = a.href = window.URL.createObjectURL(blob);
        if (click) a.click();
    }
} // 영상 저장
// click = 1 - 나가거나 분할 저장용
// click = 0 - 사용자가 클릭하여 저장

```

```

function iotlive_save(click)
{
    if (iotliveDebug) debugbox_msg_rec.innerHTML = "";
    if (runiotapp) $('#downloadLink')[0].href = "#"; // 앱에서 실행시
    else iotlive_recwebblob(click); // 웹에서 실행시
    iotlive_savefin();
}

// 단위 변환
function iotlive_calsize(bytesize)
{
    if (bytesize < 1024) // 바이트
        return bytesize + 'Byte';
    else if (bytesize < 1048576) // KB
        return (bytesize/1024).toFixed(2) + 'KB';
    else if (bytesize < 1073741824) // MB
        return (bytesize/1048576).toFixed(2) + 'MB';
    else // GB
        return (bytesize/1073741824).toFixed(2) + 'GB';
}

// 메모리 할당 해제
function iotlive_destroy()
{
    // 파일이 기록중이면 저장함
    iotlive_save(1);
    // 방 나가기
    socketok.emit('EXIT_REQ',[sessionStorage.getItem('id'),
sessionStorage.getItem('uid'), streamID]); // 인증 요청

    // 소스버퍼 해제
    try { mediaSource.removeSourceBuffer(sourceBuffer); } catch (e){}
    // 리스너 해제
    try { socketok.removeAllListeners(); } catch (e){}

    // 타이머 해제
    clearTimeout(syncTimer);
    syncTimer = null;
    clearTimeout(authTimer);
    authTimer = null;

    //////////////////////////////////////
    // 초기화
    try { $('#vtag')[0].src = ""; } catch (e){} // 미디어 해제
    socketok = null; // 웹소켓 데이터
    mediaSource = null; // 미디어 소스 API
    sourceBuffer = null; // 미디어 소스 API
    videoBufferHead = []; // 데이터 헤더

```

```

videoBuffer = []; // 데이터
getSIDXrec = false; // 처음 기록이 SIDX으로 했는지 확인
appendBuf = false; // 소스버퍼에 어펜드 여부
getHEADrec = false; // 헤더 데이터가 들어갔는지 확인
recnow = false; // 쓰기 작업여부
recfilename = ""; // 기록할 때 파일 이름
recsize = ""; // 기록 파일 크기 측정
queue = []; // 큐 데이터
streamQueue = []; // 영상데이터 큐
reaminQueue = []; // 남은 큐
streamDecProcess = false; // 임계구역 설정(병행수행시 충돌 방지)
getHeaderData = false; // 헤더 데이터 삽입 여부
getSIDXpla = false; // 처음 시작이 SIDX으로 했는지 확인
firstsidx = false; // 시작 인덱스인지 확인(채널을 변경하고 시작할 때 설정)
authloopcnt = 10; // 인증 루프 카운트(타임아웃 체크)
    datasize = 0; // 비트율 사이즈
    datapos = 0; // 비트율 계산용 시간
    datasizeprt = 0; // 디버그 프린트용 비트율 사이즈
} // 아이디 변경
function iotlive_selectCH(objvar)
{
    selectCH = true;
    if (sessionBool===true)
    {
        try { iotlive_destroy(); } catch (e){}
        iotlive_conf("wss://shana.pe.kr:1120/", objvar.value);
        iotlive_start();
    }
    if (sessionBool===false) history.back();
} // 상하 판넬 감추기
function iotlive_panel_hide()
{
    $('#bottomPanel').hide();
    $('#headerPanel').hide();
} // 상하 판넬 보이기
function iotlive_panel_show()
{
    // 목록이 있으면 보이기
    if ($('#iotlive-id')[0].childElementCount > 0)
    {
        $('#bottomPanel').show();
        $('#headerPanel').show();
    }
}

```

```

} // 영상 시작
function iotlive_start()
{
    // 디버그 정보
    iotlive_info_init();
    ///////////////
    // 복호화 수행을 위한 변수
    alertbox_msg.style.display = "block";
    alertbox_msg.innerHTML = "대기하는 중";
    if (selectCH)
    {
        decipheriv = window.prompt("다음 영상에 접근하려면 올바른 암호키
를 입력하세요.", undefined);
        if (decipheriv == null)
        {
            alertbox_msg.innerHTML = "영상 연결 취소";
            return;
        }
        alertbox_msg.innerHTML = "대기하는 중";

        // 암호가 입력되었으면
        if (decipheriv != "")
        {
            // 해시함수 처리
            var md = forge.md.sha256.create();
            md.update(decipheriv);
            decipheriv = md.digest().toHex();
        }
        selectCH = false;
    }

    socketok = io.connect(streamURL); // 영상 주소
    socketok.emit('GET_TYPE', streamID);
    socketok.on('GET_TYPE2', function (data)
    {
        if (videoBufferHead.length == 2) return;
        if (data == null)
        {
            alertbox_msg.innerHTML = streamID + "의 헤더데이터가 없
습니다.";
            return;
        }
    }
    );
}

```

```

    }
    var dataView = new DataView(data);
    if (dataView.getUint8(2).toString(16) == "2") // video
        playOption = "video";
    else
        playOption = "";

    if (window.MediaSource) mediaSource = new
window.MediaSource();
    else if (window.WebKitMediaSource) mediaSource = new
window.WebKitMediaSource();
    $('#vtag')[0].src = window.URL.createObjectURL(mediaSource); //
blob주소 얻기

    $('#vtag')[0].play(); // 재생 모드로 설정
    mediaSource.addEventListener('sourceopen', iotlive_callback,
false); // 콜백함수
    mediaSource.addEventListener('webkitsourceopen',
iotlive_callback, false); // 콜백함수
    // 인증 요청(1초간격 체크)
    alertbox_msg.innerHTML = "채널에 접근 권한이 있는지 확인 중";
    iotlive_authreq();
    clearTimeout(authTimer);
    authTimer = setInterval(iotlive_authreq, 1000);
});

// 헤더 데이터를 넣는 콜백 함수
socketok.on('HEADER_BUFFER', function (data)
{
    if (mediaSource.readyState == 'open' && !getHeaderData &&
videoBufferHead.length == 0)
    {
        // 헤더 데이터 삽입
        videoBufferHead.push(data[0]);
        videoBufferHead.push(data[1]);
        for (var i=0; i<2; i++) putData(data[i]);
    }
    if (videoBufferHead.length == 2)
    {
        getHeaderData = true;

        // 암호가 입력되었으면

```

```

        if (decipheriv != "")
            setInterval(iotlive_buffer_split, 10);
    }
});

    // 연결 재시도 함수
function iotlive_reconn()
{
    // 연결을 재시도한다.
    appendBuf = false;
    iotlive_destroy(); // 데스트로이
    iotlive_start(); // 시작
}

    // 인증 요청
function iotlive_authreq()
{
    if (videoBufferHead.length == 0)
        socketok.emit('AUTH_REQ',[sessionStorage.getItem('id'),
sessionStorage.getItem('uid'), streamID]);
    else if (videoBufferHead.length == 2)
    {
        // 타임아웃 체크기
        if (authloopcnt >= 6)
            alertbox_msg.innerHTML = streamID + "영상에 접속하는 중";
        else
            alertbox_msg.innerHTML = streamID + "영상에 접속하는 중 [" + authloopcnt + '초]';
            authloopcnt--;

        // 연결 재시도 요청
        if (authloopcnt<0)
        {
            alertbox_msg.innerHTML = "패스워드가 잘못 되었거나 연결할 수 없음";

            clearTimeout(authTimer);
            selectCH = true;
            iotlive_reconn();
            return;
        }
    }
}

```



```

'2'); // 영상 저장용(추가)
                                };
                                }
                                }
                                }
                                else // 웹web
                                {
                                    recsize += videoBuffer[videoBuffer.length-1].byteLength;
// 파일크기 측정
                                    $('#downloadLink').text(iotlive_calsize(recsize)); // 저장하
기 버튼에 파일 사이즈 표시
                                    if (getHEADrec && (recsize >= (recSizeWeb * 1024 *
1024))) // 분할 저장 부분
                                    {
                                        iotlive_save(1); // 저장
                                        iotlive_rec(1); // 다시 기록 시작
                                    }
                                }
                                }
                                // 데이터를 받아서 넣는 함수
                                function putData(data)
                                {
                                    appendBuf = true;
                                    try
                                    {
                                        if (sourceBuffer.updating || queue.length > 0)
queue.push(data);
                                        else sourceBuffer.appendBuffer(data);
                                    }
                                    catch (e)
                                    {
                                        iotlive_reconn();
                                    }
                                    appendBuf = false;
                                }
                                // 실시간으로 데이터를 받아오는 콜백 함수
                                socketok.on('DATA_BUFFER', function (data)
                                {
                                    // 비트율 계산용
                                    if (parseInt($('#vtag')[0].currentTime) == datapos) datasize +=

```

```

data.byteLength:
    else
    {
        datapos = parseInt($('#vtag')[0].currentTime);
        datasizeprt = datasize;
        datasize = 0;
    }

    // 암호가 입력되었으면
    if (decipheriv != "")
        streamQueue.push(data);
    else // 암호가 입력되지 않았으면
        iotlive_tls_proc(data);
});

// 암호를 입력받지 않고 TLS 전송만 처리
function iotlive_tls_proc(data)
{
    // 디버그 화면
    if (iotliveDebug)
        debugbox_msg_cip.innerHTML = '전송타입: TLS(HTTPS)' +

'<br>DASH사용: yes' +

'<br>버퍼모드: ' + mediaSource.activeSourceBuffers[0].mode +

'<br>해상도: ' + $('#vtag')[0].videoWidth + 'x' + $('#vtag')[0].videoHeight +

'<br>비트레이트: ' + datasizeprt*8/1000 + ' kbps' +

'<br>재생시간: ' + $('#vtag')[0].currentTime + ' 초' +

'<br>데이터: ' + data.byteLength + ' 바이트';

    try
    {
        if (mediaSource.readyState == 'open' && getHeaderData
&& typeof data !== 'string')
        {
            var sidx = false; // 시작 인덱스인지 확인

```

	// 시작 데이터인지 확인
	if (iotlive_chk4sidx(data) == 'eq') sidx = true;
	else sidx = false;
	// 녹화 데이터 삽입
데이터부터 넣기	if (getHEADrec && !getSIDXrec && sidx) // 시작
	{
	putDataRec(data);
	getSIDXrec=true;
	}
터 들어가면 이어서 넣기	else if (getHEADrec && getSIDXrec) // 시작 데이
	putDataRec(data);
	// 재생 데이터 삽입
작 데이터부터 넣기	if (getHeaderData && !getSIDXpla && sidx) // 시
	{
	putData(data);
	getSIDXpla=true;
	}
이터 들어가면 이어서 넣기	else if (getHeaderData && getSIDXpla) // 시작 데
	putData(data);
	}
	}
	catch (e)
	{
	}
	}
	// 숫자인지 체크
	function isNumber(s)
	{
	s += '';
	s = s.replace(/^s* \s*\$/g, '');
	if (s == '' isNaN(s)) return false;
	return true;
	}
	// 버퍼 정리(큐에 쌓고 플래그에 정해진 암호문의 길이를 추출하여 그

길이만 복호화해 큐에 저장)

```
function iotlive_buffer_split()
{
    if (streamDecProcess || streamQueue.length < 5) return;
    streamDecProcess = true;
    try
    {
        var arr = new Uint8Array(streamQueue.shift());
        if (reaminQueue.length == 0 && arr.length == 0) {
streamDecProcess = false; return; }
        var snumarr = [];
        var snumdec = "";
        var dataarr = [];
        // 잉여분 합치기
        for (var i=0; i<reaminQueue.length; i++)
dataarr.push(reaminQueue[i]); dataarr = [];

        // 데이터 처리 함수
        for (var i=0; i<arr.length; i++) { if (i<16)
snumarr.push(arr[i]); else dataarr.push(arr[i]); }
        // 길이 가져오기
        for (var i=0; i<16; i++) { if (snumarr[i] == 0) break;
snumdec += String.fromCharCode(snumarr[i]); }

        // 시작 데이터가 없고 숫자가 아니면.. 스킵
        if (firstsidx == false && isNumber(snumdec) == false)
        {
            streamDecProcess = false;
            return;
        }

        // 같으면
        if (dataarr.length == snumdec && snumdec != "")
        {
            iotlive_decipher(dataarr);
        }
        else // 같지 않으면 이어 붙이기.
        {
            while(dataarr.length != snumdec)
            {
```

```

var arr2 = new
Uint8Array(streamQueue.shift());
for (var i=0; i<arr2.length; i++)
{
    if (dataarr.length < snumdec)
dataarr.push(arr2[i]);
    else reaminQueue.push(arr2[i]);
}
if (arr2.length == 0)
{
    arr2 = NULL; // 메모리 해제
    break;
}

}

if (dataarr.length != snumdec)
{
    if (iotliveDebug)
        debugbox_msg_dat.innerHTML = '
필요 데이터: ' + snumdec + '<br>현재 데이터: ' + dataarr.length;
    }
    else
        iotlive_decipher(dataarr);
}
arr = NULL; // 메모리 해제
}
catch (e) { }
streamDecProcess = false;
}
function iotlive_decipher(arr)
{
    var arraybuf = new Uint8Array(arr);
    var decipher = forge.cipher.createDecipher('AES-CBC', ['0', '0',
'0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0']);
    decipher.start({iv:decipheriv});
    decipher.update(forge.util.createBuffer(arraybuf));
    if (decipher.finish() === true)
    {
        var decstorage = new

```

```

ArrayBuffer(decipher.output.data.length);
        var decstorageDV = new DataView(decstorage);
        for (var ci=0; ci<decstorage.byteLength; ci++)
            d e c s t o r a g e D V . s e t U i n t 8 ( c i ,
decipher.output.data.charCodeAt(ci).toString(10));
        if (iotliveDebug)
            debugbox_msg_cip.innerHTML = '암호알고리즘:
AES-CBC(IOTCAM)' +

'<br>전송타입: TLS(HTTPS)' +

'<br>DASH사용: yes' +

'<br>버퍼모드: ' + mediaSource.activeSourceBuffers[0].mode +

'<br>해상도: ' + $('#vtag')[0].videoWidth + 'x' + $('#vtag')[0].videoHeight
+

'<br>비트레이트: ' + datasizeprt*8/1000 + ' kbps' +

'<br>재생시간: ' + $('#vtag')[0].currentTime + ' 초' +

'<br>암호문: ' + arraybuf.length + ' 바이트' +

'<br>평문: ' + decstorage.byteLength + ' 바이트' +

'<br>큐사이즈: ' + streamQueue.length;
        iotlive_push(decstorage);
        decstorage = NULL; // 메모리 해제
        decstorageDV = NULL; // 메모리 해제
    }
    decipher = NULL; // 메모리 해제
    arraybuf = NULL; // 메모리 해제
}

function iotlive_push(decstorage)
{
    var sidx = false; // 시작 인덱스인지 확인
    if (mediaSource.readyState == 'open' && getHeaderData)
    {

```

```

// 시작 데이터인지 확인
if (iotlive_chk4sidx(decstorage) == 'eq') { firstsidx = true;
sidx = true; }

else sidx = false;
// 녹화 데이터 삽입
if (getHEADrec && !getSIDXrec && sidx) // 시작 데이터부
터 넣기
{
    putDataRec(decstorage);
    getSIDXrec=true;
}
else if (getHEADrec && getSIDXrec) // 시작 데이터 들어가
면 이어서 넣기
    putDataRec(decstorage);

// 재생 데이터 삽입
if (getHeaderData && !getSIDXpla && sidx) // 시작 데이터
부터 넣기
{
    putData(decstorage);
    getSIDXpla=true;
}
else if (getHeaderData && getSIDXpla) // 시작 데이터 들어
가면 이어서 넣기
{
    putData(decstorage);
}
}

// 큐에 쌓인 데이터를 처리하는 함수
function iotlive_callback(e)
{
    if (playOption == "video")
        sourceBuffer = mediaSource.addSourceBuffer('video/mp4;
codecs="avc1.64001E");
    else
        sourceBuffer = mediaSource.addSourceBuffer('video/mp4;
codecs="avc1.64001E, mp4a.40.2");

    if (isMSEdge) // 엣지

```

```

        sourceBuffer.mode = 'sequence'; // 0초부터 시작
    else // 크롬
    {
        if (playOption == "video")
            sourceBuffer.mode = 'sequence'; // 0초부터 시작
        else
            sourceBuffer.mode = 'segments'; // chrome 50
    }

    sourceBuffer.addEventListener('update', function()
    {
        try
        {
            if (queue.length > 0 && !sourceBuffer.updating
            && !appendBuf)
                sourceBuffer.appendBuffer(queue.shift());
        }
        catch (e) {}
    });
}

```

iotsession.js

```

/**
 * Created by LEEKIWON on 2015-11-08.
 * Last modification date 2016-05-15.
 */
var sessionSocket = io.connect('wss://shana.pe.kr:1115');
var sessionBool:
    function sessionRefresh()
{
    var page = $.mobile.activePage.attr("id");
    if (page=="index")
    {
        if (sessionBool===true)
        {
            $("#memberlayout").empty();
            $("#menulayout").empty();
            $('#iot_logininfo')[0].innerHTML = "<b>" +
            sessionStorage.getItem('id') + "</b>님 안녕하세요?<br />";
            $("#memberlayout").append("<li><a data-transition='slide'

```

```

href='logout.html'><span class='glyphicon glyphicon-log-out'></span>&nbsp;로그
아웃</a></li>");
        $("#menulayout").append("<li><a      data-transition='slide'
href='live.html'>영상 보기</a></li>");
        $("#menulayout").append("<li><a      data-transition='slide'
href='config.html'>영상 접근제어</a></li>");
    }
    if (sessionBool===false)
    {
        setTimeout("window.location.assign('login.html');", 1300);
        $('body').addClass('ui-page-theme-a');
        $('body').addClass('bgimg');
        $('body').fadeOut(1500);
        document.body.innerHTML = "<div
style='position:absolute; top:50%; left:50%; overflow:hidden; margin-top:-43px;
margin-left:-139px'><center><img src='css/images/security86.png' height='86px'
alt='logo' /></center></div>";
        /*
        $("#memberlayout").empty();
        $("#menulayout").empty();
        $('#iot_logininfo')[0].innerHTML = "서비스를 이용하려면 로
그인하세요.<br />";
        $("#memberlayout").append("<li><a      data-transition='slide'
href='join.html'><span class='glyphicon glyphicon-user'></span>&nbsp;회원가입
</a></li>");
        $("#memberlayout").append("<li><a      data-transition='slide'
href='login.html'><span class='glyphicon glyphicon-log-in'></span>&nbsp;로그인
</a></li>");
        */
    }
}
else if(page=="login")
{
    if (sessionBool===true) history.back();
    document.getElementById('inputfile').addEventListener('change',
readFile, false);
}
else if(page=="logout")
{
    if (sessionBool===true)

```

