보안강화 홈 *CCTV*

목차

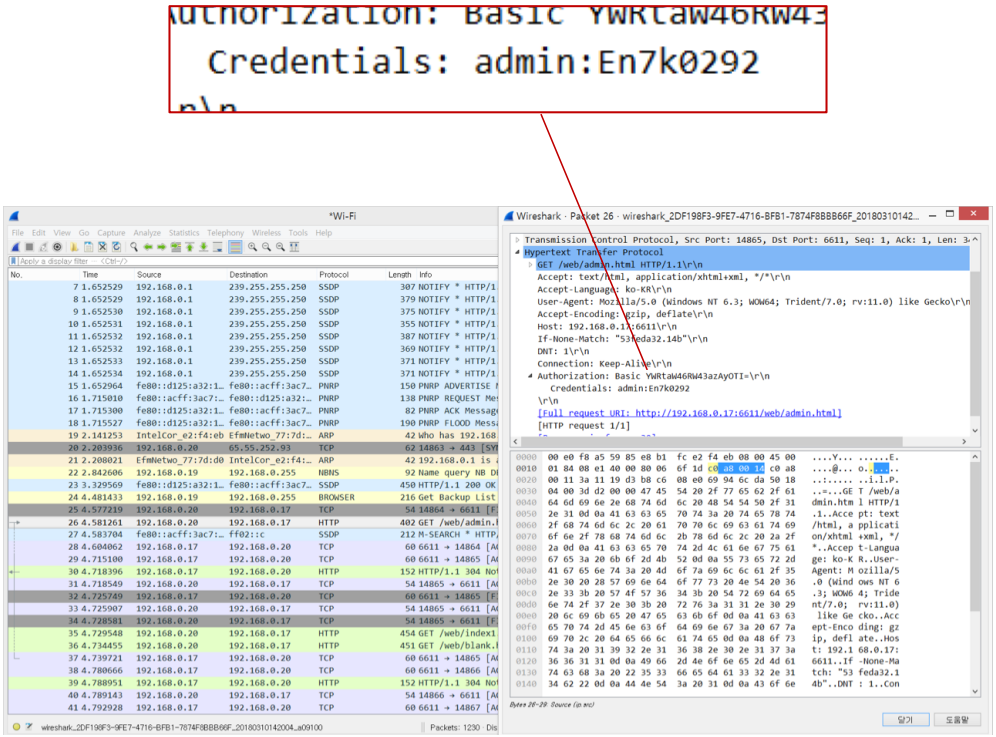
1.Project 목표

1.프로젝트 주제 선정 계기

시중에서 판매되는 가정용 홈CCTV는 실외에서도 실시간으로 집안의 상황을 볼 수 있습니다. 편리함이 중요시되는 요즘 해당 IOT 제품은 많은 인기를 끌고 있습니다.
제품의 원래 목적은 홈CCTV를 통해 혼자 있는 아이를 보호하거나 강아지를 살펴보는것과 같이 안전을 위한 용도입니다. 하지만 최근 홈CCTV를 공격해 다른 사람의 집안을 들여다 보는 등의 사건이 발생했습니다. 이로 인하여 많은 사람들이 피해를 입었고 사용자들은 결국 제품의 전원을 끄기 시작했습니다.
이후 관련 회사들은 제품에 보안 기능을 추가했다고 보도하였고, 저희는 그것이 사실인지 알아보기 위해 홈CCTV로 대표적인 회사(이지엔)의 제품을 구입해 데이터 패킷을 분석했습니다. 그 결과, 뉴스의 내용과는 달리 사용자의 ID와 패스워드가 전혀 암호화되지 않은 것을 발견했습니다.
따라서 본 프로젝트의 목적은 ID기반의 서명방식을 이용한 상호인증 및 키교환 프로토콜(IBS)을 설계해 제3의 외부공격자 뿐만 아니라 악의적인 의도를 가진 IP카메라 제조사로부터도 안전한 홈CCTV를 설계하는 것입니다. 프로젝트의 결과물은 IBS기능을 탑재한 라즈베리파이 보드와 홈CCTV를 구동시킬 안드로이드 기반의 애플리케이션, 그리고 시연을 보일 IP카메라 입니다.