```

        {
            delete sessionStorage.id;
            delete sessionStorage.uid;
            sessionBool = undefined;
        }
        history.back();
    }
    else if(page=="join")
    {
        if (sessionBool===true)
        {
            alert('이미 가입하셨습니다!');
            history.back();
        }
    }
    else if(page=="live")
    {
        if (sessionBool===false) history.back();
    }
    else if(page=="config")
    {
        if (sessionBool===false) history.back();
    }
}function iotlive_resize()
{
    $('#vtag').css('height',$ (window).height());
}$(document).on("pageshow", "#live", function()
{
    iotlive_resize();
    $("#vtag").click(function() {
        // 목록이 있으면 토글
        if ($('#iotlive-id')[0].childElementCount > 0)
        {
            $('#bottomPanel').toggle();
            $('#headerPanel').toggle();
        }
    });
});
$(document).ready(function() {
    $(window).resize(function(){
        iotlive_resize();
    });
});

```

```

});
$(document).on("pageshow", "#config", function()
{
    $('#config-id').children('li').on('click', function ()
    {
        // 아이디 삭제
        var selch = $(this).attr('data-name');
        if (sessionStorage.getItem('id') == selch)
        {
            alert("자기 자신을 삭제할 수 없습니다.");
            return;
        }
        var retVal = confirm(selch + "을(를) 삭제하겠습니까?\n삭제 후에는  
해당 영상에 접근할 수 없습니다.");
        if (retVal != true) return; // 삭제 취소
        // 삭제 요청
        sessionSocket.emit('ID_DEL', [sessionStorage.getItem('id'), selch]);

        sessionSocket.on('ID_DELFAIL', function(data) // 실패
        {
            sessionSocket.removeAllListeners();
            alert('삭제 실패!');
        });
        sessionSocket.on('ID_DELOK', function(data) // 성공
        {
            sessionSocket.removeAllListeners();
            $("#config-id").find("li:contains(" + selch + ")").remove();
            alert('아이디 ' + selch + '이(가) 삭제되었습니다.');
        });
    });
});

$('#config-req-id').children('li').on('click', function ()
{
    // 아이디 수락
    var selch = $(this).attr('data-name');
    var retVal = confirm(selch + "이(가) 나의 영상을 볼 수 있도록 수락  
하겠습니까?");
    if (retVal != true) return; // 수락 취소

    // 수락 요청
    sessionSocket.emit('ID_APTREQ', [sessionStorage.getItem('id'),

```

```

selch]);

        sessionSocket.on('ID_APTREQFAIL', function(data) // 실패
        {
            sessionSocket.removeAllListeners();
            alert('수락 실패!');
        });
        sessionSocket.on('ID_APTREQOK', function(data) // 성공
        {
            sessionSocket.removeAllListeners();
            $("#config-req-id").find("li:contains(" + selch +
            ")).remove();

            alert('아이디 ' + selch + '이(가) 수락되었습니다.');
        });
    });
    // 아이디 추가 요청
    function chAddReq()
    {
        // 체크
        for (var i=0; i<$("#config-id").children("li").length; i++)
        {
            var ids = $("#config-id").children("li").eq(i).attr("data-name");
            if ($('#configaddid')[0].value.toLowerCase() == ids)
            {
                alert('해당 아이디는 이미 추가되어있습니다!');
                return;
            }
        }

        // 정보 보내기
        sessionSocket.emit('ID_ADDREQ', [sessionStorage.getItem('id'),
        $('#configaddid')[0].value.toLowerCase()]);
        sessionSocket.on('ID_ADDREQOK', function(data)
        {
            sessionSocket.removeAllListeners();
            alert('아이디 추가요청을 하였습니다.');
        });
        sessionSocket.on('ID_ADDREQFAIL', function(data)
        {
            sessionSocket.removeAllListeners();
            alert('아이디 추가요청 실패!');
        });
    }

```

```

}function getChConfig()
{
    ///////////////////////////////////////////////////
    //아이디 삭제 부
    ///////////////////////////////////////////////////
        // 아이디 받아오기
    sessionSocket.emit('ID_REQ', sessionStorage.getItem('id'));
    // 변수
    var firstid="";
    sessionSocket.on('ID_OKAY', function(data) // 성공
    {
        //sessionSocket.removeAllListeners();
        $('#config-id').empty(); // 목록 비움

        var chs = data.split(',');
        for (var i=0; i<chs.length; i++)
        {
            var detach = chs[i].split(':');
            if (detach[0] == "") break;
            $('#config-id').append('<li data-icon="delete" data-name="'
+ detach[1] + "><a href="#">' + detach[0] + '</a></li>').listview("refresh");
        }
    });

    sessionSocket.on('ID_FAIL', function(data) // 실패
    {
        sessionSocket.removeAllListeners();
        alert('아이디가 없습니다!');
        history.back();
    });

    ///////////////////////////////////////////////////
    //요청 수락 부
    ///////////////////////////////////////////////////
    // reqdb 받아오기
    sessionSocket.emit('ID_REQDBREQ', sessionStorage.getItem('id'));
    // 변수
    sessionSocket.on('ID_REQDBREQOKAY', function(data) // 성공
    {
        sessionSocket.removeAllListeners();
        $('#config-req-id').empty(); // 목록 비움

        var chs = data.split(',');
    }

```

```

        for (var i=0; i<chs.length; i++)
        {
            if (chs[i] == "") break;
            $('#config-req-id').append('<li data-icon="check"
data-name="" + chs[i] + "><a href="#">' + chs[i] + '</a></li>').listview("refresh");
        }
    });

    sessionSocket.on('ID_REQDBREQFAIL', function(data) // 실패
    {
        sessionSocket.removeAllListeners();
    });
}function getChLive()
{
    // 상하 패널 감추기
    iotlive_panel_hide();
    // 상단 패널 투명도 설정
    $("#iotlive-id-button").css('border', '0');
    $("#iotlive-id-button").css('background-color', 'rgba(34, 34, 34, 0.4)');
    // 아이디 받아오기
    setTimeout(function()
    {

        // 요청
        sessionSocket.emit('ID_REQ', sessionStorage.getItem('id'));

        var live_firstid="";
        sessionSocket.on('ID_OKAY', function(data) // 성공
        {
            sessionSocket.removeAllListeners();
            $('#iotlive-id').empty(); // 목록 비움

            var chs = data.split(',');
            for (var i=0; i<chs.length; i++)
            {
                var detach = chs[i].split(':');
                if (detach[0] == "") break;

                $('#iotlive-id').append('<option value="" + detach[1]
+ ">' + detach[0] + '</option>').selectmenu("refresh");
                // 첫 번째 영상 선택
                if (live_firstid == "") live_firstid = detach[1];
            }
        });
    }, 1000);
}

```

```

    }

    iotlive_panel_show(); // 상하 패널 보이기

    // 영상 재생 시작
    var obj = {value: live_firstid};
    iotlive_selectCH(obj);
});

sessionSocket.on('ID_FAIL', function(data) // 실패
{
    sessionSocket.removeAllListeners();
    alert('아이디가 없습니다!');
    history.back();
});

    }, 100);
}$(document).on("pagecreate", "#config", getChConfig);
$(document).on("pagecreate", "#live", getChLive);
$(document).on("pageshow", function()
{
    if (sessionStorage.getItem('id') !== null && sessionStorage.getItem('uid')
    !== null)
    {
        sessionSocket.emit('SESSION_REQ', [sessionStorage.getItem('id'),
sessionStorage.getItem('uid')]);
        sessionSocket.on('SESSION_OKAY', function(data)
        {
            sessionSocket.removeAllListeners();
            sessionStorage.setItem('id', data[0]);
            sessionStorage.setItem('uid', data[1]);
            sessionBool=true;
            sessionRefresh();
        });
        sessionSocket.on('SESSION_FAIL', function(data)
        {
            sessionSocket.removeAllListeners();
            sessionBool=false;
            sessionRefresh();
        });
    }
}

```

```
        else
        {
            sessionSocket.removeAllListeners();
            sessionBool=false;
            sessionRefresh();
        }
    });
    $(document).on("pagebeforehide","#live",function()
    {
        try { iotlive_destroy(); } catch (e){}
    });
```

IOTCAM

iotcam.c

```
/**
 * Created by LEEKIWON on 2015-09-08.
 * Last modification date 2016-05-15.
 */
/*
    IOTCAPS 졸업작품
    카메라 -> 서버 전송 프로그램
 */
//-----
// iotcam 라이브러리
//-----
#include "iotcam.h"
#ifdef IOT_VS_TESTMODE
#include <stdio.h>
#include <io.h>
#endif
#include <ctype.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
#include <errno.h>
#include <limits.h>
#include <stdint.h>
#define CONFIG_PATH "iotcam.c25"
#define IOTCAPS_SVR "https://shana.pe.kr:8080/"
//-----
// iotcam 전역변수
//-----
char *iotcam_dshow_video[32];
char *iotcam_dshow_audio[32];
int iotcam_dshow_cntv=0;
int iotcam_dshow_cnta=0;
int iotcam_exit_sync=0;
int iotcam_errorcode=0;
iotcamfs iot_conf_data;
char **iotcam_global_argv;
unsigned char iotcam_stream_pw[128];
unsigned char iotcam_pbUserKey[16] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}; // 맷주소 기반 키
unsigned int iotcam_encrypt_mode;
//-----
// iotcam 함수선언
//-----
static int iotcam_config(int newconf);
static void iotcam_view_config(void);
//-----
// KISA_SHA256.c (선언)
// 출처 http://seed.kisa.or.kr/ -> 알림마당 -> 자료실
//-----
#ifndef OUT
#define OUT
#endif
#ifndef IN
#define IN
#endif
#ifndef INOUT
#define INOUT
#endif
#ifndef BIG_ENDIAN
#define BIG_ENDIAN
#endif
#ifndef LITTLE_ENDIAN
#define LITTLE_ENDIAN
#endif
#if defined(USER_BIG_ENDIAN)
    #define BIG_ENDIAN
#elif defined(USER_LITTLE_ENDIAN)
    #define LITTLE_ENDIAN
#else
    #if 0
        #define BIG_ENDIAN
    #else
        #define LITTLE_ENDIAN
    #endif
#endif
typedef unsigned char SHA256_BYTE;
typedef unsigned long* SHA256_ULONG_PTR;
typedef unsigned char* SHA256_UCHAR_PTR;
#define SHA256_DIGEST_BLOCKLEN    64
#define SHA256_DIGEST_VALUELEN    32
typedef struct{
    unsigned int uChainVar[SHA256_DIGEST_VALUELEN / 4];

```

```

        unsigned int uHighLength;
        unsigned int uLowLength;
        SHA256_BYTE szBuffer[SHA256_DIGEST_BLOCKLEN];
    } SHA256_INFO;
    const unsigned int SHA256_K[64] =
    {
        0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5, 0x3956c25b, 0x59f111f1,
        0x923f82a4, 0xab1c5ed5, 0xd807aa98, 0x12835b01, 0x243185be,
        0x550c7dc3,
        0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174, 0xe49b69c1,
        0xefbe4786,
        0x0fc19dc6, 0x240ca1cc, 0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc,
        0x76f988da,
        0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3,
        0xd5a79147,
        0x06ca6351, 0x14292967, 0x27b70a85, 0x2e1b2138, 0x4d2c6dfc,
        0x53380d13,
        0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85, 0xa2bfe8a1,
        0xa81a664b,
        0xc24b8b70, 0xc76c51a3, 0xd192e819, 0xd6990624, 0xf40e3585,
        0x106aa070,
        0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5, 0x391c0cb3,
        0x4ed8aa4a,
        0x5b9cca4f, 0x682e6ff3, 0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208,
        0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2
    };
    #if defined(_MSC_VER)
        #define ROTL_ULONG(x, n) _lrotl((x), (n))
        #define ROTR_ULONG(x, n) _lrotr((x), (n))
    #else
        #define ROTL_ULONG(x, n) ((unsigned long)((x) << (n)) | (unsigned
long)((x) >> (32 - (n))))
        #define ROTR_ULONG(x, n) ((unsigned long)((x) >> (n)) | (unsigned
long)((x) << (32 - (n))))
    #endif
    #define ENDIAN_REVERSE_ULONG(dwS) ( (ROTL_ULONG((dwS), 8) & 0x00ff00ff) \
|
(ROTL_ULONG((dwS), 24) & 0xff00ff00) )
    #if defined(BIG_ENDIAN)
        #define BIG_B2D(B, D)          D = *(SHA256_ULONG_PTR)(B)
        #define BIG_D2B(D, B)          *(SHA256_ULONG_PTR)(B) = (ULONG)(D)
    #endif

```

```

#define LITTLE_B2D(B, D)      D =
ENDIAN_REVERSE_ULONG(*(SHA256_ULONG_PTR)(B))
#define LITTLE_D2B(D, B)      *(SHA256_ULONG_PTR)(B) =
ENDIAN_REVERSE_ULONG(D)
#elif defined(LITTLE_ENDIAN)
#define BIG_B2D(B, D)          D =
ENDIAN_REVERSE_ULONG(*(SHA256_ULONG_PTR)(B))
#define BIG_D2B(D, B)          *(SHA256_ULONG_PTR)(B) =
ENDIAN_REVERSE_ULONG(D)
#define LITTLE_B2D(B, D)      D = *(SHA256_ULONG_PTR)(B)
#define LITTLE_D2B(D, B)      *(SHA256_ULONG_PTR)(B) = (ULONG)(D)
#else
#error ERROR : Invalid DataChangeType
#endif
#define RR(x, n)               ROTR_ULONG(x, n)
#define SS(x, n)               (x >> n)
#define Ch(x, y, z)            ((x & y) ^ ((~x) & z))
#define Maj(x, y, z)           ((x & y) ^ (x & z) ^ (y & z))
#define Sigma0(x)              (RR(x, 2) ^ RR(x, 13) ^ RR(x, 22))
#define Sigma1(x)              (RR(x, 6) ^ RR(x, 11) ^ RR(x, 25))
#define RHO0(x)                (RR(x, 7) ^ RR(x, 18) ^ SS(x, 3))
#define RHO1(x)                (RR(x, 17) ^ RR(x, 19) ^ SS(x, 10))
//-----
// KISA_SEED_ECB.c (선언)
// 출처 http://seed.kisa.or.kr/ -> 알림마당 -> 자료실
//-----
// macros for left or right rotations
#if defined(_MSC_VER)
#define ROTL(x, n)              (_lrotl((x), (n)))           // left rotation
#define ROTR(x, n)              (_lrotr((x), (n)))           // right rotation
#else
#define ROTL(x, n)              (((x) << (n)) | ((x) >> (32-(n)))) // left
rotation
#define ROTR(x, n)              (((x) >> (n)) | ((x) << (32-(n)))) // right
rotation
#endif
// macroses for converting endianess
#define EndianChange(dwS)      \
( (ROTL((dwS), 8) & (unsigned long int)0x00ff00ff) | \
(ROTL((dwS), 24) & (unsigned long int)0xff00ff00) )

```

```
static unsigned long int SS0[256] = {
0x2989a1a8, 0x05858184, 0x16c6d2d4, 0x13c3d3d0, 0x14445054, 0x1d0d111c,
0x2c8ca0ac, 0x25052124,
0x1d4d515c, 0x03434340, 0x18081018, 0x1e0e121c, 0x11415150, 0x3cccf0fc,
0x0acac2c8, 0x23436360,
0x28082028, 0x04444044, 0x20002020, 0x1d8d919c, 0x20c0e0e0, 0x22c2e2e0,
0x08c8c0c8, 0x17071314,
0x2585a1a4, 0x0f8f838c, 0x03030300, 0x3b4b7378, 0x3b8bb3b8, 0x13031310,
0x12c2d2d0, 0x2ecee2ec,
0x30407070, 0x0c8c808c, 0x3f0f333c, 0x2888a0a8, 0x32023230, 0x1dcdd1dc,
0x36c6f2f4, 0x34447074,
0x2ccce0ec, 0x15859194, 0x0b0b0308, 0x17475354, 0x1c4c505c, 0x1b4b5358,
0x3d8db1bc, 0x01010100,
0x24042024, 0x1c0c101c, 0x33437370, 0x18889098, 0x10001010, 0x0cccc0cc,
0x32c2f2f0, 0x19c9d1d8,
0x2c0c202c, 0x27c7e3e4, 0x32427270, 0x03838380, 0x1b8b9398, 0x11c1d1d0,
0x06868284, 0x09c9c1c8,
0x20406060, 0x10405050, 0x2383a3a0, 0x2bcbe3e8, 0x0d0d010c, 0x3686b2b4,
0x1e8e929c, 0x0f4f434c,
0x3787b3b4, 0x1a4a5258, 0x06c6c2c4, 0x38487078, 0x2686a2a4, 0x12021210,
0x2f8fa3ac, 0x15c5d1d4,
0x21416160, 0x03c3c3c0, 0x3484b0b4, 0x01414140, 0x12425250, 0x3d4d717c,
0x0d8d818c, 0x08080008,
0x1f0f131c, 0x19899198, 0x00000000, 0x19091118, 0x04040004, 0x13435350,
0x37c7f3f4, 0x21c1e1e0,
0x3dcd1fc, 0x36467274, 0x2f0f232c, 0x27072324, 0x3080b0b0, 0x0b8b8388,
0x0e0e020c, 0x2b8ba3a8,
0x2282a2a0, 0x2e4e626c, 0x13839390, 0x0d4d414c, 0x29496168, 0x3c4c707c,
0x09090108, 0x0a0a0208,
0x3f8fb3bc, 0x2fcfe3ec, 0x33c3f3f0, 0x05c5c1c4, 0x07878384, 0x14041014,
0x3ecef2fc, 0x24446064,
0x1eced2dc, 0x2e0e222c, 0x0b4b4348, 0x1a0a1218, 0x06060204, 0x21012120,
0x2b4b6368, 0x26466264,
0x02020200, 0x35c5f1f4, 0x12829290, 0x0a8a8288, 0x0c0c000c, 0x3383b3b0,
0x3e4e727c, 0x10c0d0d0,
0x3a4a7278, 0x07474344, 0x16869294, 0x25c5e1e4, 0x26062224, 0x00808080,
0x2d8dalac, 0x1fcfd3dc,
0x2181a1a0, 0x30003030, 0x37073334, 0x2e8ea2ac, 0x36063234, 0x15051114,
0x22022220, 0x38083038,
0x34c4f0f4, 0x2787a3a4, 0x05454144, 0x0c4c404c, 0x01818180, 0x29c9e1e8,
```

```

0x04848084, 0x17879394,
0x35053134, 0x0bcb3c8, 0x0ecec2cc, 0x3c0c303c, 0x31417170, 0x11011110,
0x07c7c3c4, 0x09898188,
0x35457174, 0x3bcbf3f8, 0x1acad2d8, 0x38c8f0f8, 0x14849094, 0x19495158,
0x02828280, 0x04c4c0c4,
0x3fcff3fc, 0x09494148, 0x39093138, 0x27476364, 0x00c0c0c0, 0x0fcfc3cc,
0x17c7d3d4, 0x3888b0b8,
0x0f0f030c, 0x0e8e828c, 0x02424240, 0x23032320, 0x11819190, 0x2c4c606c,
0x1bcbd3d8, 0x2484a0a4,
0x34043034, 0x31c1f1f0, 0x08484048, 0x02c2c2c0, 0x2f4f636c, 0x3d0d313c,
0x2d0d212c, 0x00404040,
0x3e8eb2bc, 0x3e0e323c, 0x3c8cb0bc, 0x01c1c1c0, 0x2a8aa2a8, 0x3a8ab2b8,
0x0e4e424c, 0x15455154,
0x3b0b3338, 0x1cccd0dc, 0x28486068, 0x3f4f737c, 0x1c8c909c, 0x18c8d0d8,
0x0a4a4248, 0x16465254,
0x37477374, 0x2080a0a0, 0x2dcde1ec, 0x06464244, 0x3585b1b4, 0x2b0b2328,
0x25456164, 0x3acaf2f8,
0x23c3e3e0, 0x3989b1b8, 0x3181b1b0, 0x1f8f939c, 0x1e4e525c, 0x39c9f1f8,
0x26c6e2e4, 0x3282b2b0,
0x31013130, 0x2acae2e8, 0x2d4d616c, 0x1f4f535c, 0x24c4e0e4, 0x30c0f0f0,
0x0dcdc1cc, 0x08888088,
0x16061214, 0x3a0a3238, 0x18485058, 0x14c4d0d4, 0x22426260, 0x29092128,
0x07070304, 0x33033330,
0x28c8e0e8, 0x1b0b1318, 0x05050104, 0x39497178, 0x10809090, 0x2a4a6268,
0x2a0a2228, 0x1a8a9298
};

static unsigned long int SS1[256] = {
0x38380830, 0xe828c8e0, 0x2c2d0d21, 0xa42686a2, 0xcc0fcfc3, 0xdc1eced2,
0xb03383b3, 0xb83888b0,
0xac2f8fa3, 0x60204060, 0x54154551, 0xc407c7c3, 0x44044440, 0x6c2f4f63,
0x682b4b63, 0x581b4b53,
0xc003c3c3, 0x60224262, 0x30330333, 0xb43585b1, 0x28290921, 0xa02080a0,
0xe022c2e2, 0xa42787a3,
0xd013c3d3, 0x90118191, 0x10110111, 0x04060602, 0x1c1c0c10, 0xbc3c8cb0,
0x34360632, 0x480b4b43,
0xec2fcfe3, 0x88088880, 0x6c2c4c60, 0xa82888a0, 0x14170713, 0xc404c4c0,
0x14160612, 0xf434c4f0,
0xc002c2c2, 0x44054541, 0xe021c1e1, 0xd416c6d2, 0x3c3f0f33, 0x3c3d0d31,
0x8c0e8e82, 0x98188890,
0x28280820, 0x4c0e4e42, 0xf436c6f2, 0x3c3e0e32, 0xa42585a1, 0xf839c9f1,

```

0x0c0d0d01,	0xdc1fcfd3,				
0xd818c8d0,	0x282b0b23,	0x64264662,	0x783a4a72,	0x24270723,	0x2c2f0f23,
0xf031c1f1,	0x70324272,				
0x40024242,	0xd414c4d0,	0x40014141,	0xc000c0c0,	0x70334373,	0x64274763,
0xac2c8ca0,	0x880b8b83,				
0xf437c7f3,	0xac2d8da1,	0x80008080,	0x1c1f0f13,	0xc80acac2,	0x2c2c0c20,
0xa82a8aa2,	0x34340430,				
0xd012c2d2,	0x080b0b03,	0xec2ecee2,	0xe829c9e1,	0x5c1d4d51,	0x94148490,
0x18180810,	0xf838c8f0,				
0x54174753,	0xac2e8ea2,	0x08080800,	0xc405c5c1,	0x10130313,	0xcc0dcdc1,
0x84068682,	0xb83989b1,				
0xfc3fcff3,	0x7c3d4d71,	0xc001c1c1,	0x30310131,	0xf435c5f1,	0x880a8a82,
0x682a4a62,	0xb03181b1,				
0xd011c1d1,	0x20200020,	0xd417c7d3,	0x00020202,	0x20220222,	0x04040400,
0x68284860,	0x70314171,				
0x04070703,	0xd81bcbd3,	0x9c1d8d91,	0x98198991,	0x60214161,	0xbc3e8eb2,
0xe426c6e2,	0x58194951,				
0xdc1dcdd1,	0x50114151,	0x90108090,	0xdc1cccd0,	0x981a8a92,	0xa02383a3,
0xa82b8ba3,	0xd010c0d0,				
0x80018181,	0x0c0f0f03,	0x44074743,	0x181a0a12,	0xe023c3e3,	0xec2ccce0,
0x8c0d8d81,	0xbc3f8fb3,				
0x94168692,	0x783b4b73,	0x5c1c4c50,	0xa02282a2,	0xa02181a1,	0x60234363,
0x20230323,	0x4c0d4d41,				
0xc808c8c0,	0x9c1e8e92,	0x9c1c8c90,	0x383a0a32,	0x0c0c0c00,	0x2c2e0e22,
0xb83a8ab2,	0x6c2e4e62,				
0x9c1f8f93,	0x581a4a52,	0xf032c2f2,	0x90128292,	0xf033c3f3,	0x48094941,
0x78384870,	0xcc0cccc0,				
0x14150511,	0xf83bcbf3,	0x70304070,	0x74354571,	0x7c3f4f73,	0x34350531,
0x10100010,	0x00030303,				
0x64244460,	0x6c2d4d61,	0xc406c6c2,	0x74344470,	0xd415c5d1,	0xb43484b0,
0xe82acae2,	0x08090901,				
0x74364672,	0x18190911,	0xfc3ecef2,	0x40004040,	0x10120212,	0xe020c0e0,
0xbc3d8db1,	0x04050501,				
0xf83acaf2,	0x00010101,	0xf030c0f0,	0x282a0a22,	0x5c1e4e52,	0xa82989a1,
0x54164652,	0x40034343,				
0x84058581,	0x14140410,	0x88098981,	0x981b8b93,	0xb03080b0,	0xe425c5e1,
0x48084840,	0x78394971,				
0x94178793,	0xfc3cccf0,	0x1c1e0e12,	0x80028282,	0x20210121,	0x8c0c8c80,
0x181b0b13,	0x5c1f4f53,				
0x74374773,	0x54144450,	0xb03282b2,	0x1c1d0d11,	0x24250521,	0x4c0f4f43,

```
0x00000000, 0x44064642,
0xec2dcde1, 0x58184850, 0x50124252, 0xe82bcbe3, 0x7c3e4e72, 0xd81acad2,
0xc809c9c1, 0xfc3dcdf1,
0x30300030, 0x94158591, 0x64254561, 0x3c3c0c30, 0xb43686b2, 0xe424c4e0,
0xb83b8bb3, 0x7c3c4c70,
0x0c0e0e02, 0x50104050, 0x38390931, 0x24260622, 0x30320232, 0x84048480,
0x68294961, 0x90138393,
0x34370733, 0xe427c7e3, 0x24240420, 0xa42484a0, 0xc80bcbcb, 0x50134353,
0x080a0a02, 0x84078783,
0xd819c9d1, 0x4c0c4c40, 0x80038383, 0x8c0f8f83, 0xcc0ecec2, 0x383b0b33,
0x480a4a42, 0xb43787b3
};
static unsigned long int SS2[256] = {
0xa1a82989, 0x81840585, 0xd2d416c6, 0xd3d013c3, 0x50541444, 0x111c1d0d,
0xa0ac2c8c, 0x21242505,
0x515c1d4d, 0x43400343, 0x10181808, 0x121c1e0e, 0x51501141, 0xf0fc3ccc,
0xc2c80aca, 0x63602343,
0x20282808, 0x40440444, 0x20202000, 0x919c1d8d, 0xe0e020c0, 0xe2e022c2,
0xc0c808c8, 0x13141707,
0xa1a42585, 0x838c0f8f, 0x03000303, 0x73783b4b, 0xb3b83b8b, 0x13101303,
0xd2d012c2, 0xe2ec2ece,
0x70703040, 0x808c0c8c, 0x333c3f0f, 0xa0a82888, 0x32303202, 0xd1dc1dcd,
0xf2f436c6, 0x70743444,
0xe0ec2ccc, 0x91941585, 0x03080b0b, 0x53541747, 0x505c1c4c, 0x53581b4b,
0xb1bc3d8d, 0x01000101,
0x20242404, 0x101c1c0c, 0x73703343, 0x90981888, 0x10101000, 0xc0cc0ccc,
0xf2f032c2, 0xd1d819c9,
0x202c2c0c, 0xe3e427c7, 0x72703242, 0x83800383, 0x93981b8b, 0xd1d011c1,
0x82840686, 0xc1c809c9,
0x60602040, 0x50501040, 0xa3a02383, 0xe3e82bcb, 0x010c0d0d, 0xb2b43686,
0x929c1e8e, 0x434c0f4f,
0xb3b43787, 0x52581a4a, 0xc2c406c6, 0x70783848, 0xa2a42686, 0x12101202,
0xa3ac2f8f, 0xd1d415c5,
0x61602141, 0xc3c003c3, 0xb0b43484, 0x41400141, 0x52501242, 0x717c3d4d,
0x818c0d8d, 0x00080808,
0x131c1f0f, 0x91981989, 0x00000000, 0x11181909, 0x00040404, 0x53501343,
0xf3f437c7, 0xe1e021c1,
0xf1fc3dcd, 0x72743646, 0x232c2f0f, 0x23242707, 0xb0b03080, 0x83880b8b,
0x020c0e0e, 0xa3a82b8b,
0xa2a02282, 0x626c2e4e, 0x93901383, 0x414c0d4d, 0x61682949, 0x707c3c4c,
```

```

0x01080909, 0x02080a0a,
0xb3bc3f8f, 0xe3ec2fcf, 0xf3f033c3, 0xc1c405c5, 0x83840787, 0x10141404,
0xf2fc3ece, 0x60642444,
0xd2dc1ece, 0x222c2e0e, 0x43480b4b, 0x12181a0a, 0x02040606, 0x21202101,
0x63682b4b, 0x62642646,
0x02000202, 0xf1f435c5, 0x92901282, 0x82880a8a, 0x000c0c0c, 0xb3b03383,
0x727c3e4e, 0xd0d010c0,
0x72783a4a, 0x43440747, 0x92941686, 0xe1e425c5, 0x22242606, 0x80800080,
0xa1ac2d8d, 0xd3dc1fcf,
0xa1a02181, 0x30303000, 0x33343707, 0xa2ac2e8e, 0x32343606, 0x11141505,
0x22202202, 0x30383808,
0xf0f434c4, 0xa3a42787, 0x41440545, 0x404c0c4c, 0x81800181, 0xe1e829c9,
0x80840484, 0x93941787,
0x31343505, 0xc3c80bcb, 0xc2cc0ece, 0x303c3c0c, 0x71703141, 0x11101101,
0xc3c407c7, 0x81880989,
0x71743545, 0xf3f83bcb, 0xd2d81aca, 0xf0f838c8, 0x90941484, 0x51581949,
0x82800282, 0xc0c404c4,
0xf3fc3fcf, 0x41480949, 0x31383909, 0x63642747, 0xc0c000c0, 0xc3cc0fcf,
0xd3d417c7, 0xb0b83888,
0x030c0f0f, 0x828c0e8e, 0x42400242, 0x23202303, 0x91901181, 0x606c2c4c,
0xd3d81bcb, 0xa0a42484,
0x30343404, 0xf1f031c1, 0x40480848, 0xc2c002c2, 0x636c2f4f, 0x313c3d0d,
0x212c2d0d, 0x40400040,
0xb2bc3e8e, 0x323c3e0e, 0xb0bc3c8c, 0xc1c001c1, 0xa2a82a8a, 0xb2b83a8a,
0x424c0e4e, 0x51541545,
0x33383b0b, 0xd0dc1ccc, 0x60682848, 0x737c3f4f, 0x909c1c8c, 0xd0d818c8,
0x42480a4a, 0x52541646,
0x73743747, 0xa0a02080, 0xe1ec2dcd, 0x42440646, 0xb1b43585, 0x23282b0b,
0x61642545, 0xf2f83aca,
0xe3e023c3, 0xb1b83989, 0xb1b03181, 0x939c1f8f, 0x525c1e4e, 0xf1f839c9,
0xe2e426c6, 0xb2b03282,
0x31303101, 0xe2e82aca, 0x616c2d4d, 0x535c1f4f, 0xe0e424c4, 0xf0f030c0,
0xc1cc0dcd, 0x80880888,
0x12141606, 0x32383a0a, 0x50581848, 0xd0d414c4, 0x62602242, 0x21282909,
0x03040707, 0x33303303,
0xe0e828c8, 0x13181b0b, 0x01040505, 0x71783949, 0x90901080, 0x62682a4a,
0x22282a0a, 0x92981a8a
};
static unsigned long int SS3[256] = {
0x08303838, 0xc8e0e828, 0x0d212c2d, 0x86a2a426, 0xcfc3cc0f, 0xcd2dc1e,

```

0x83b3b033,	0x88b0b838,				
0x8fa3ac2f,	0x40606020,	0x45515415,	0xc7c3c407,	0x44404404,	0x4f636c2f,
0x4b63682b,	0x4b53581b,				
0xc3c3c003,	0x42626022,	0x03333033,	0x85b1b435,	0x09212829,	0x80a0a020,
0xc2e2e022,	0x87a3a427,				
0xc3d3d013,	0x81919011,	0x01111011,	0x06020406,	0x0c101c1c,	0x8cb0bc3c,
0x06323436,	0x4b43480b,				
0xcfe3ec2f,	0x88808808,	0x4c606c2c,	0x88a0a828,	0x07131417,	0xc4c0c404,
0x06121416,	0xc4f0f434,				
0xc2c2c002,	0x45414405,	0xc1e1e021,	0xc6d2d416,	0x0f333c3f,	0xd313c3d,
0x8e828c0e,	0x88909818,				
0x08202828,	0x4e424c0e,	0xc6f2f436,	0x0e323c3e,	0x85a1a425,	0xc9f1f839,
0xd010c0d,	0xcfd3dc1f,				
0xc8d0d818,	0x0b23282b,	0x46626426,	0x4a72783a,	0x07232427,	0xf232c2f,
0xc1f1f031,	0x42727032,				
0x42424002,	0xc4d0d414,	0x41414001,	0xc0c0c000,	0x43737033,	0x47636427,
0x8ca0ac2c,	0x8b83880b,				
0xc7f3f437,	0x8da1ac2d,	0x80808000,	0xf131c1f,	0xcac2c80a,	0xc202c2c,
0x8aa2a82a,	0x04303434,				
0xc2d2d012,	0x0b03080b,	0xcee2ec2e,	0xc9e1e829,	0x4d515c1d,	0x84909414,
0x08101818,	0xc8f0f838,				
0x47535417,	0x8ea2ac2e,	0x08000808,	0xc5c1c405,	0x03131013,	0xcdc1cc0d,
0x86828406,	0x89b1b839,				
0xcff3fc3f,	0x4d717c3d,	0xc1c1c001,	0x01313031,	0xc5f1f435,	0x8a82880a,
0x4a62682a,	0x81b1b031,				
0xc1d1d011,	0x00202020,	0xc7d3d417,	0x02020002,	0x02222022,	0x04000404,
0x48606828,	0x41717031,				
0x07030407,	0xcbd3d81b,	0x8d919c1d,	0x89919819,	0x41616021,	0x8eb2bc3e,
0xc6e2e426,	0x49515819,				
0xcdd1dc1d,	0x41515011,	0x80909010,	0xccd0dc1c,	0x8a92981a,	0x83a3a023,
0x8ba3a82b,	0xc0d0d010,				
0x81818001,	0xf030c0f,	0x47434407,	0x0a12181a,	0xc3e3e023,	0xcce0ec2c,
0x8d818c0d,	0x8fb3bc3f,				
0x86929416,	0x4b73783b,	0x4c505c1c,	0x82a2a022,	0x81a1a021,	0x43636023,
0x03232023,	0x4d414c0d,				
0xc8c0c808,	0x8e929c1e,	0x8c909c1c,	0x0a32383a,	0xc000c0c,	0xe222c2e,
0x8ab2b83a,	0x4e626c2e,				
0x8f939c1f,	0x4a52581a,	0xc2f2f032,	0x82929012,	0xc3f3f033,	0x49414809,
0x48707838,	0xccc0cc0c,				
0x05111415,	0xcbf3f83b,	0x40707030,	0x45717435,	0x4f737c3f,	0x05313435,

```

0x00101010, 0x03030003,
0x44606424, 0x4d616c2d, 0xc6c2c406, 0x44707434, 0xc5d1d415, 0x84b0b434,
0xcae2e82a, 0x09010809,
0x46727436, 0x09111819, 0xcef2fc3e, 0x40404000, 0x02121012, 0xc0e0e020,
0x8db1bc3d, 0x05010405,
0xcaf2f83a, 0x01010001, 0xc0f0f030, 0x0a22282a, 0x4e525c1e, 0x89a1a829,
0x46525416, 0x43434003,
0x85818405, 0x04101414, 0x89818809, 0x8b93981b, 0x80b0b030, 0xc5e1e425,
0x48404808, 0x49717839,
0x87939417, 0xccf0fc3c, 0x0e121c1e, 0x82828002, 0x01212021, 0x8c808c0c,
0x0b13181b, 0x4f535c1f,
0x47737437, 0x44505414, 0x82b2b032, 0x0d111c1d, 0x05212425, 0x4f434c0f,
0x00000000, 0x46424406,
0xcde1ec2d, 0x48505818, 0x42525012, 0xcbe3e82b, 0x4e727c3e, 0xcad2d81a,
0xc9c1c809, 0xcdflfc3d,
0x00303030, 0x85919415, 0x45616425, 0x0c303c3c, 0x86b2b436, 0xc4e0e424,
0x8bb3b83b, 0x4c707c3c,
0x0e020c0e, 0x40505010, 0x09313839, 0x06222426, 0x02323032, 0x84808404,
0x49616829, 0x83939013,
0x07333437, 0xc7e3e427, 0x04202424, 0x84a0a424, 0xcbc3c80b, 0x43535013,
0x0a02080a, 0x87838407,
0xc9d1d819, 0x4c404c0c, 0x83838003, 0x8f838c0f, 0xcec2cc0e, 0x0b33383b,
0x4a42480a, 0x87b3b437
};