2.기존 홈CCTV의 문제점

당시 대중적으로 많이 사용되었던 회사의 제품을 구입해 문제점을 파악하고자 했습니다. 네트워크 분석 프로그램인 Wireshark를 통해 패킷 덤프를 떠 본 결과, 아래 첨부화면과 같이 IP카메라의 아이디(admin)와 패스워드(En7k0292)가 암호화 되지 않은 채 패킷 헤더에 노출되는 것을 확인했습니다. 기본 제공된 패스워드를 변경하더라도 동일한 결과가 발생하는 것을 분석했습니다.



*3.*연구내용 및 방법

1) IP카메라 제품들의 동작 구조 분석
2) 3단계 인증서 체인 구조 사용시 발생할 수 있는 문제점 분석
3) ID기반의 서명방식(IBS)을 이용한 상호인증 및 키교환 프로토콜 설계
4) 라즈베리파이 보드에 IP카메라 연결
5) 회전 모듈 동작 설계
6) IP카메라 명령어 AES-GCM 암/복호화
7) 안드로이드 애플리케이션 개발
8) Gstreamer를 통한 실시간 영상 스트리밍
9) AES-ICM 방식을 사용한 실시간 영상 스트리밍 암/복호화
10) 네트워크 분석 프로그램(Wireshark)을 통한 암호화 패킷 분석, 확인

*4.*프로젝트의 목표

[외부 공격자 뿐만 아니라 호기심을 가진(honest-but-curious) 제조사로 부터 안전한 제품 제작]

처음에는 3단계 인증서체인 구조를 고려했습니다. 이 경우에는 IP카메라 제조사가 신뢰된 루트 CA(Certificate Authority)로부터 인증서를 발급받고, 자신이 중간 CA가 되어 각 IP카메라마다 인증서를 만들어 탑재시킨 후 제품을 출시합니다. 이때 제조사는 인증서를 통해 IP카메라를 특정할 수 있으며, 물론 공개키 인증서의 복호화 키를 알 수 있으므로 쉽게 암호화된 세션키를 복구할 수 있는 문제가 발생합니다.
저희는 단순히 사용자의 정보와 영상을 암호화 하는 것에만 초점을 두지 않았습니다. 위의 경우와 같이 처음 제품을 출시했을 때에는 악의적인 의도가 없었으나, 이후 특정 사용자의 사생활을 침해할 수 있는 제조사로부터도 안전한 홈CCTV를 설계하고자 했습니다

2.개발환경 및 방법론

1.개발환경

Raspberry Pi 3, Android OS Device, Servo motor, Raspberry Pi camera, Open SSL, Android studio, Gstreamer(미디어 스트리밍 오픈소스 프레임워크)

2.개발언어

Java, C

3.기타 사용 프로그램

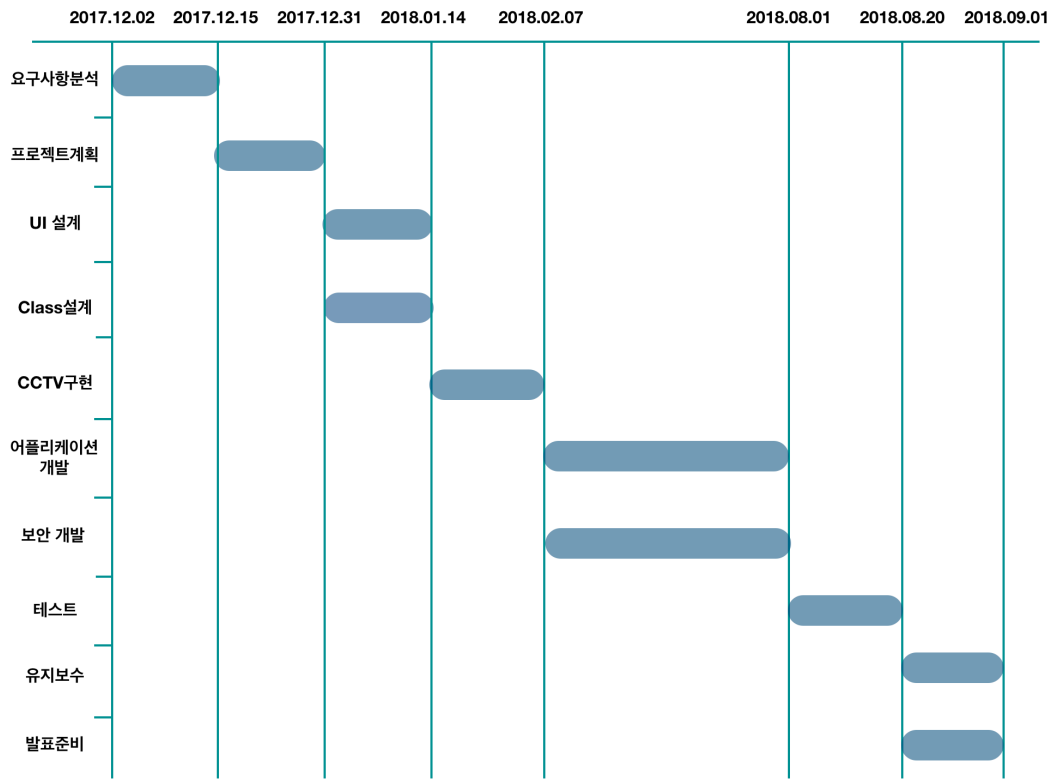
Wireshark

4.개발 방법론

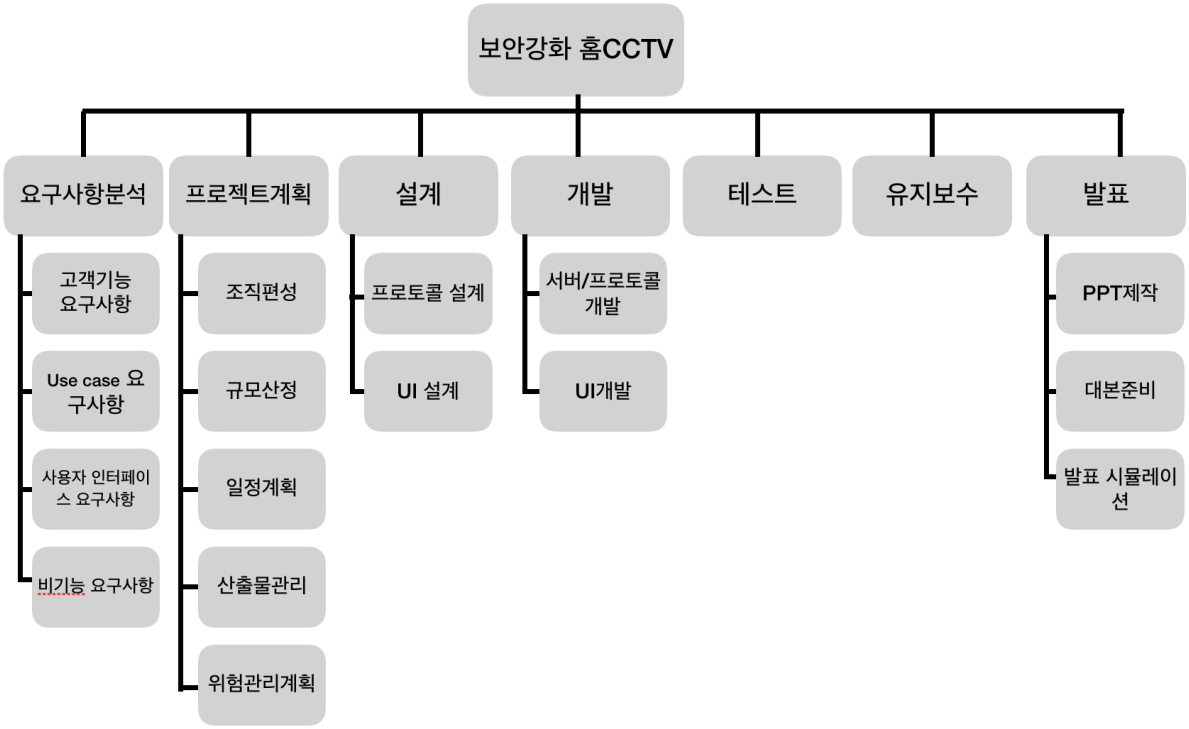
[Waterfall 모델]
프로젝트 구현에 앞서, 저희는 프로토콜 설계가 매우 중요하다 판단했습니다. 따라서 긴 기간동안 담당 교수님과 함께 안전한 프로토콜이 무엇일지 분석, 설계했습니다.