/***** Macros for Encryption and Decryption *****/

#define GetB0(A) ( (unsigned char)((A) ) )
#define GetB1(A) ( (unsigned char)((A)>> 8) )
#define GetB2(A) ( (unsigned char)((A)>>16) )
#define GetB3(A) ( (unsigned char)((A)>>24) )
// Round function F and adding output of F to L.
// L0, L1 : left input values at each round
// R0, R1 : right input values at each round
// K : round keys at each round
#define SEED_KeySched(L0, L1, R0, R1, K) { \
    T0 = R0 ^ (K)[0]; \
    T1 = R1 ^ (K)[1]; \
    T1 ^= T0; \
    T1 = SS0[GetB0(T1)] ^ SS1[GetB1(T1)] ^ \
    SS2[GetB2(T1)] ^ SS3[GetB3(T1)]; \
}

```

```

T0 += T1; \
T0 = SS0[GetB0(T0)] ^ SS1[GetB1(T0)] ^ \
      SS2[GetB2(T0)] ^ SS3[GetB3(T0)]; \
T1 += T0; \
T1 = SS0[GetB0(T1)] ^ SS1[GetB1(T1)] ^ \
      SS2[GetB2(T1)] ^ SS3[GetB3(T1)]; \
T0 += T1; \
L0 ^= T0; L1 ^= T1; \
} //-----
// MAC주소 얻기 라이브러리
//-----
#ifdef _WIN32 // 윈도우
#include <WinSock.h>
#include <IPHlpApi.h>
#pragma comment(lib, "iphlpapi.lib")
#else // 리눅스
#include <sys/socket.h>
#include <linux/if.h>
#include <netinet/in.h>
#endif
unsigned char iotcam_macaddress[6];

/* 중간코드가 100페이지 가량 되어 생략 */

//-----
// KISA_SHA256.c (함수)
// 출처 http://seed.kisa.or.kr/ -> 알림마당 -> 자료실
//-----
static void SHA256_Transform(SHA256_ULONG_PTR Message,
SHA256_ULONG_PTR ChainVar)
{
    unsigned long a, b, c, d, e, f, g, h, T1, X[64];
    unsigned long j;
#define FF(a, b, c, d, e, f, g, h, j) {
        \
        T1 = h + Sigma1(e) + Ch(e, f, g) + SHA256_K[j] + X[j]; \
        d += T1;
        \
        h = T1 + Sigma0(a) + Maj(a, b, c);
        \
    }
#ifdef(BIG_ENDIAN)

```

```

#define GetData(x)      x
#else
#define GetData(x)      ENDIAN_REVERSE_ULONG(x)
#endif

for (j = 0; j < 16; j++)
    X[j] = GetData(Message[j]);
for (j = 16; j < 64; j++)
    X[j] = RHO1(X[j - 2]) + X[j - 7] + RHO0(X[j - 15]) + X[j - 16];
a = ChainVar[0];
b = ChainVar[1];
c = ChainVar[2];
d = ChainVar[3];
e = ChainVar[4];
f = ChainVar[5];
g = ChainVar[6];
h = ChainVar[7];
for (j = 0; j < 64; j += 8)
{
    FF(a, b, c, d, e, f, g, h, j + 0);
    FF(h, a, b, c, d, e, f, g, j + 1);
    FF(g, h, a, b, c, d, e, f, j + 2);
    FF(f, g, h, a, b, c, d, e, j + 3);
    FF(e, f, g, h, a, b, c, d, j + 4);
    FF(d, e, f, g, h, a, b, c, j + 5);
    FF(c, d, e, f, g, h, a, b, j + 6);
    FF(b, c, d, e, f, g, h, a, j + 7);
}
ChainVar[0] += a;
ChainVar[1] += b;
ChainVar[2] += c;
ChainVar[3] += d;
ChainVar[4] += e;
ChainVar[5] += f;
ChainVar[6] += g;
ChainVar[7] += h;
}static void SHA256_Init( OUT SHA256_INFO *Info )
{
    Info->uChainVar[0] = 0x6a09e667;
    Info->uChainVar[1] = 0xbb67ae85;
    Info->uChainVar[2] = 0x3c6ef372;
    Info->uChainVar[3] = 0xa54ff53a;

```

```

        Info->uChainVar[4] = 0x510e527f;
        Info->uChainVar[5] = 0x9b05688c;
        Info->uChainVar[6] = 0x1f83d9ab;
        Info->uChainVar[7] = 0x5be0cd19;
        Info->uHighLength = Info->uLowLength = 0;
}static void SHA256_Process( OUT SHA256_INFO *Info, IN const SHA256_BYTE
*pszMessage, IN unsigned int uDataLen )
{
    if ((Info->uLowLength += (uDataLen << 3)) < 0)
        Info->uHighLength++;
    Info->uHighLength += (uDataLen >> 29);
    while (uDataLen >= SHA256_DIGEST_BLOCKLEN)
    {
        memcpy((SHA256_UCHAR_PTR)Info->szBuffer, pszMessage, (signed
int)SHA256_DIGEST_BLOCKLEN);
        SHA256_Transform((SHA256_ULONG_PTR)Info->szBuffer,
(SHA256_ULONG_PTR)Info->uChainVar);
        pszMessage += SHA256_DIGEST_BLOCKLEN;
        uDataLen -= SHA256_DIGEST_BLOCKLEN;
    }
    memcpy((SHA256_UCHAR_PTR)Info->szBuffer, pszMessage, uDataLen);
}static void SHA256_Close( OUT SHA256_INFO *Info, IN SHA256_BYTE *pszDigest
)
{
    unsigned long i, Index;
    Index = (Info->uLowLength >> 3) % SHA256_DIGEST_BLOCKLEN;
    Info->szBuffer[Index++] = 0x80;
    if (Index > SHA256_DIGEST_BLOCKLEN - 8)
    {
        memset((SHA256_UCHAR_PTR)Info->szBuffer + Index, 0, (signed
int)(SHA256_DIGEST_BLOCKLEN - Index));
        SHA256_Transform((SHA256_ULONG_PTR)Info->szBuffer,
(SHA256_ULONG_PTR)Info->uChainVar);
        memset((SHA256_UCHAR_PTR)Info->szBuffer, 0, (signed
int)SHA256_DIGEST_BLOCKLEN - 8);
    }
    else
        memset((SHA256_UCHAR_PTR)Info->szBuffer + Index, 0, (signed
int)(SHA256_DIGEST_BLOCKLEN - Index - 8));
#ifdef LITTLE_ENDIAN
    Info->uLowLength = ENDIAN_REVERSE_ULONG(Info->uLowLength);
    Info->uHighLength = ENDIAN_REVERSE_ULONG(Info->uHighLength);
#endif
}

```

```

#endif

    ((SHA256_ULONG_PTR)Info->szBuffer)[SHA256_DIGEST_BLOCKLEN / 4 -
2] = Info->uHighLength;
    ((SHA256_ULONG_PTR)Info->szBuffer)[SHA256_DIGEST_BLOCKLEN / 4 -
1] = Info->uLowLength;
    SHA256_Transform((SHA256_ULONG_PTR)Info->szBuffer,
(SHA256_ULONG_PTR)Info->uChainVar);
    for (i = 0; i < SHA256_DIGEST_VALUELEN; i += 4)
        BIG_D2B((Info->uChainVar)[i / 4], &(pszDigest[i]));
}static void SHA256_Encrpyt( IN const SHA256_BYTE *pszMessage, IN unsigned
int uPlainTextLen, OUT SHA256_BYTE *pszDigest )
{
    SHA256_INFO info;
    SHA256_Init( &info );
    SHA256_Process( &info, pszMessage, uPlainTextLen );
    SHA256_Close( &info, pszDigest );
}

//-----
// KISA_SEED_ECB.c (함수)
// 출처 http://seed.kisa.or.kr/ -> 알림마당 -> 자료실
//-----
/***** Encryption *****/
static void SEED_Encrypt (
    unsigned char *pbData,          /
[in,out] data to be encrypted
    unsigned long int *pdwRoundKey) // [in]
    round keys for encryption
{
    unsigned long int L0, L1, R0, R1; // Input/output values at
each rounds
    unsigned long int T0, T1;        // Temporary
variables for round function F
    unsigned long int *K = pdwRoundKey; // Pointer of round keys
// Set up input values for first round
    L0 = ((unsigned long int *)pbData)[0];
    L1 = ((unsigned long int *)pbData)[1];
    R0 = ((unsigned long int *)pbData)[2];
    R1 = ((unsigned long int *)pbData)[3];

// Reorder for big endian
// Because SEED use little endian order in default
#ifndef BIG_ENDIAN

```

```

    L0 = EndianChange(L0);
    L1 = EndianChange(L1);
    R0 = EndianChange(R0);
    R1 = EndianChange(R1);
#endif

    SEED_KeySched(L0, L1, R0, R1, K    );    // Round 1
    SEED_KeySched(R0, R1, L0, L1, K+ 2);    // Round 2
    SEED_KeySched(L0, L1, R0, R1, K+ 4);    // Round 3
    SEED_KeySched(R0, R1, L0, L1, K+ 6);    // Round 4
    SEED_KeySched(L0, L1, R0, R1, K+ 8);    // Round 5
    SEED_KeySched(R0, R1, L0, L1, K+10);    // Round 6
    SEED_KeySched(L0, L1, R0, R1, K+12);    // Round 7
    SEED_KeySched(R0, R1, L0, L1, K+14);    // Round 8
    SEED_KeySched(L0, L1, R0, R1, K+16);    // Round 9
    SEED_KeySched(R0, R1, L0, L1, K+18);    // Round 10
    SEED_KeySched(L0, L1, R0, R1, K+20);    // Round 11
    SEED_KeySched(R0, R1, L0, L1, K+22);    // Round 12
    SEED_KeySched(L0, L1, R0, R1, K+24);    // Round 13
    SEED_KeySched(R0, R1, L0, L1, K+26);    // Round 14
    SEED_KeySched(L0, L1, R0, R1, K+28);    // Round 15
    SEED_KeySched(R0, R1, L0, L1, K+30);    // Round 16

#ifndef BIG_ENDIAN
    L0 = EndianChange(L0);
    L1 = EndianChange(L1);
    R0 = EndianChange(R0);
    R1 = EndianChange(R1);
#endif

// Copying output values from last round to pbData
((unsigned long int *)pbData)[0] = R0;
((unsigned long int *)pbData)[1] = R1;
((unsigned long int *)pbData)[2] = L0;
((unsigned long int *)pbData)[3] = L1;
}/*****
*****/
// Same as encrypt, except that round keys are applied in reverse order
static void SEED_Decrypt (
    unsigned char *pbData,                /
[in,out] data to be decrypted
    unsigned long int *pdwRoundKey)      // [in]
    round keys for decryption

```

Decryption

```

{      unsigned long int L0, L1, R0, R1;          // Input/output values at
each rounds
        unsigned long int T0, T1;                //      Temporary
variables for round function F
        unsigned long int *K = pdwRoundKey;      // Pointer of round keys
// Set up input values for first round
    L0 = ((unsigned long int *)pbData)[0];
    L1 = ((unsigned long int *)pbData)[1];
    R0 = ((unsigned long int *)pbData)[2];
    R1 = ((unsigned long int *)pbData)[3];
// Reorder for big endian
#ifdef BIG_ENDIAN
    L0 = EndianChange(L0);
    L1 = EndianChange(L1);
    R0 = EndianChange(R0);
    R1 = EndianChange(R1);
#endif
    SEED_KeySched(L0, L1, R0, R1, K+30);          // Round 1
    SEED_KeySched(R0, R1, L0, L1, K+28);          // Round 2
    SEED_KeySched(L0, L1, R0, R1, K+26);          // Round 3
    SEED_KeySched(R0, R1, L0, L1, K+24);          // Round 4
    SEED_KeySched(L0, L1, R0, R1, K+22);          // Round 5
    SEED_KeySched(R0, R1, L0, L1, K+20);          // Round 6
    SEED_KeySched(L0, L1, R0, R1, K+18);          // Round 7
    SEED_KeySched(R0, R1, L0, L1, K+16);          // Round 8
    SEED_KeySched(L0, L1, R0, R1, K+14);          // Round 9
    SEED_KeySched(R0, R1, L0, L1, K+12);          // Round 10
    SEED_KeySched(L0, L1, R0, R1, K+10);          // Round 11
    SEED_KeySched(R0, R1, L0, L1, K+ 8);          // Round 12
    SEED_KeySched(L0, L1, R0, R1, K+ 6);          // Round 13
    SEED_KeySched(R0, R1, L0, L1, K+ 4);          // Round 14
    SEED_KeySched(L0, L1, R0, R1, K+ 2);          // Round 15
    SEED_KeySched(R0, R1, L0, L1, K+ 0);          // Round 16
#ifdef BIG_ENDIAN
    L0 = EndianChange(L0);
    L1 = EndianChange(L1);
    R0 = EndianChange(R0);
    R1 = EndianChange(R1);
#endif
// Copy output values from last round to pbData

```

```

((unsigned long int *)pbData)[0] = R0;
((unsigned long int *)pbData)[1] = R1;
((unsigned long int *)pbData)[2] = L0;
((unsigned long int *)pbData)[3] = L1;
}/***** Constants for Key schedule
*****/
// KC0 = golden ratio; KCi = ROTL(KCi-1, 1)
#define KC0 0x9e3779b9UL
#define KC1 0x3c6ef373UL
#define KC2 0x78dde6e6UL
#define KC3 0xf1bbcdccUL
#define KC4 0xe3779b99UL
#define KC5 0xc6ef3733UL
#define KC6 0x8dde6e67UL
#define KC7 0x1bbcdccfUL
#define KC8 0x3779b99eUL
#define KC9 0x6ef3733cUL
#define KC10 0xddde6e67UL
#define KC11 0xbbcdccf1UL
#define KC12 0x779b99e3UL
#define KC13 0xef3733c6UL
#define KC14 0xde6e678dUL
#define KC15 0xbcdccf1bUL
/***** Macros for Key schedule
*****/
#define RoundKeyUpdate0(K, A, B, C, D, KC) { \
    T0 = A + C - KC; \
    T1 = B + KC - D; \
    (K)[0] = SS0[GetB0(T0)] ^ SS1[GetB1(T0)] ^ \
        SS2[GetB2(T0)] ^ SS3[GetB3(T0)]; \
    (K)[1] = SS0[GetB0(T1)] ^ SS1[GetB1(T1)] ^ \
        SS2[GetB2(T1)] ^ SS3[GetB3(T1)]; \
    T0 = A; \
    A = (A>>8) ^ (B<<24); \
    B = (B>>8) ^ (T0<<24); \
}#define RoundKeyUpdate1(K, A, B, C, D, KC) { \
    T0 = A + C - KC; \
    T1 = B + KC - D; \
    (K)[0] = SS0[GetB0(T0)] ^ SS1[GetB1(T0)] ^ \
        SS2[GetB2(T0)] ^ SS3[GetB3(T0)]; \

```

```

(K)[1] = SS0[GetB0(T1)] ^ SS1[GetB1(T1)] ^ \
        SS2[GetB2(T1)] ^ SS3[GetB3(T1)]; \
T0 = C; \
C = (C<<8) ^ (D>>24); \
D = (D<<8) ^ (T0>>24); \
}/****** Key Schedule *****/
static void SEED_KeySchedKey(
    unsigned long int *pdwRoundKey, // [out]
    round keys for encryption or decryption
    unsigned char *pbUserKey) // [in]
    secret user key
{
    unsigned long int A, B, C, D; // Input/output
    values at each rounds
    unsigned long int T0, T1; // / /
    Temporary variable
    unsigned long int *K = pdwRoundKey; // Pointer of
    round keys
    // Set up input values for Key Schedule
    A = ((unsigned long int *)pbUserKey)[0];
    B = ((unsigned long int *)pbUserKey)[1];
    C = ((unsigned long int *)pbUserKey)[2];
    D = ((unsigned long int *)pbUserKey)[3];

    // Reorder for big endian
    #ifndef BIG_ENDIAN
        A = EndianChange(A);
        B = EndianChange(B);
        C = EndianChange(C);
        D = EndianChange(D);
    #endif
    // i-th round keys( Ki,0 and Ki,1 ) are denoted as K[2*(i-1)] and K[2*i-1],
    respectively
    RoundKeyUpdate0(K, A, B, C, D, KC0 ); // K1,0 and K1,1
    RoundKeyUpdate1(K+ 2, A, B, C, D, KC1 ); // K2,0 and K2,1
    RoundKeyUpdate0(K+ 4, A, B, C, D, KC2 ); // K3,0 and K3,1
    RoundKeyUpdate1(K+ 6, A, B, C, D, KC3 ); // K4,0 and K4,1
    RoundKeyUpdate0(K+ 8, A, B, C, D, KC4 ); // K5,0 and K5,1
    RoundKeyUpdate1(K+10, A, B, C, D, KC5 ); // K6,0 and K6,1
    RoundKeyUpdate0(K+12, A, B, C, D, KC6 ); // K7,0 and K7,1

```