- 현재 보급된 제품의 문제점 분석
- 문제 해결을 위한 프로토콜 설계
- IP카메라 서버 / 안드로이드 어플리케이션 구현
- 제품 테스트

[Gantt chart]
프로젝트 작업 진행의 계획을 설정하고 실제 프로젝트 진행상황을 비교해 일정관리를 조정

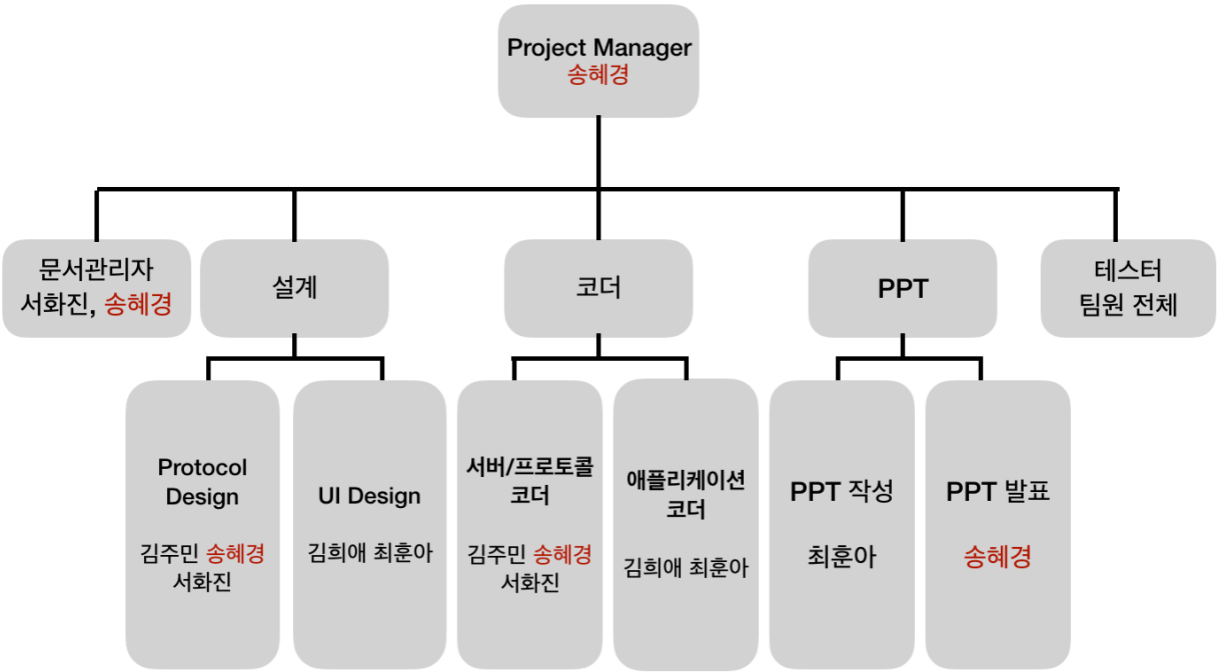


[Work-Breakdown Structure(WBS)]
모듈단위로 프로젝트를 세분화해 일정을 짜고, 역할과 일을 세분화



5.조직도

[역할 및 책임]



3. 요구사항과 제약사항

1.요구사항

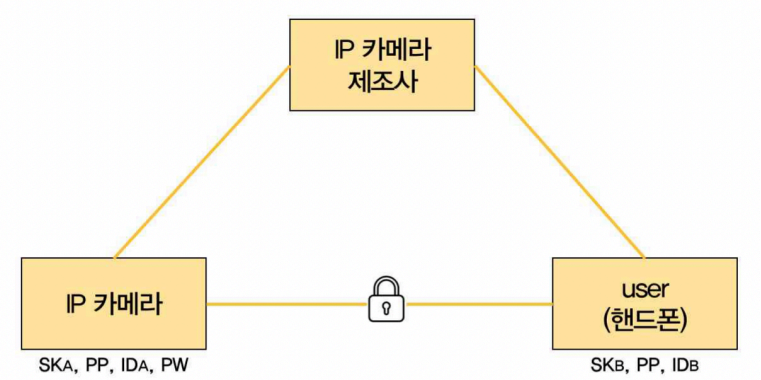
- 국제표준에 부합하는 암호화/복호화 알고리즘
- 프로토콜에 노출되지 않는 사용자의 아이디와 패스워드
- 프로그램 실행시 매번 새롭게 생성한 랜덤상수
- 랜덤상수를 기반으로 매번 새롭게 생성한 세션키
- Diffie-Hellman 방식의 키교환 프로토콜
- IP카메라로 촬영한 영상 스트리밍을 애플리케이션 화면에 보여줄 때 지연시간을 최소화(1초 미만)
- IP카메라의 상하좌우 회전
- 회전 명령어 패턴을 알아낼 수 없도록, 매번 새롭게 생성한 랜덤값으로 명령어 암호화