```

RoundKeyUpdate1(K+14, A, B, C, D, KC7 ); // K_8,0 and K_8,1
RoundKeyUpdate0(K+16, A, B, C, D, KC8 ); // K_9,0 and K_9,1
RoundKeyUpdate1(K+18, A, B, C, D, KC9 ); // K_10,0 and K_10,1
RoundKeyUpdate0(K+20, A, B, C, D, KC10); // K_11,0 and K_11,1
RoundKeyUpdate1(K+22, A, B, C, D, KC11); // K_12,0 and K_12,1
RoundKeyUpdate0(K+24, A, B, C, D, KC12); // K_13,0 and K_13,1
RoundKeyUpdate1(K+26, A, B, C, D, KC13); // K_14,0 and K_14,1
RoundKeyUpdate0(K+28, A, B, C, D, KC14); // K_15,0 and K_15,1
T0 = A + C - KC15;
T1 = B - D + KC15;
K[30] = SS0[GetB0(T0)] ^ SS1[GetB1(T0)] ^ // K_16,0
        SS2[GetB2(T0)] ^ SS3[GetB3(T0)];
K[31] = SS0[GetB0(T1)] ^ SS1[GetB1(T1)] ^ // K_16,1
        SS2[GetB2(T1)] ^ SS3[GetB3(T1)];
}/******
*****/
//-----
// iotcam.c
//-----
// 인수 분할 함수
static int split_params(const char *param, char **tmp)
{
    int cnt, kp;
    tmp[0] = (char*)calloc(strlen(iotcam_global_argv[0])+1, sizeof(char));
    strcpy(tmp[0], iotcam_global_argv[0]);
    for (cnt=1; ;cnt++)
    {
        kp=strcspn(param, "|");
        if (kp==0) break;
        tmp[cnt] = (char*)calloc(kp+1, sizeof(char));
        strncpy(tmp[cnt], param, kp);
        param = param+kp+1;
    }
    tmp[cnt] = NULL;

    return cnt;
}
}/* FFmpeg 함수 사용
static int call_ffmpegfunction(const char *param)
{
    char *tmp[100];
    int cnt;//, i;
    cnt = split_params(param, tmp);

```

END

```

        #ifndef IOT_VS_TESTMODE
            return ffmpeg_main(cnt, tmp);
        #endif
        return 0;
    }
    // 오버레이 스레드 실행(리눅스, 유닉스용)
    #if HAVE_PTHREADS
    static void iotcam_thread_main(void)
    {
        iotcam_save_text_json();
        pthread_exit((void *)0);
    }
    #endif
    // 설정보기 함수
    static void iotcam_view_config(void)
    {
        iotcam_print("", 1);
        if (!strcmp(iot_conf_data.in_device, "file"))
            printf("1. 파일 위치 및 이름 보기\n");
        else
            printf("1. 장치이름 보기\n");
        printf("   %s\n\n", iot_conf_data.in_name);
        printf("2. IoTCaps 웹사이트의 아이디\n");
        printf("   %s\n\n", iot_conf_data.id);
        printf("3. 해상도\n");
        printf("   가로 %u 픽셀, 세로 %u 픽셀\n\n", iot_conf_data.width,
iot_conf_data.height);
        printf("4. 비트레이트\n");
        printf("   영상 %ubps(%d KB/초), 음성 %ubps(%d KB/초)로 전송\n\n",
iot_conf_data.bitrate, iot_conf_data.bitrate/8/1024
, iot_conf_data.abitrage,
iot_conf_data.abitrage/8/1024);
        printf("5. 영상을 보려면 입력해야할 암호키\n");
        if (iot_conf_data.encryptmode == 1) printf("   %s\n", "암호키가 설정됨");
        else if (iot_conf_data.encryptmode == 2) printf("   %s\n", "암호키가 설정되
지 않음");
        iotcam_print("", 1);
    }
    // MAC주소 얻기 (윈도, 리눅스)
    static void getMacAddress(void)
    {
        int loopi;
        #ifdef _WIN32
            DWORD size = sizeof(PIP_ADAPTER_INFO);

```

```

PIP_ADAPTER_INFO Info;
ZeroMemory(&Info, size);
    if (GetAdaptersInfo(Info, &size) == ERROR_BUFFER_OVERFLOW)
    {
        Info = (PIP_ADAPTER_INFO)malloc(size);
        GetAdaptersInfo(Info, &size);
    }
    if(Info) for (loopi=0; loopi<6; loopi++) iotcam_macaddress[loopi] =
Info->Address[loopi];
#else
    struct ifreq ifr;
    struct ifconf ifc;
    char buf[1024];
    int success = 0, i;
    int sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_IP);
    if (sock == -1) { /* handle error*/ };
    ifc.ifc_len = sizeof(buf);
    ifc.ifc_buf = buf;
    if (ioctl(sock, SIOCGIFCONF, &ifc) == -1) { /* handle error */ }
    struct ifreq* it = ifc.ifc_req;
    const struct ifreq* const end = it + (ifc.ifc_len / sizeof(struct ifreq));
    for (; it != end; ++it)
    {
        strcpy(ifr.ifr_name, it->ifr_name);
        if (ioctl(sock, SIOCGIFFLAGS, &ifr) == 0)
        {
            if (! (ifr.ifr_flags & IFF_LOOPBACK))
            { // don't count loopback
                if (ioctl(sock, SIOCGIFHWADDR, &ifr) == 0)
                {
                    success = 1;
                    break;
                }
            }
        }
    }
    else { /* handle error */ }
}

const unsigned char* mac=(unsigned char*)ifr.ifr_hwaddr.sa_data;
for (loopi=0; loopi<6; loopi++) iotcam_macaddress[loopi] = mac[loopi];
#endif

```

```

} // SEED 함수 테스트
static void iotcam_seed_crypto(int encmode, int decmode)
{
    unsigned long int pdwRoundKey[32];
    unsigned char pbDataArray[128]; // HEX문자열을 BYTE로 바꿀때
    사용할 버퍼
    unsigned char pbData[2][16]; // SEED 함수로 암호화 또는 복호화할 때 사용
    하는 변수
    char *pos; // HEX -> BYTE 시 주소
    int i, j, k=-2; // 반복 변수
    // 맥주소 기반 키 삽입
    SEED_KeySchedKey(pdwRoundKey, iotcam_pbUserKey);
    // 비밀번호 SHA-256 -> 평문에 삽입
    pos = iot_conf_data.pw - 2; // 주소 삽입
    for (i=0; i<32; i++) sscanf(pos+=2, "%02x", &pbDataArray[i]); // 바이트 단위
    로 변환
    for (i=0; i<2; i++) for (j=0; j<16; j++) pbData[i][j] = pbDataArray[i*16+j]; //
    2차원 배열로 변환
    if (encmode==1)
        for (i=0; i<2; i++) SEED_Encrypt(pbData[i], pdwRoundKey); // 암호
    화
    if (decmode==1)
        for (i=0; i<2; i++) SEED_Decrypt(pbData[i], pdwRoundKey); // 복호
    화
    if (encmode==1 || decmode==1)
        for (i=0; i<2; i++) for (j=0; j<16; j++)
        sprintf(&iot_conf_data.pw[k+=2], "%02x", pbData[i][j]); // 적용
} // 환경설정 함수
static int iotcam_config(int newconf)
{
    char ch=1;
    int skipmode = 0, ffret = 0, ret, i, devnov, devnoa;
    #ifndef IOT_VS_TESTMODE
    int nowloglevel = av_log_get_level(); // 현재 로그 설정 백업
    av_log_set_level(AV_LOG_QUIET); // 로그 출력 안 함
    #endif
    // SHA256 처리용 변수
    SHA256_BYTE output_byte_text[32] = {0};
    char output_text[65] = "";
    char *addr = NULL;
    int loopi;

```

```

        if (!newconf) goto config_chk;
        iotcam_clear();
        iotcam_print("", 1);
        iotcam_print("-----", 1);
        iotcam_print("IoT Caps IOTCAM NEW Configuration", 1);
        iotcam_print("=====", 1);
        iotcam_print("", 1);
config_111:
        /* 운영체제별 장치 자동 설정 */
        #ifdef _WIN32
                while(1)
                {
                        iotcam_print("* 전송매체를 선택하세요 *", 1);
                        iotcam_print("[1] 카메라 영상 및 오디오", 1);
                        iotcam_print("[2] 저장된 파일", 1);
                        scanf("%c", &ch); fflush(stdin);
                        switch(ch)
                        {
                                case '1' :
                                        strcpy(iot_conf_data.in_device, "dshow");
// 카메라
                                        goto config_111next;
                                case '2' :
                                        strcpy(iot_conf_data.in_device, "file"); //
파일
                                        goto config_111next;
                        }
                }
        #else
                strcpy(iot_conf_data.in_device, "v4l2"); // 리눅스
        #endif
config_111next:
        iotcam_print("", 1);
        #ifdef _WIN32
                if (!strcmp(iot_conf_data.in_device, "dshow"))
                {
                        // 초기화
                        iotcam_dshow_cntv=0;
                        iotcam_dshow_cnta=0;
                        ffret

```

```

call_ffmpegfunction("-list_devices|true|-f|dshow|-i|dummy|-v|panic");
    // 비디오
    if (!iotcam_dshow_cntv) goto config_111_def;
    iotcam_print("*** dshow(DirectShow) 비디오 장치 목록
***", 1);
    for (i=0; i<iotcam_dshow_cntv; i++) printf("[%d] : %s\n",
i+1, iotcam_dshow_video[i]);
    iotcam_print("1-1. 비디오 장치를 선택하세요 : ", 1);
    scanf("%d", &devnov); fflush(stdin);
    sprintf(iot_conf_data.in_name, "video=%s",
iotcam_dshow_video[devnov-1]);
    // 오디오
    iotcam_print("*** dshow(DirectShow) 오디오 장치 목록
***", 1);
    for (i=0; i<iotcam_dshow_cnta; i++) printf("[%d] : %s\n",
i+1, iotcam_dshow_audio[i]);
    if (!iotcam_dshow_cnta)
    {
        iotcam_print("오디오 장치가 없습니다.", 1);
        iotcam_print("", 1); if (skipmode) goto config_chk;
        goto config_222;
    }
    iotcam_print("1-2. 오디오 장치를 선택하세요 : ", 1);
    scanf("%d", &devnoa); fflush(stdin);
    sprintf(iot_conf_data.in_name, "video=%s:audio=%s",
iotcam_dshow_video[devnov-1], iotcam_dshow_audio[devnoa-1]);
    iotcam_print("", 1); if (skipmode) goto config_chk;
}
else
{
config_111_def:
    if (!strcmp(iot_conf_data.in_device, "file"))
        iotcam_print("1. 파일명을 포함한 파일 위치를 입력하
세요.", 1);
    else
        iotcam_print("1. 장치 이름을 입력하세요.", 1);
#ifdef _WIN32
        scanf("%1023[^\n]s", iot_conf_data.in_name);
fflush(stdin);
        if (!strncmp((char*)iot_conf_data.in_name, "\", 1))

```

```

        {
            strncpy(iot_conf_data.in_name,
iot_conf_data.in_name + 1, strlen(iot_conf_data.in_name) - 1);

iot_conf_data.in_name[strlen(iot_conf_data.in_name) - 2] = '\0';
        }

        #else
            scanf("%s", iot_conf_data.in_name); fflush(stdin);
        #endif
        iotcam_print("", 1); if (skipmode) goto config_chk;
    }
#else
    iotcam_print("1. 장치이름을 입력하세요.", 1);
    printf("장치이름이 자동으로 입력되었습니다.\n");
    strcpy(iot_conf_data.in_name, "/dev/video0"); // 리눅스
    iotcam_print("", 1); if (skipmode) goto config_chk;
#endif
config_222:
    iotcam_print("2. IoTCaps 웹사이트의 아이디와 비밀번호를 입력하세요.", 1);
    iotcam_print("아이디 : ", 0);
    #ifdef _WIN32
        scanf("%127[^\n]s", iot_conf_data.id); fflush(stdin);
    #else
        scanf("%s", iot_conf_data.id); fflush(stdin);
    #endif
    iotcam_print("비밀번호 : ", 0);
    #ifdef _WIN32
        scanf("%127[^\n]s", iot_conf_data.pw); fflush(stdin);
    #else
        scanf("%s", iot_conf_data.pw); fflush(stdin);
    #endif

    /* SHA-256 처리 */
    S H A 2 5 6 _ E n c r p y t ( ( S H A 2 5 6 _ B Y T E * ) i o t _ c o n f _ d a t a . p w ,
strlen(iot_conf_data.pw) ,output_byte_text);
    addr = output_text;
    for (loopi=0; loopi<32; loopi++)
    {
        sprintf(addr, "%02x", (unsigned char)output_byte_text[loopi]);
        addr +=2;
    }

```

```

strcpy(iot_conf_data.pw, output_text); // 해시해서 비밀번호 저장
/* SHA-256 처리 */
    // 영상 접근용 패스워드 암호화 1차
    iotcam_seed_crypto(1, 0);
    // 영상 접근용 패스워드 암호화 2차
    { int i; for (i=0; i<64; i++) iot_conf_data.pw[i] = iot_conf_data.pw[i] ^
    iotcam_pbUserKey[i%16]; }
        iotcam_print("", 1); if (skipmode) goto config_chk;
config_333:
    iotcam_print("3. 영상의 해상도를 입력하세요(1920 1080)", 1);
    scanf("%u%u", &iot_conf_data.width, &iot_conf_data.height); fflush(stdin);
    iotcam_print("", 1); if (skipmode) goto config_chk;
config_444:
    iotcam_print("4. 영상의 비디오와 오디오 비트레이트를 입력하세요(1572864
240000)", 1);
    scanf("%u%u", &iot_conf_data.bitrate, &iot_conf_data.abitrage);
    fflush(stdin);
    iotcam_print("", 1); if (skipmode) goto config_chk;
    /* 프레임 레이트 */
    iot_conf_data.framerate = 15;
    /* 운영체제별 키프레임 자동 설정 */
    #ifdef _WIN32
        iot_conf_data.intra_period = 10;
    #else
        iot_conf_data.intra_period = 5;
    #endif
config_555:
    memset(iot_conf_data.streampw, 0x0, sizeof(iot_conf_data.streampw));
    iotcam_print("5. 영상데이터를 암호화 하시겠습니까?(예: 1, 아니오: 2)", 1);
    while(1)
    {
        scanf("%c", &ch); fflush(stdin);
        switch(ch)
        {
            case '1' : iot_conf_data.encryptmode = 1; goto
config_555_2;
            case '2' : iot_conf_data.encryptmode = 2; goto
config_555_3;
        }
    }
}

```

```

config_555_2:
    iotcam_print("암호키를 입력하세요 : ", 0);
    #ifdef _WIN32
        scanf("%15[^\n]s", iot_conf_data.streampw); fflush(stdin);
    #else
        scanf("%s", iot_conf_data.streampw); fflush(stdin);
    #endif
config_555_3:
    /* SHA-256 처리 */
    SHA256_Encrypt((SHA256_BYTE*)iot_conf_data.streampw,
strlen(iot_conf_data.streampw) ,output_byte_text);
    addr = output_text;
    for (loopi=0; loopi<32; loopi++)
    {
        sprintf(addr, "%02x", (unsigned char)output_byte_text[loopi]);
        addr +=2;
    }
    strcpy(iot_conf_data.streampw, output_text); // 해시해서 비밀번호 저장
    /* SHA-256 처리 */
    // 영상 시청용 패스워드 암호화
    { int i; for (i=0; i<64; i++) iot_conf_data.streampw[i] =
iot_conf_data.streampw[i] ^ iotcam_pbUserKey[i%16]; }
    iotcam_print("", 1); if (skipmode) goto config_chk;
config_chk:
    skipmode = 1;
    iotcam_clear();
#ifndef IOT_VS_TESTMODE
    if (iotcam_errorcode == -5)
    {
        av_log_set_level(AV_LOG_FATAL); // 로그 레벨 복원
        av_log(NULL, AV_LOG_FATAL, "\n[인증실패] 아이디 또는 비밀번호가
잘못되었습니다.\n");
        av_log_set_level(AV_LOG_QUIET); // 로그 출력 안 함
    }
    else if (iotcam_errorcode == -1)
    {
        av_log_set_level(AV_LOG_FATAL); // 로그 레벨 복원
        av_log(NULL, AV_LOG_FATAL, "\n[입출력오류] 서버가 꺼져있거나 입
력파일이 잘못되었습니다.\n");
        av_log_set_level(AV_LOG_QUIET); // 로그 출력 안 함
    }

```

```

    }
#endif

    iotcam_errorcode = 0; // 오류코드 초기화
    iotcam_print("", 1);
    iotcam_print("-----", 1);
    iotcam_print("IoTCaps IOTCAM Configuration", 1);
    iotcam_print("=====", 1);
    iotcam_print("", 1);
    iotcam_print("설정을 저장하려면 숫자 [0]번을, 수정하려면 해당 번호를 입력하
세요.", 1);
    iotcam_view_config();
    iotcam_print("번호를 입력하세요 : ", 1);
    while(1)
    {
        scanf("%c", &ch); fflush(stdin);
        if (ch == '0') break;
        switch(ch)
        {
            case '1' : iotcam_print("", 1); goto config_111;
            case '2' : iotcam_print("", 1); goto config_222;
            case '3' : iotcam_print("", 1); goto config_333;
            case '4' : iotcam_print("", 1); goto config_444;
            case '5' : iotcam_print("", 1); goto config_555;

        }
    }

    // 저장
resave:
    ret = iotcam_saveconf(CONFIG_PATH);
    if (ret)
    {
        iotcam_print("* 환경 설정파일을 저장할 수 없습니다 *", 1);
        iotcam_print("다시 시도하려면 1을 입력하세요 : ", 1);
        scanf("%c", &ch); fflush(stdin);
        if (ch == '1') goto resave;
    }

    iotcam_print("설정이 저장되었습니다!", 1);
#ifndef IOT_VS_TESTMODE
    av_log_set_level(nowloglevel); // 로그 레벨 복원
#endif

    return ffret;

```

```

}static int iotcam_video4linux2_cmd(void)
{
    char buf[1024] = "";
    system("sudo pkill uv4l");
    sprintf(buf, "sudo uv4l --driver raspicam --auto-video_nr --framerate
%u --encoding h264 --profile high --width %u --height %u --bitrate %u
--intra-period %u --text-overlay --text-filename=/tmp/iotcam.json",
        iot_conf_data.framerate,
        iot_conf_data.width,
        iot_conf_data.height,
        iot_conf_data.bitrate,
        iot_conf_data.intra_period);
    system(buf);
    system("export LD_PRELOAD=/usr/lib/uv4l/uv4lext/armv6l/libuv4lext.so");
    // 영상 접근용 패스워드 복호화 1차
    { int i; for (i=0; i<64; i++) iot_conf_data.pw[i] = iot_conf_data.pw[i] ^
iotcam_pbUserKey[i%16]; }
    // 영상 접근용 패스워드 복호화 2차
    iotcam_seed_crypto(0, 1);

                                                                    sprintf(buf,
"-f|%s|-input_format|h264|-i|%s|-f|mp4|-movflags|dash|-c:v|copy|-v|fatal|%s%s/%s|
",
        iot_conf_data.in_device,
        iot_conf_data.in_name,
        IOTCAPS_SVR,
        iot_conf_data.id,
        iot_conf_data.pw);

    // 영상 접근용 패스워드 암호화 1차
    iotcam_seed_crypto(1, 0);
    // 영상 접근용 패스워드 암호화 2차
    { int i; for (i=0; i<64; i++) iot_conf_data.pw[i] = iot_conf_data.pw[i] ^
iotcam_pbUserKey[i%16]; }
    // 영상 시청용 패스워드 복호화
    { int i; for (i=0; i<64; i++) iotcam_stream_pw[i] =
iot_conf_data.streampw[i] ^ iotcam_pbUserKey[i%16]; }
    // 영상 데이터 암호화 여부
    iotcam_encrypt_mode = iot_conf_data.encryptmode;
#ifdef HAVE_PTHREADS
    pthread_create(&threads, NULL, &iotcam_thread_main, (void *)0);
#endif
}

```

```

        iotcam_banner(); // 배너
        return call_ffmpegfunction(buf); // 호출
    }static int iotcam_default_cmd(void)
    {
        char buf[1024] = "";
        char vbuf[512] = "";
        const char *iot_drawtext =
        ",drawtext=fontsize=18:fontfile=iotcam.ttf:x=2:y=2:fontcolor=0xFFFFFFFF:shadowcolo
        r=0x000000FF:shadowx=1:shadowy=1:text=%{localtime}|";
        // 폰트파일이 없으면 iot_drawtext 글자 비움
        #ifdef _WIN32
            if (_access("iotcam.ttf", 0))
                iot_drawtext = "|";
        #else
            if (access("iotcam.ttf", 0))
                iot_drawtext = "|";
        #endif
        sprintf(vbuf, "-vf|scale=%d:%d,setsar=1/1%s", iot_conf_data.width,
        iot_conf_data.height, iot_drawtext);
        // 영상 접근용 패스워드 복호화 1차
        { int i; for (i=0; i<64; i++) iot_conf_data.pw[i] = iot_conf_data.pw[i] ^
        iotcam_pbUserKey[i%16]; }
        // 영상 접근용 패스워드 복호화 2차
        iotcam_seed_crypto(0, 1);
        if (!strcmp(iot_conf_data.in_device, "file")) // 파일 전송
            sprintf(buf,
            "-re|-rtbufsize|100000k|-i|%s|-f|mp4|-movflags|dash|-c:v|libx264|-bufsize|%u|-profi
            le:v|baseline|-preset|fast|-strict|experimental|-c:a|aac|-b:a|%u|-g|%d|-pix_fmt|yuv4
            20p|-map_metadata|-1|-map_chapters|-1|-v|error|-b:v|%u|%s%s%s/%s|",
                iot_conf_data.in_name,
                iot_conf_data.bitrate,
                iot_conf_data.abitrage,
                iot_conf_data.intra_period,
                iot_conf_data.bitrate,
                vbuf,
                IOTCAPS_SVR,
                iot_conf_data.id,
                iot_conf_data.pw);
        else // 카메라 영상 전송
            sprintf(buf,
            "-re|-rtbufsize|100000k|-f|%s|-i|%s|-f|mp4|-movflags|dash|-c:v|libx264|-bufsize|%u|

```

```

-profile:v|baseline|-preset|fast|-strict|experimental|-c:a|aac|-b:a|%u|-g|%d|-pix_fmt
|yuv420p|-v|error|-b:v|%u|-r|%u|%%s%%s/%s|",
        iot_conf_data.in_device,
        iot_conf_data.in_name,
        iot_conf_data.bitrate,
        iot_conf_data.abitrage,
        iot_conf_data.intra_period,
        iot_conf_data.bitrate,
        iot_conf_data.framerate,
        vfbuf,
        IOTCAPS_SVR,
        iot_conf_data.id,
        iot_conf_data.pw);

// 영상 접근용 패스워드 암호화 1차
iotcam_seed_crypto(1, 0);
// 영상 접근용 패스워드 암호화 2차
{ int i; for (i=0; i<64; i++) iot_conf_data.pw[i] = iot_conf_data.pw[i] ^
iotcam_pbUserKey[i%16]; }
// 영상 시청용 패스워드 복호화
{ int i; for (i=0; i<64; i++) iotcam_stream_pw[i] =
iot_conf_data.streampw[i] ^ iotcam_pbUserKey[i%16]; }
// 영상 데이터 암호화 여부
iotcam_encrypt_mode = iot_conf_data.encryptmode;
iotcam_banner(); // 배너
return call_ffmpegfunction(buf); // 호출
}int main(int argc, char *argv[])
{
    int i, ret = 0;
        // MAC주소 얻기 및 기반 키 삽입
getMacAddress();
for (i=0; i<16; i++) iotcam_pbUserKey[i] = iotcam_macaddress[i%6];
        // 초기화 코드
iotcam_global_argv = argv;
#ifdef IOT_VS_TESTMODE
        ffmpeg_exited=0;
#endif
// 설정 코드
if (iotcam_openconf(CONFIG_PATH)) iotcam_config(1); // 새로운 환경설정
파일 생성(파일 열기 실패시)
else if(argc>1 && !strcmp(argv[1], "config"))

```

```

    {
        // 수정하기
        iotcam_config(0);
        goto terminate;
    }

    // 작업시작
    if (!strcmp(iot_conf_data.in_device, "v4l2")) ||
    !strcmp(iot_conf_data.in_device, "video4linux2")) ret = iotcam_video4linux2_cmd();
    else ret = iotcam_default_cmd();
terminate:
    // 에러 -5 아이디/비번오류, 종료 -1 서버꺼져있거나 입력오류
    // printf("\n에러코드 %d, 종료코드 %d\n", iotcam_errorcode, ret);
    if (iotcam_errorcode == -5 || ret == -1)
    {
        if (ret == -1) iotcam_errorcode = -1;
        iotcam_config(0); // 수정하기
    }

    return 0;
}

```

iotcam.h

```

/**
 * Created by LEEKIWON on 2015-09-08.
 * Last modification date 2016-05-15.
 */
/*
    IOTCAPS 졸업작품
 */
//-----
// iotcam.h
//-----
// 구조체 - 수정시 iotcam_utils.o 파일 삭제후 재 컴파일
typedef struct iotcamfs
{
    unsigned int encryptmode;    // 영상 데이터 암호화 여부
    unsigned int width;          // 가로 해상도
    unsigned int height;         // 세로 해상도
    unsigned int bitrate;        // 초당 비디오 비트율
    unsigned int abitrage;       // 초당 오디오 비트율
    unsigned int framerate;      // 초당 프레임
    unsigned int intra_period;    // 키 프레임

```

```

        char in_device[64];           // 장치 포맷
        char id[128];                 // 아이디
        char pw[128];                 // 비밀번호
        char streampw[128];           // 영상 전송용 패스워드
        char in_name[512];             // 장치이름
    }iotcamfs;
// 변수
extern char *iotcam_dshow_video[32];
extern char *iotcam_dshow_audio[32];
extern int iotcam_dshow_cntv;
extern int iotcam_dshow_cnta;
extern iotcamfs iot_conf_data;
extern unsigned char iotcam_stream_pw[128]; // 영상 전송용 패스워드
extern unsigned int iotcam_encrypt_mode; // 영상 데이터 암호화 여부
// 함수
void iotcam_print(const char *str, unsigned char newline);
int iotcam_saveconf(const char *filename);
int iotcam_openconf(const char *filename);
void iotcam_clear(void);
int iotcam_save_text_json(void);
unsigned char* iotcam_aes_cbc_crypto(unsigned char *iv, unsigned char
*plainbyte, int plainsize, int *ciphersize);
void iotcam_get_buffer_size(char *hexvalue2, int size);
void iotcam_aschex_to_int(char *hexvalue2, int *size, int *csize);
void iotcam_banner(void);

```

iotcam_utils.cpp

```

/**
 * Created by LEEKIWON on 2015-09-08.
 * Last modification date 2016-05-15.
 */
//
// IOTCAPS 졸업작품
// 시간 오버레이용 text.json 파일 생성 및 cpp 함수
//
//-----
// iotcam 라이브러리
//-----
extern "C" {
#include "iotcam.h"
}

```

```

//-----
// 기본 라이브러리
//-----
#include <cstddef>
#ifndef IOT_VS_TESTMODE
#include "cryptopp/modes.h"
#include "cryptopp/aes.h"
#include "cryptopp/filters.h"
#endif
#include "config.h"
#include <fcntl.h>
#include <string>
#include <iostream>
#include <fstream>
#include <iomanip>
#include <cstdlib>
#include <stdint>
#include <iterator>
#include <ctime>
#if HAVE_STRUCT_V4L2_FRMIVALENUM_DISCRETE
#include <unistd.h>
#include <sys/ioctl.h>
#include <linux/videodev2.h>
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/json_parser.hpp>
#endif

//-----
// 기본 전역변수
//-----
using namespace std;
#if HAVE_STRUCT_V4L2_FRMIVALENUM_DISCRETE
namespace pt = boost::property_tree;
constexpr unsigned int FPS = 15;
constexpr unsigned int YV = 24;
constexpr unsigned int LPCNT = 15;
#endif

//-----

```

```

// iotcam_utils.cpp
//-----
#ifndef IOT_VS_TESTMODE
// 데이터 전송시 버퍼 크기 구하는 용도 함수
void iotcam_get_buffer_size(char *hexvalue2, int size)
{
    int li;
    char hexvalue[16] = {0,};

    hexvalue[0] = '\0'; hexvalue2[0] = '\0';
    // 10 -> 16
    snprintf(hexvalue + strlen(hexvalue), sizeof(hexvalue) - strlen(hexvalue),
"%x", size);
    // 16 -> 각자리수 HEX
    for (li=0; li<strlen(hexvalue); li++) snprintf(hexvalue2 + strlen(hexvalue2),
2, "%c", (byte)hexvalue[li]);
    // cr lf NULL 삽입
    hexvalue2[li] = '\r'; hexvalue2[li+1] = '\n'; hexvalue2[li+2] = '\0';
    return;
}

// 아스키 hex사파일 값을 십진수로
void iotcam_aschex_to_int(char *hexvalue2, int *size, int *csize)
{
    int li;
    char hexvalue[16] = {0,};
    // 각자리수 HEX -> 16
    for (li=0; li<strlen(hexvalue2); li++) snprintf(hexvalue + strlen(hexvalue),
2, "%c", (unsigned char)hexvalue2[li]);
    // 16 -> 10
    *size = strtol(hexvalue2, NULL, 16);
    *csize = (*size/16)*16;
    if (*csize < *size) *csize += 16;
    return;
}

// AES 암호화 함수
unsigned char* iotcam_aes_cbc_crypto(unsigned char *iv, unsigned char
*plainbyte, int plainsize, int *ciphersize)
{

```

```

        std::string ciphertext;

        unsigned char key[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
        try
        {
            std::string ciphertext;
            CryptoPP::CBC_Mode<CryptoPP::AES>::Encryption e;
            e.SetKeyWithIV(key, sizeof(key), iv);

            CryptoPP::StreamTransformationFilter          stf(e,          new
CryptoPP::StringSink(ciphertext));
            stf.Put((byte*)plainbyte, plainsize);
            stf.MessageEnd();

            unsigned char * writable = new unsigned char[ciphertext.size()];
            std::copy(ciphertext.begin(), ciphertext.end(), writable);
            *ciphersize = ciphertext.size();
            return writable;
        }
        catch(const CryptoPP::Exception& e)
        {
            return NULL;
        }
    }
#endif

// iotcam 설정 쓰기 함수
int iotcam_saveconf(const char *filename)
{
    ofstream fout;
    fout.open(filename, ios::binary);
    if (fout.fail()) return 1;
    fout.write((char *)&iot_conf_data,sizeof(iot_conf_data));
    fout.close();
    return 0;
}

// iotcam 설정 읽기 함수
int iotcam_openconf(const char *filename)
{

```

```

        ifstream fin;
        fin.open(filename, ios::binary);
        if (fin.fail())
        {
            cout<<"환경 설정파일을 읽어올 수 없습니다.\n";
            return 1;
        }
        fin.read((char*)&iot_conf_data, sizeof(iot_conf_data));
        fin.seekg(0, ios::end);
        fin.close();
        return 0;
    }

// iotcam 프린트 함수
void iotcam_print(const char *str, unsigned char newline)
{
    cout << str << ((newline==1) ? "\n" : "");
}

// iotcam 화면 청소 함수
void iotcam_clear(void)
{
#ifdef _WIN32
    system("cls");
#else
    system("clear");
#endif
}

// iotcam 배너 출력
void iotcam_banner(void)
{
    iotcam_clear();
    iotcam_print(" ", 1);
    iotcam_print("-----", 1);
    iotcam_print("IoTCaps IOTCAM Version 2.5", 1);
    iotcam_print("=====", 1);
    iotcam_print(" -      로그 간략히 보기", 1);
    iotcam_print(" +      로그 자세히 보기", 1);
    iotcam_print(" b      배너 표시", 1);
}

```

```

        iotcam_print(" s      설정 수정", 1);
        iotcam_print(" i      설정 보기", 1);
        iotcam_print(" q      프로그램 종료", 1);
        iotcam_print("-----", 1);
        iotcam_print(" ", 1);
    }

// text.json 파일 생성 함수
int iotcam_save_text_json(void)
{
    printf("오버레이 작업이 시작됩니다.\n");
#ifdef HAVE_STRUCT_V4L2_FRMIVALENUM_DISCRETE
    int i;
    char *dev = "/dev/video0";
    char *fname = "/tmp/iotcam.json";
    time_t currentTime;
    struct tm *localTime;
    char text[32];
    pt::ptree ptr;
    pt::ptree back[LPCNT];
    pt::ptree timept;

    for (i=0; i<LPCNT; i++)
    {
        back[i].put("font", "italic");
        back[i].put("scale", 1.0);
        back[i].put("thickness", 2);
        back[i].put("red", 0);
        back[i].put("green", 0);
        back[i].put("blue", 0);
        back[i].put("y", YV);
        back[i].put("x", -i+1);
        back[i].put("text_line", "||||||||||||||||||||||||||||||||");
    }

    timept.put("font", "italic");
    timept.put("scale", 0.8);
    timept.put("thickness", 1);
    timept.put("red", 255);
    timept.put("green", 255);

```

```

timept.put("blue", 255);
timept.put("y", YV);
timept.put("x", 2);

for (i=0; i<LPCNT; i++) ptr.push_back(std::make_pair("", back[i]));

auto fd = open(dev, O_RDWR | O_NONBLOCK);
if (fd == -1)
{
    printf("열기 실패\n");
    return EXIT_FAILURE;
}

struct v4l2_queryctrl queryctrl{};
struct v4l2_control ctrl{0, 1};
for (queryctrl.id = V4L2_CID_PRIVATE_BASE;; ++queryctrl.id)
{
    if (!ioctl(fd, VIDIOC_QUERYCTRL, &queryctrl))
    {
        const std::string name{std::begin(queryctrl.name),
std::end(queryctrl.name)};
        if (name.find("overlay") != std::string::npos)
        {
            ctrl.id = queryctrl.id;
            break;
        }
    }
    else
        break;
}

if (!ctrl.id) {
    printf("텍스트 오버레이를 지원하지 않습니다.\n");
    return EXIT_FAILURE;
}

do
{
    time(&currentTime);
    localTime = localtime(&currentTime);

```

```
        sprintf(text, "%d/%02d/%02d %02d:%02d:%02d",
localTime->tm_year+1900, localTime->tm_mon+1, localTime->tm_mday,
        localTime->tm_hour, localTime->tm_min, localTime->tm_sec);
        timept.put("text_line", text); // 시간 넣기

        ptr.push_back(std::make_pair("", timept));
        pt::write_json(fname, ptr); // 저장
        ptr.pop_back();

        if (ioctl(fd, VIDIOC_S_CTRL, &ctrl) == -1) {
            printf("장치에 알림 실패\n");
            break;
        }

        usleep((1000 / FPS) * 1000);
    } while(true);