2.제약사항

- 제조단계에서부터 악의적인(malicious) 공격자는 고려하지 않음 - 악의적인 제조사라면, 난수 발생기를 위조하거나 시스템에 침투할 수 있는 백도어(backdoor)를 만드는 것도 가능하므로, 해당 공격까지 고려한 안정성 모델 설정이 쉽지 않아 제외했습니다.
- 영상 녹화 기능을 탑재하지 않음
- 지연시간과 반비례되는 카메라 해상도

4. 적용한 알고리즘

[IBS 기반 서명 및 키교환 프로토콜을 구성하는 네 가지 알고리즘과 현재 IP카메라 동작 관계에서의 알고리즘 적용 방법]



[IP카메라의 동작 관계도]

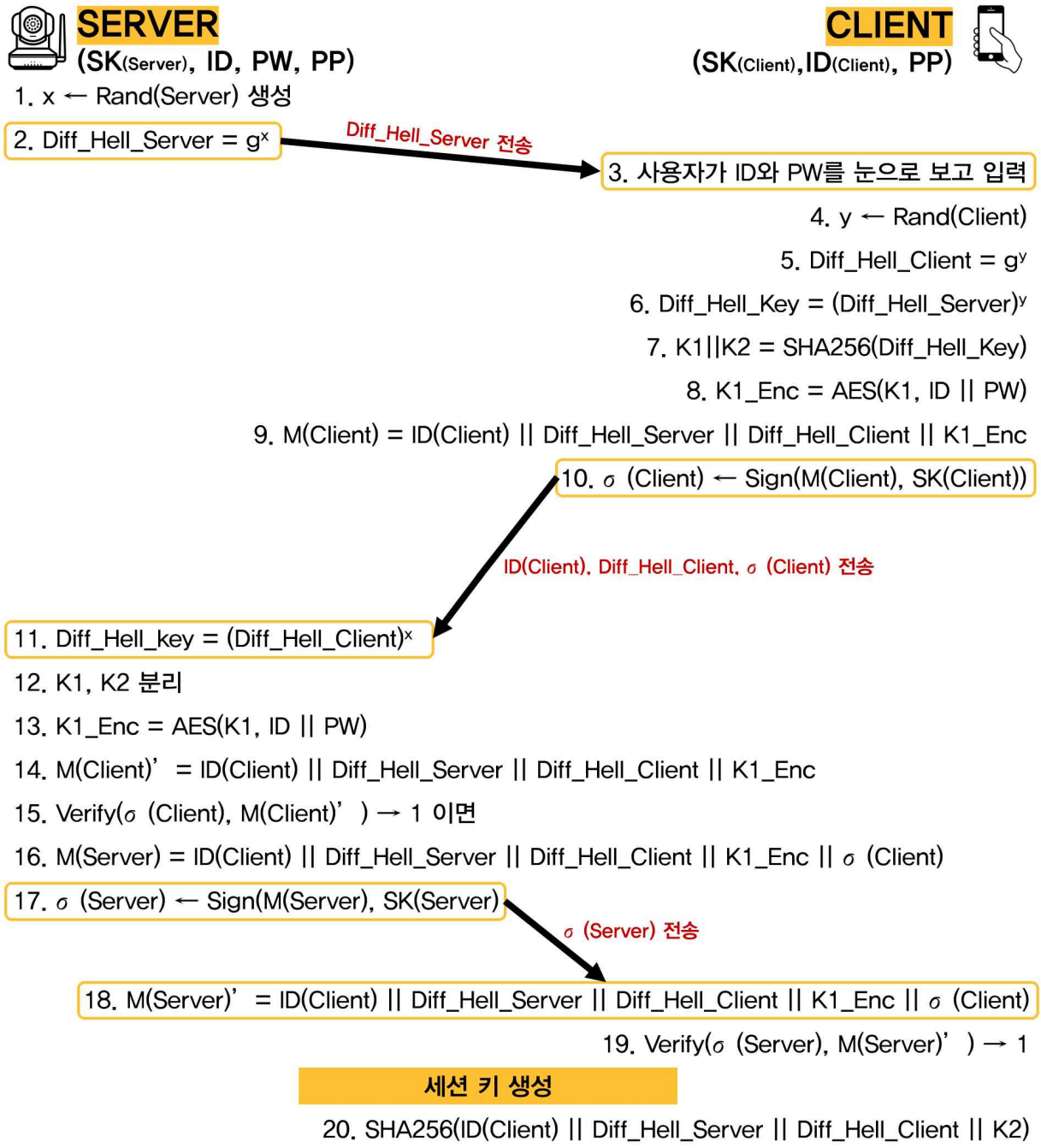
1.Setup(1^n) -> (PP, msk)
: IP카메라 제조사는 시스템 전체에 사용되는 공개파라미터(PP)와 마스터키(msk)를 출력합니다.

2.KeyGen(ID, msk) -> SK_{ID}
: IP카메라 제조사는 msk로 IP카메라의 ID와 사용자의 ID에 대응하는 서명키를 만듭니다. IP카메라의 서명키 SKserver는 제품에 탑재되어져 출시됩니다. 사용자의 서명키 SKclient는 애플리케이션을 다운로드할 때 애플리케이션 스토어에서 제공하는 보안채널을 통해 다운로드됩니다.

3.Sign(M, SK_{ID}) -> σ
: 메시지M을 서명키로 서명해 메시지서명(σ)을 출력합니다.

4.Verify(σ, M, PP) -> T/F
: 입력으로 받은 메시지서명(σ)이 메시지M에 대응하는 것이 맞는지 검증하는 알고리즘입니다. 올바른 서명이면 TRUE(1)를 출력하고 공격자에 의해 임의로 만들어진 서명이면 FALSE(0)를 출력합니다.

5. IBS기반 상호인증 및 키교환 프로토콜



[IBS기반 상호인증 및 키교환 프로토콜]

*1.*프로토콜 설명

[서버]
1) 서버가 생성한 랜덤 x를 시스템이 만들어낸 Generator g에 지수승해 g^x를 생성합니다.
2) 해당 값을 Client에게 전송합니다.

[클라이언트]
1) 사용자는 IP카메라에 붙어있는 ID, 패스워드를 눈으로 보고 입력합니다.
2) 자신의 랜덤 y를 애플리케이션 내부에 생성된 Generator g에 지수승해 g^y를 만듭니다.
3) 서버로부터 받았던 g^x에 자신의 랜덤값 y를 지수승해 Diffie-Hellman 키값인 g^xy(modulo n)을 생성합니다.
4) Diffie-Hellman Key를 SHA256 Hash function으로 256-bit의 결과물로 만듭니다. 이를 128-bit씩 둘로 쪼개어 Key1, Key2로 분리합니다.
5) IP카메라의 ID와 PW를 Key1으로 AES-GCM 암호화합니다.
6) 위의 모든 과정을 통해 나온 결과를 조합해 클라이언트의 메시지를 만듭니다.
7) 서명키 SKclient로 메시지를 서명합니다.
8) IDclient와 g^y, σclient를 서버에게 전송합니다.

[서버]
1) 전송받은 g^y를 자신의 랜덤x로 지수승해 클라이언트와 동일한 Diffie-Hellman Key를 생성합니다.
2) 위 클라이언트의 *(4), (5)* 과정을 동일하게 진행합니다.
3) 클라이언트에서 만들었던 메세지와 동일한 메세지를 형성합니다.
4) Verify 알고리즘을 통해 클라이언트로부터 전송받은 서명 σclient이 올바른지 검증합니다.
5) 올바른 서명일 경우 위의 모든 과정을 통해 나온 결과를 조합해 서버의 메시지를 만듭니다.
6) 서명키 SKserver로 메시지를 서명합니다.
7) σserver를 클라이언트에게 전송합니다.

[클라이언트]
1) 서버에서 만든 메시지와 동일한 형식의 메시지를 만들어 서버의 서명 σclient이 올바른지 검증합니다. 올바른 서명일 경우 다음 과정을 진행합니다.