    close(fd);
#else
    printf("윈도에서는 지원하지 않는 함수입니다.\n");
#endif
    return 0;
}
```

사물인터넷(IoT) 기반의 스트리밍 데이터 암호화

2016. 05. 24

담당 교수 : 양정모 교수님
IoTcaps

목 차

- 조원 소개
- 주제 선정 이유
- 구상도
- 추진 일정
- 개발 환경
- 프로그램 설명(IOTCAPS, IOTCAM)
- 결론 및 기대효과

조원 소개

이름	역할
조장 이기원	프로젝트 총괄, 프로그램 개발
조원 윤대영	프로그램 개발
조원 김혜영	자료 조사

주제 선정 이유



2016년 JTBC기사

- **필요성:** 어린이집 CCTV 설치가 의무화 되고, 사물인터넷의 주요 서비스인 IP CCTV 가 보급되면서 CCTV 해킹 사례가 급증하여 이를 통한 악용 가능성이 높아짐.
- **해결방안:** 스트리밍 데이터를 암호화 하여 중간에 가로채기를 하여도 안전.

<http://media.daum.net/society/others/newsview?newsid=20160120213138924>

개발 환경

- 지원하는 운영체제
 - IOTCAM(송신): Windows(x86), Linux(ARM)
 - IOTCAPS(수신): Windows, Linux, Android 등
- 개발 언어
 - IOTSERVER: node.js(Server-Side JavaScript)
 - IOTCAM: C, C++
 - IOTCAPS: HTML5, CSS, JavaScript
- 데이터베이스
 - MySQL

프로그램 설명(IOTCAPS)

- 회원가입/로그인
 - App에서 회원가입
 - App에서 인증키 로그인, Web에서 인증서 및 ID/PW 로그인
- 영상 접근제어
 - 영상 접근 요청
 - 영상 접근 수락
 - 목록삭제

프로그램 설명(IOTCAPS)

- 영상보기

- 영상보기(잘못된 암호키)
- 영상보기(올바른 암호키)
- 시간동기/녹화
- 재생정보

* Android에서는 App또는 Web(Mobile Chrome)을 통해 사용할 수 있음.

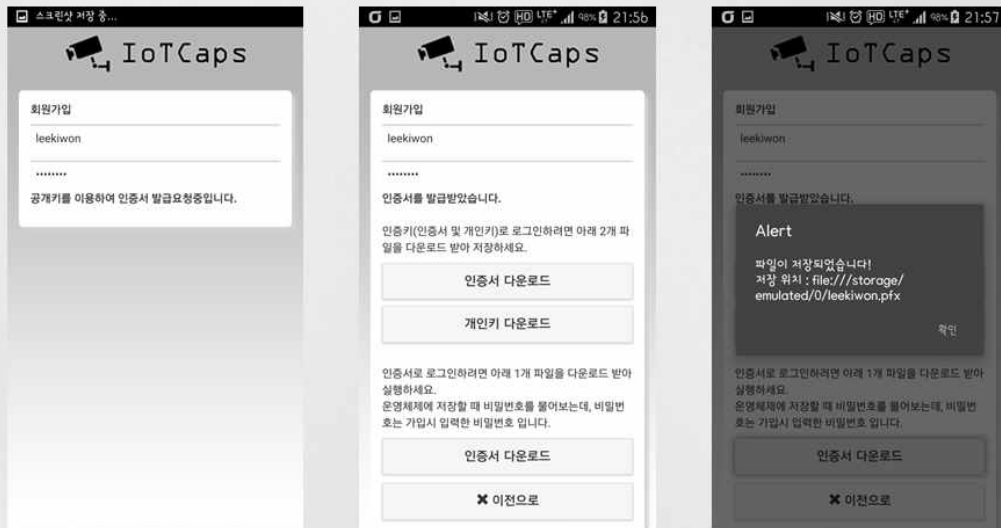
* Windows에서는 Web(Chrome 또는 Microsoft Edge) 을 통해 사용할 수 있음.

웹사이트 주소 <https://shana.pe.kr/iotcaps/>

IOTCAPS — App에서 회원가입

The screenshot displays the IoTcaps App interface. On the left is a large logo for 'IoTcaps'. In the center is a login form titled '로그인 방식' (Login Method) with fields for '아이디 및 패스워드 로그인' (ID and Password Login), 'ID', and 'Password'. Below these fields is a '로그인' (Login) button. At the bottom of the login form is a box asking 'IoTcaps를 처음 이용하시나요?' (Are you using IoTcaps for the first time?) with a '회원 가입' (Sign Up) button. On the right is a mobile phone screen showing the '회원가입' (Sign Up) form. The form has fields for '이메일' (Email) with the value 'leekiwon', a password field with a strength indicator, and buttons for '✓ 가입하기' (Sign Up) and '✕ 이전으로' (Previous). Below the phone screen is a numeric keypad and a QWERTY keyboard.

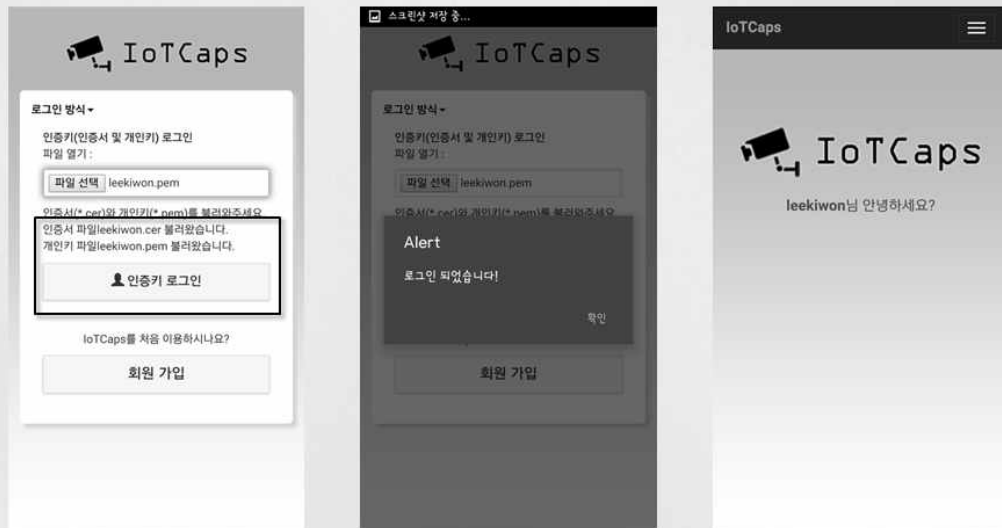
IOTCAPS — App에서 회원가입



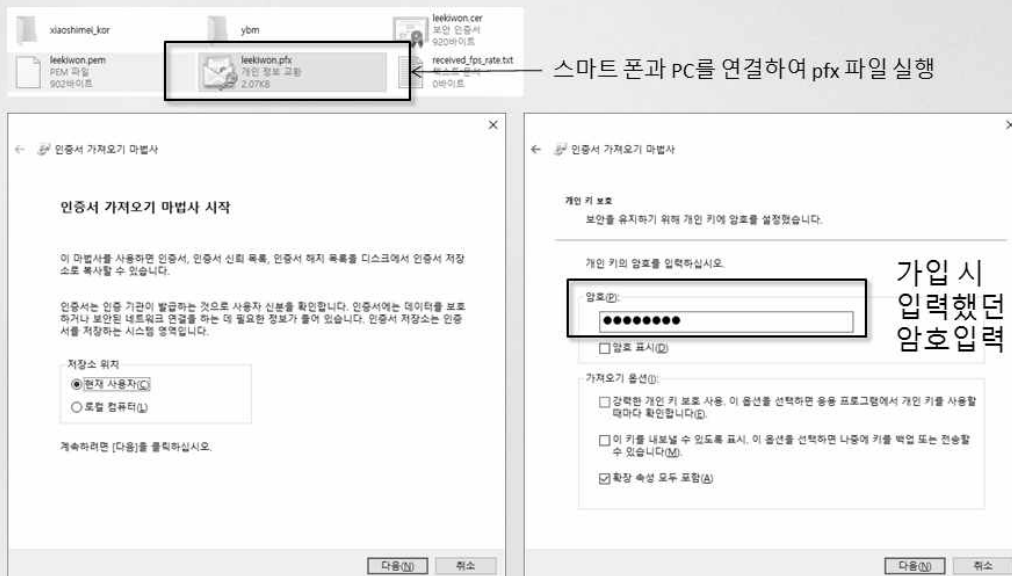
IOTCAPS — App에서 인증키 로그인



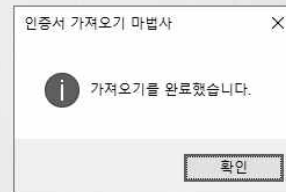
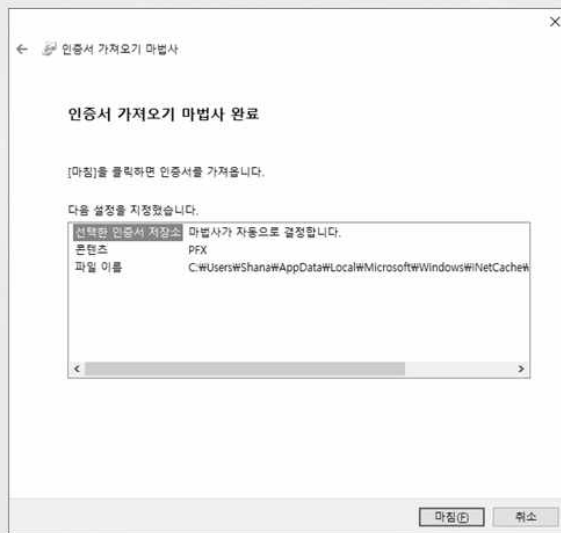
IOTCAPS — App에서 인증키 로그인



IOTCAPS — Web에서 인증서 로그인



IOTCAPS - Web에서 인증서 로그인



IOTCAPS - Web에서 인증서 로그인



IOTCAPS — Web에서 인증서 로그인

인증서 선택

인증서 선택



leekiwon

발급자: IoTCapsCA

유효 기간(시작): 2016-05-16 부터 2017-05-16

인증서 속성을 보려면 여기를 클릭하십시오.

다른 옵션 선택



shana

발급자: IoTCapsCA

유효 기간(시작): 2016-04-07 부터 2017-04-07

확인

취소

IOTCAPS — Web에서 인증서 로그인

자격 증명 필요

자격 증명 필요

앱이 개인 키에 액세스하도록 허용하시겠습니까?

키 설명 : CryptoAPI 개인 키

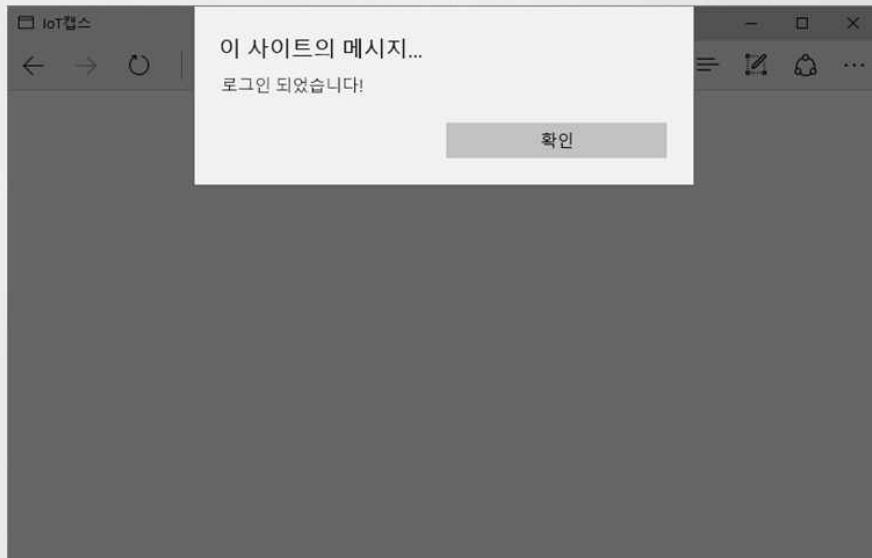


☒ 액세스 항상 허용

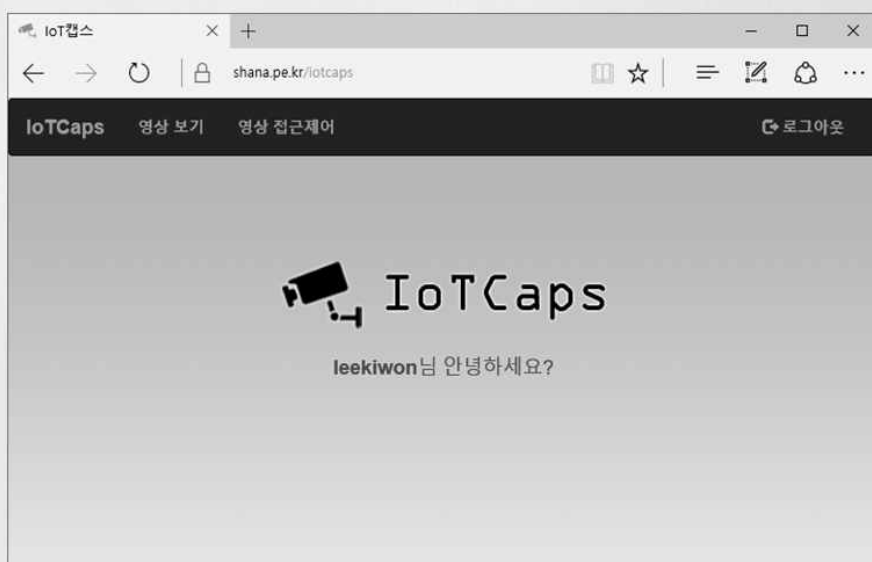
허용

허용 안 함

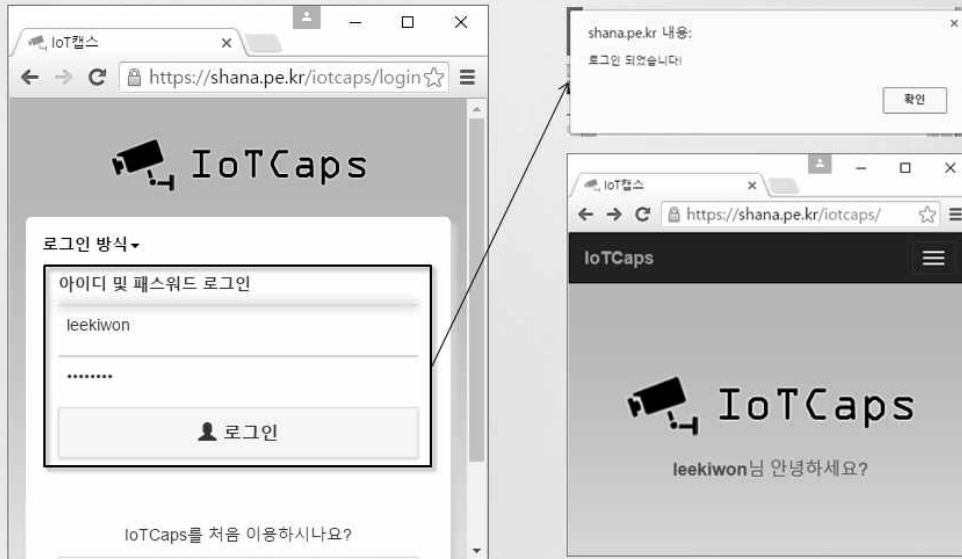
IOTCAPS — Web에서 인증서 로그인



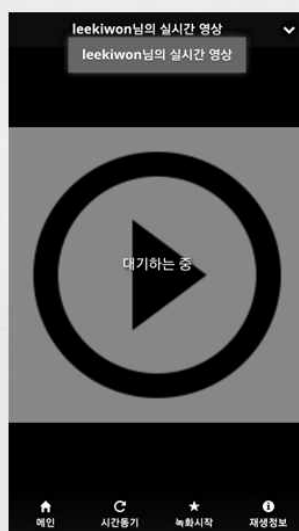
IOTCAPS — Web에서 인증서 로그인



IOTCAPS — Web에서 ID/PW로그인



IOTCAPS — 영상 접근 요청

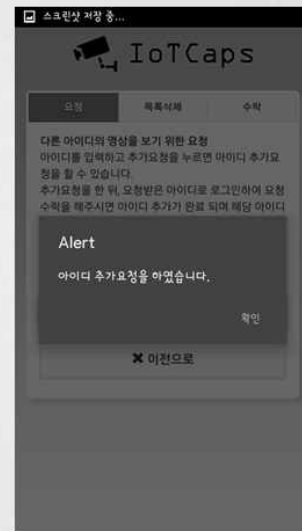
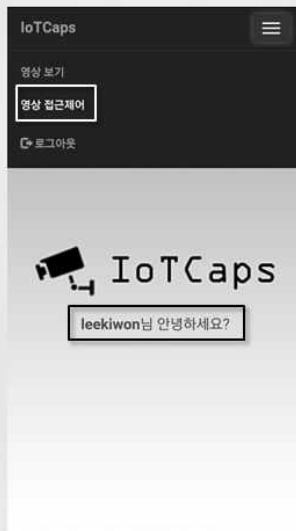


영상 접근 요청을 하는 이유 : 기밀성

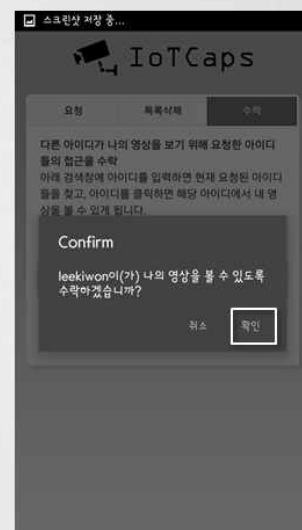
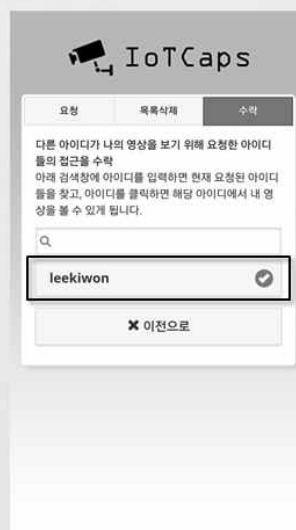
서버에서는 접근 제어 리스트를 통해 사용자가 해당 영상을 받을 권한이 있는지 확인

- 좌측 이미지는 영상 보기를 눌렀을 때 나오는 화면
- 처음 가입 후 영상 목록은 자기 자신 밖에 없음
- A라는 사용자가 B라는 사용자의 영상을 보려면 A사용자가 B사용자에게 요청을 하고 B가 수락해야 A가 B사용자의 영상을 볼 수 있음

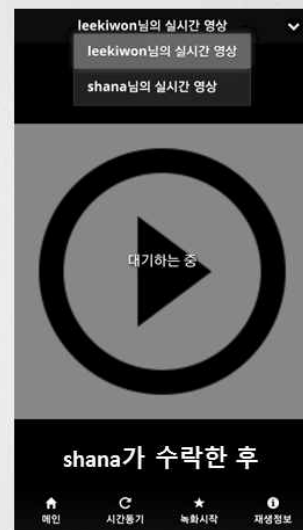
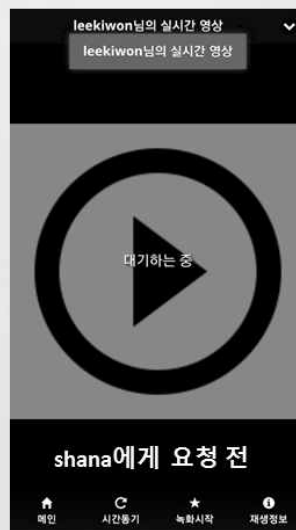
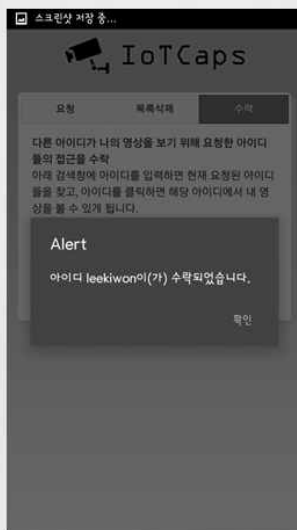
IOTCAPS — 영상 접근 요청



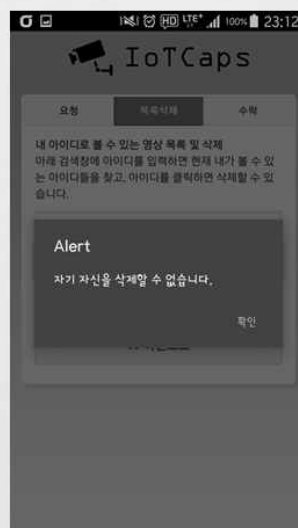
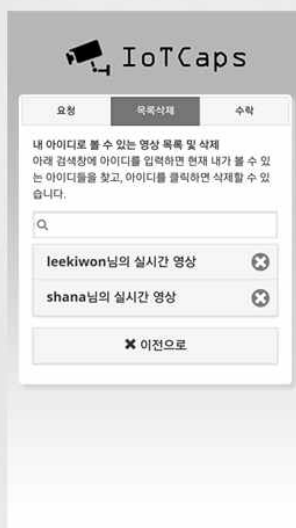
IOTCAPS — 영상 접근 수락



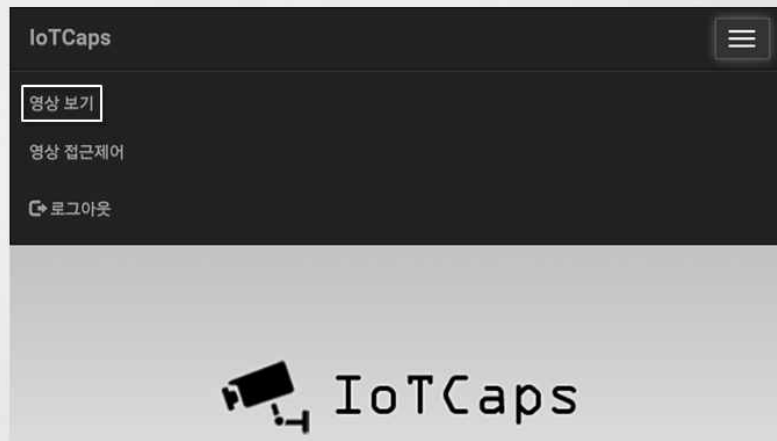
IOTCAPS — 영상 접근 수락



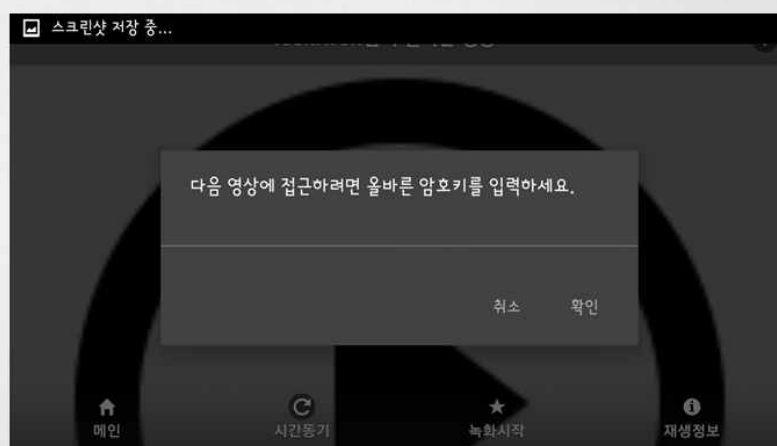
IOTCAPS — 목록삭제



IOTCAPS — 영상보기

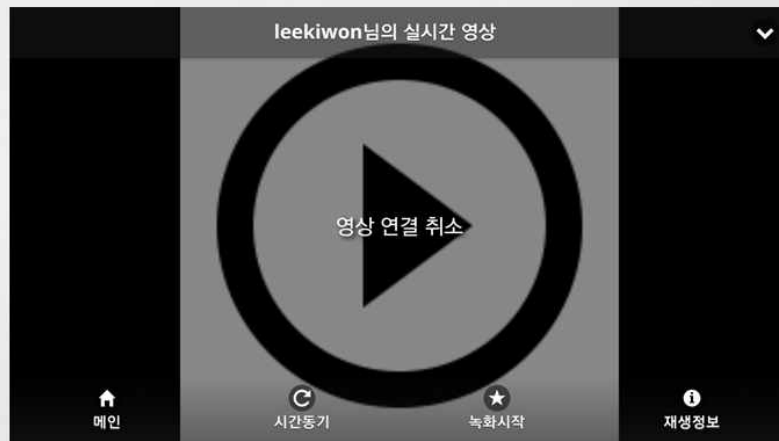


IOTCAPS — 영상보기



영상 보기를 누르면 자신의 영상을 자동으로 재생
암호키가 없으면 확인 버튼만 누르면 됨

IOTCAPS – 영상보기



암호키 입력 창에서 취소 버튼을 눌렀을 때.

IOTCAPS – 영상보기(잘못된 암호키)



- 영상을 선택하고 암호키 입력창에 잘못된 암호키 입력
- 채널에 접근 권한이 있는지 확인(인증 요청)
- 잘못된 암호키를 입력하면 바로 영상이 나오지 않고 10초 타임아웃 시작
- 10초 후에도 올바른 영상 데이터가 아니면 암호키 입력창 다시 출력

IOTCAPS - 영상보기(올바른 암호키)



server영상 선택 후 올바른 암호키 입력



채널에 접근 권한이 있는지 확인



영상에 접속하는 중



영상 출력

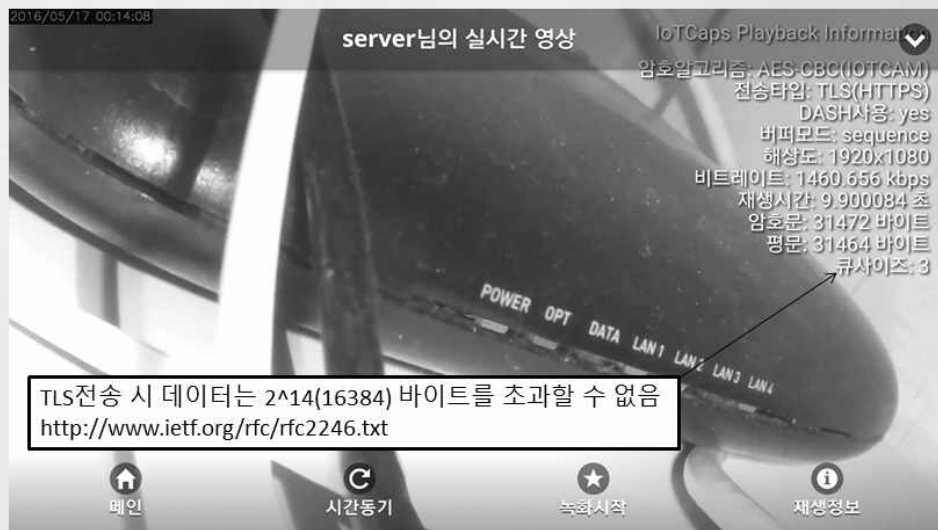
IOTCAPS - 시간동기/녹화



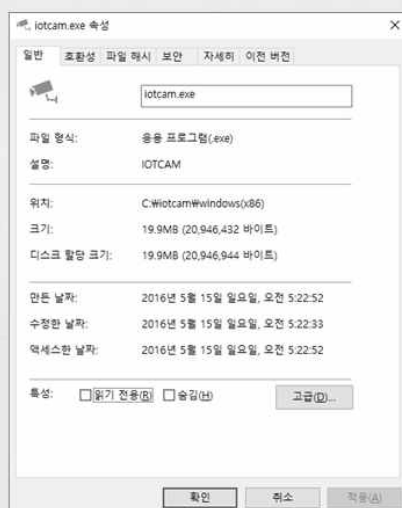
저장용량을 실시간으로 볼 수 있음

- 시간동기
영상 지연이 발생할 경우 지연을 최소화 해주는 기능
- 녹화(좌측 스크린샷)
현재 영상을 녹화해주는 기능
App에서는 1GB 단위
Web에서는 230MB 단위로 분할 저장

IOTCAPS – 재생정보



프로그램 설명(IOTCAM)



윈도용 IOTCAM(x86)



라즈베리파이
리눅스용
IOTCAM(ARM)

프로그램 설명

- 리눅스에서 영상 전송
- 윈도우에서 영상 전송
- 잘못된 ID/PW 입력시

IOTCAM — 리눅스에서 영상 전송

- iotcam 권한을 755로 설정하고 터미널에서 iotcam 을 실행
- 우측과 같이 정보를 입력하고 엔터키를 누르면서 진행
- 우측 5번에 암호키는 영상을 시청할 때 사용해야하는 암호키
- 암호키를 설정하지 않았다면 영상을 볼 때 암호키 입력 창에 암호키를 입력하지 않고 확인 버튼 클릭