[서버, 클라이언트]
1) SHA-256을 통해 얻어냈던 결과물 Key2를 포함시켜 SHA256 Hash function을 호출합니다.
2) 이렇게 만들어진 결과는 세션키(SSK)로 사용됩니다

*2.*프로토콜의 핵심

[프로토콜에 노출되지 않는 사용자의 정보]
상호인증시 IP카메라의 ID와 패스워드를 사용자가 눈으로 보고 입력합니다. 그리고 매번 새롭게 만들어낸 랜덤값으로 Diffie-Hellman키를 생성합니다. 이 키를 통해 사용자의 ID와 패스워드를 암호화합니다. 만들어진 암호문은 이전 프로토콜에서 만들어진 모든 결과와 같이 메시지로 조립되어 Sign 및 Verify됩니다. 즉, 전송되는 메시지에서 중요 정보가 노출되지 않습니다.

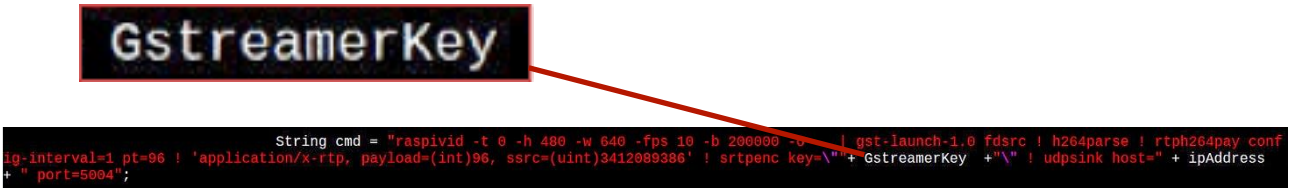
[도전-응답 방식을 통해 강화된 안정성]
서버와 클라이언트는 서로 도전 및 응답 방식을 통해 프로토콜을 진행합니다. 도전값에 대응하는 응답값이 올바르지 않을 경우 프로토콜은 종료됩니다.

[전방향 안정성 보장]
공유된 세션키는 Diffie-Hellman 방식의 키교환을 따름으로서 전방향 안정성을 보장합니다.

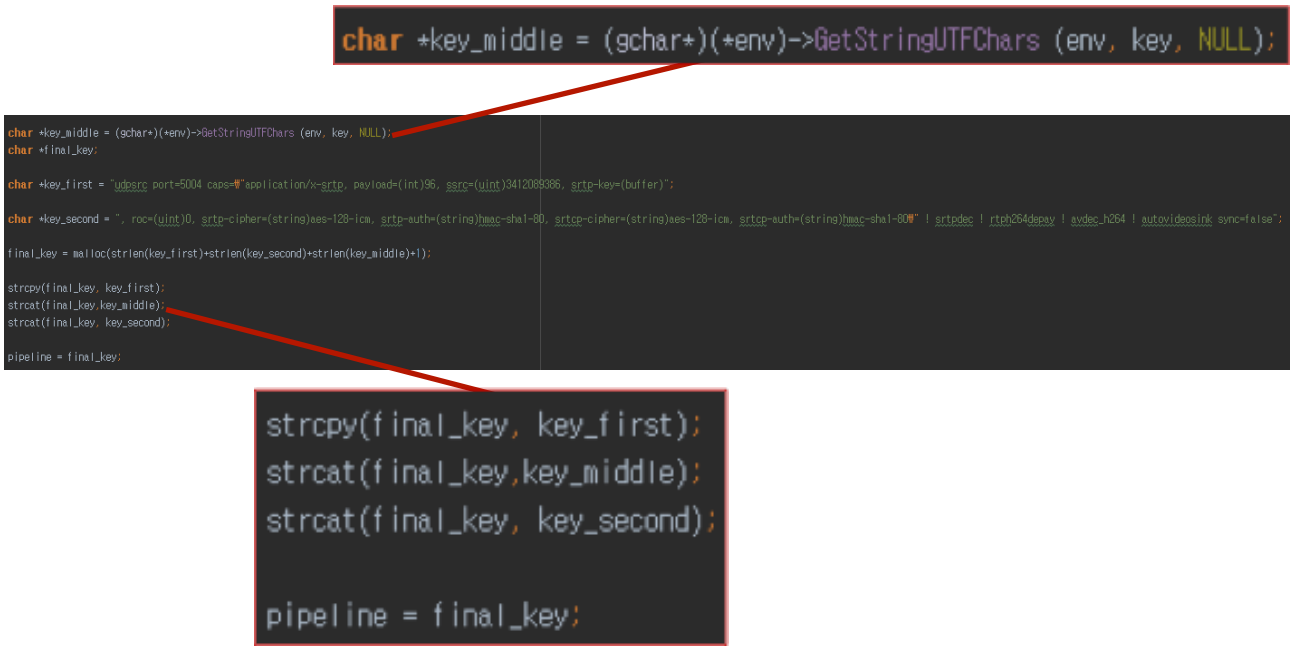
3. 세션키가 서버와 클라이언트에서 활용되는 방법

프로토콜 과정을 통해 생성된 세션키를 Gstreamer가 제공하는 명령어에 파이프라인을 통해 삽입합니다.

[서버측 파이프라인]



[클라이언트측 파이프라인]

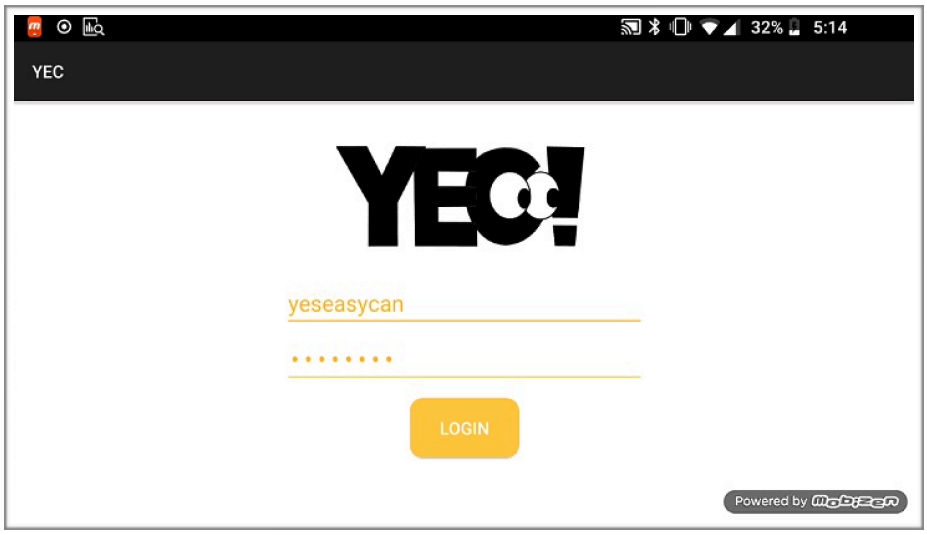


6. 프로그램 실행 모습

1.서버에서 클라이언트의 접속을 기다리는 화면

```
pi@raspberrypi:~/Project/aesIBS/aesIBS2 $ java IBSClient
Waiting for clients...
IP : 172.30.1.45
```

2.애플리케이션의 로그인 화면



3.암호화 정보들을 확인하기 위한 서버 출력 메세지

```
37DF72523930F2C959A3BD99
UPUP0000°í
ABFFBC97A2AB7A47D8C7B51D
UPUP0000 WR
ECEA72C8A754C98FDEF932B0
DOWN0000h]*Ö
E3558DA89276D823839F5FAB
DOWN0000É FP
DAFC5578C3D2180A0DBB4A93
LEFT0000ÿ 'ê
68D9EF27C1C57D415397281E
LEFT0000õP T
9C508D22DD4FA2AD0192A1C7
RIGH000048
DD6B18249E379684FD2C977E
RIGH0000 ¨
```

[회전 명령어 수행시, 매번 새롭게 암호화]

```
challen: 1068
ceiling: 1986.000000
H(A||M) length: 32

c_hashed_vrfy in veri
95CFDBAE349B967CE132B1339405851 2D07232AB0270E82894BFAC5740649B2E
7A1056B38DF3CEFC83F540A484620085 9609FA3D8F07E6CCE876F99E3A518F38
compared c
rand client: EDB9A38A3EFB1C91AE6AB81E2B95E5098B5609FB1576EE7D21B15003

============================


 test hash : 9AA4BA5FE59A64E9AC3B2289B9825531DA551923AA908BFC441580F961B51C74


============================
```

[정상 로그인시 출력되는 각종 암호문]