```
IoTCaps IOTCAM NEW Configuration
=====
1. 장치 이름을 입력하세요.
장치 이름이 자동으로 입력되었습니다.

2. IoTCaps 웹사이트의 아이디와 비밀번호를 입력하세요.
아이디 : leekiwon
비밀번호 : 

3. 영상의 해상도를 입력하세요 (1920 1080)
1920
1080

4. 영상의 비디오와 오디오 비트레이트를 입력하세요 (1572864 240000)
1572864
240000

5. 영상데이터를 암호화 하시겠습니까? (예: 1, 아니오: 2)
1
암호키를 입력하세요 : 
```

IOTCAM — 리눅스에서 영상 전송

```
IoTCaps IOTCAM Configuration
=====
설정을 저장하려면 숫자 [0]번을, 수정하려면 해당 번호를 입력하세요.

1. 장치 이름 보기
/dev/video0

2. IoTCaps 웹사이트의 아이디
leekiwon

3. 해상도
가로 1920 픽셀, 세로 1080 픽셀

4. 비트레이트
영상 1572864bps( 192 KB/초), 음성 240000bps( 29 KB/초)로 전송

5. 영상을 보려면 입력해야할 암호키
암호키가 설정됨

번호를 입력하세요 :
```

0번을 누르고 엔터키를 누르면 영상 전송 시작

IOTCAM — 리눅스에서 영상 전송

```
IoTCaps IOTCAM Version 2.5
=====
-      로그 간략히 보기
+      로그 자세히 보기
b      배너 표시
s      설정 수정
i      설정 보기
q      프로그램 종료
=====

** IoTCaps Cipher Size = 32784 **
PLA Byte Size = 32768
00 00 00 34 73 69 64 78 01 00 00 00 00 00 01 00 0F 42 40 00 00 00 00 01 58 EA 7D 00 00
CIP Byte Size = 32784
AF F5 51 45 27 B7 2E 97 27 E6 2D 29 3F 51 9B 3A 2C 10 48 80 10 4A 88 82 EF 31 14 E9 33 3B
TLS Byte Size = 69
17 03 03 00 40 16 6B 2C 5D D8 C0 3E EA 8B 9B 42 D4 1F 2E 7A C0 03 E7 15 B4 8C 56 65 12 7A
TLS Byte Size = 16485
17 03 03 40 60 AA 51 9B 95 3A FF E2 FF 94 D7 60 28 21 BE F6 85 13 87 60 0A 4B D1 B1 80 24
TLS Byte Size = 16565
17 03 03 40 B0 20 AE 9C 39 33 B7 BC 8A 06 8E D8 77 40 BA 0C E9 E9 BD 37 05 31 40 4A EF 15
TLS Byte Size = 117
17 03 03 00 70 6D 8F FD BB 68 71 7F E0 24 5B A4 95 61 63 81 56 17 11 1B E6 DF A2 52 D4 75
TLS Byte Size = 149
17 03 03 00 90 FB 43 42 18 D6 49 91 9A 07 48 B8 21 43 CD AC 2F 3E 98 13 96 2D 7E 26 E7 C2
```

+ 키를 여러 번 누르면 위와 같이 평문, 암호문, 실제 TLS로 전송되는 바이트들을 볼 수 있음

IOTCAM — 윈도우에서 영상 전송(장치)

```
IoTCaps IOTCAM NEW Configuration

* 전송매체를 선택하세요 *
(1) 카메라 영상 및 오디오
(2) 저장된 파일

*** dshow(DirectShow) 비디오 장치 목록 ***
(1) : CyberLink Webcam Splitter
(2) : USB HD Webcam
1-1. 비디오 장치를 선택하세요 :
2

*** dshow(DirectShow) 오디오 장치 목록 ***
(1) : 마이크(Realtek High Definition Audio)
(2) : 스테레오 믹스(Realtek High Definition Audio)
1-2. 오디오 장치를 선택하세요 :
1

3. IoTCaps 웹사이트의 아이디와 비밀번호를 입력하세요.
아이디 : leekwon
비밀번호 :

4. 영상의 해상도를 입력하세요(1920 1080)
640
480

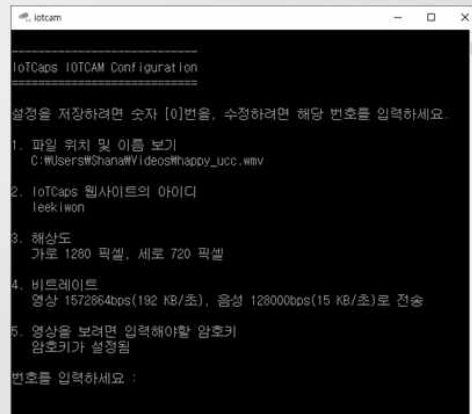
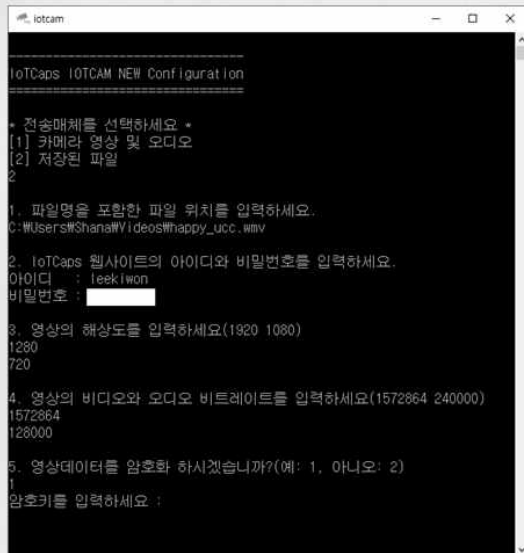
5. 영상의 비디오와 오디오 비트레이트를 입력하세요(1572864 240000)
300000
32000

6. 영상데이터를 암호화 하시겠습니까?(예: 1, 아니오: 2)
1
암호키를 입력하세요 :
```

- 윈도에서는 카메라 영상 및 오디오와 저장된 파일을 선택할 수 있음

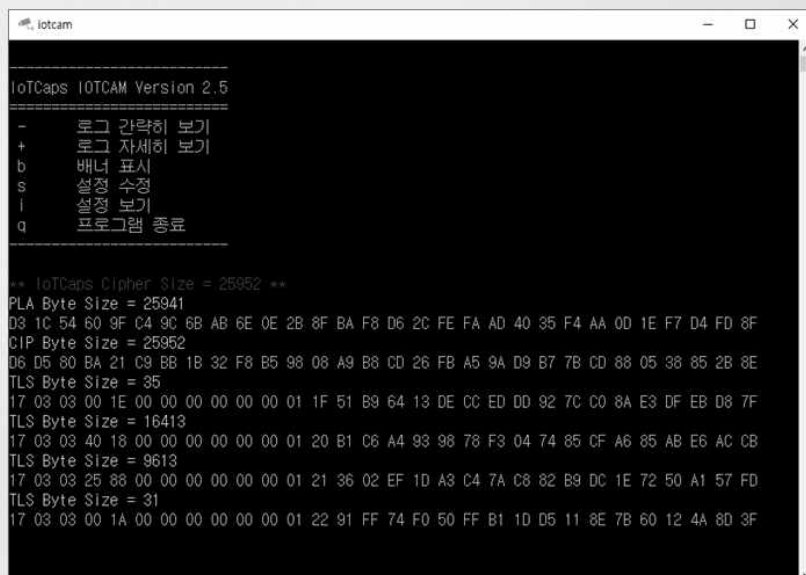
- 1번을 눌러 카메라 영상 및 오디오를 누르면 비디오 장치와 오디오 장치를 선택할 수 있음

IOTCAM — 윈도우에서 영상 전송(파일)



- 저장된 파일을 선택하면
전송하려는 동영상
파일의 경로를 입력

IOTCAM — 윈도우에서 영상 전송(파일)

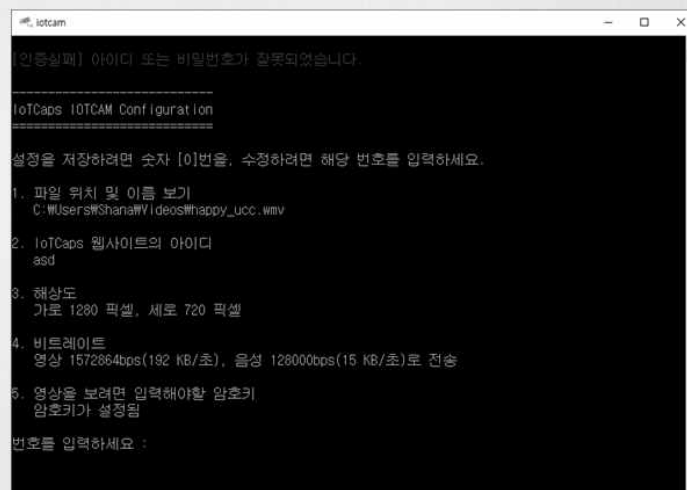


IOTCAM — 윈도우에서 영상 전송(파일)



동영상 파일은 중부대학교 홈페이지(동영상으로보는중부 -> Happy UCC 영상)
비트레이트는 동적으로 할당하므로 위와 같은 장면에서 낮게 사용
좌측 상단에 실시간 날짜 및 시간 표시

IOTCAM — 잘못된 ID/PW 입력시



잘못된 아이디 또는 비밀번호를 입력하면 영상 전송을 서버에서 거부

결론

- 스트리밍 데이터를 4단계로 보호
 - 1단계: 다른 곳에서 동일한 아이디로 로그인시 이전에 접속했던 아이디는 자동 접속 해제 및 비인가자 접근 시 차단(인증 및 세션 관리)
 - 2단계: 아무나 영상을 볼 수 없게 접근 제어(접근 제어)
 - 3단계: 암호화 알고리즘인 AES-CBC를 사용하여 실시간 스트리밍 데이터 암호화(암호화)
 - 4단계: TLS 전송(암호화)
- 서버 관리자는 서버의 실시간 상태를 로그 파일을 통해 볼 수 있음

```
[20160517_102154] ID:shana 로그인 성공! (112.161. )  
[20160517_102154] 세션데이터 변경됨: shana  
[20160517_102200] 영상 데이터 mansion에 인증 성공: id=shana  
[20160517_102200] 영상 데이터 mansion에 접근 성공: id=shana  
[20160517_111031] [데이터 수신] shana 영상을 받기 시작합니다!
```

기대효과

- 기대 효과
 - 사물인터넷(IoT)의 주요 서비스인 IP CCTV 해킹위협은 데이터 유출과 위조 및 변조, 사생활 침해 등이 있으며 이를 방지하기 위해 스트리밍 데이터 암호화가 필요
 - 차세대 IT기술인 인공지능(AI)과 IP CCTV기술이 접목된 지능형 영상감시 시스템의 관심이 높아지면서 영상 기술의 보안이 중요해지고 있으며 이를 위해 스트리밍 데이터의 암호화가 필요

지능형 영상감시 시스템은 영상을 인공지능으로 판단하여 물체 인식, 상황 인식, 교통 관제 등을 할 수 있음

A spiral-bound notebook with a light gray cover. The spiral binding is at the top. The text "Question and Answer" and "Thank You" is centered on the cover in a large, white, sans-serif font with a subtle drop shadow.

Question and Answer Thank You

Rev2. 20160517 LKW

Rev2. 20160517 LKW