안면 인식 스푸핑 방지를 위한 얼굴인식과 OTP를 이용한 2 Factor 인증

윤정빈[1]*, 정지은[1]**, 조효주[1]***, 한승민[1]**, 김선희
*성신여자대학교 수학과
**성신여자대학교 융합보안공학과
***성신여자대학교 정보시스템공학과
디티이노테크
binifia@gmail.com, ss970827@naver.com, whgywn369@gmail.com,
mininyong08@gmail.com, sunny.itpe@gmail.com

2 Factor Authentication Using Face Recognition and One-Time Password for Spoofing Attack Prevention

Jung Bin Yun[1]*, Ji Eun Jung[1]**, Hyo Ju Jo[1]***, Seung Min Han[1]**, Sun Hee Kim
*Dept. of Mathematics, SungShin University
**Dept. of Convergence Security Engineering, SungShin University
***Dept. of Information Systems Engineering, SungShin University
DTInnoTech

요 약

사진, 동영상 또는 3D 프린터로 변조된 얼굴을 이용한 안면인식 보안 우회 사례가 증가하였다. 따라서 본 논문에서는 얼굴인증 단계인 본인 여부와 Liveness Detection 및 OTP를 통한 다중인증 프레임워크를 구현함으로써 기존 단일인증 대비 더욱 안전한 인증 환경을 기대한다.

1. 서론

FIDO(Fast IDentity Online)와 같은 생체 인증 기술의 발전은 사용자가 외우지 않아도 되고 비밀번호를 주기적으로 변경하거나 복잡하게 설정할 필요가 없으므로 호평을 받고 있다. 특히 안면인식은 비접촉식으로 진행되고 딥러닝과 같은 AI 기술을 활용하여 주변환경 변화에도 특화된 서비스를 제공하면서 사물인터넷(IoT), 핀테크, 헬스케어 등의 신규 서비스와 결합되어 다양한 분야에서 활용되는 추세이다[1]. 그에 따라 얼굴 인식 시장 규모는 점차 확대되어 2020년까지 국내 연평균 성장률은 11.8%, 세계 연평균 성장률은 13.3%를 기록할 것으로 전망되고 있다[2].

단순 얼굴인식의 경우, 사진, 동영상, 그리고 3D 프린터 등의 매체를 통해 변조된 얼굴을 사용하여 인증을 시도하는 Spoofing 공격 횟수가 증가함에 따라 심각한 보안 문제가 발생하게 되었다 더 이상의 피해를 방지하기 위하여 많은 이들로부터 고안된 Anti-Spoofing 기법은 실제 얼굴의 주파수와 인쇄된 이미지 또는 영상 속 얼굴의 주파수를 비교하기, 피사체의 특정 모션을 탐지하여 인쇄된 모습과 실제 얼굴을 구분하기 등이 있다. 이와 같은 효과적인 방어 기술의 효과에 대한 학술적 연구는 현재 진행 중이다[3].

이에 본 논문에서는 딥러닝 학습으로 사용자의 얼굴을 인식한 후, 타인이 승인받은 사용자 행세를 하며 속이려는 것은 아닌지를 방지하는 차원에서 실제 사용자의 얼굴(이하 real)인지 사진 혹은 영상 속 얼굴(이하 fake)인지를 구분한다. 또한, 1차 인증에서 식별된 사용자는 애플리케이션으로 구글 OTP를 발급받는 2 Factor 인증 방식의 보안 프레임워크를 제안하고자 한다. 이로부터 보안의 취약점을 보완하며 사용자의 개인 정보 유출 및 도용을 예방할 수 있을 것으로 예상된다.

2. 관련연구

2.1 얼굴인식(Face Recognition)

판독 대상 이미지에서 얼굴 영역을 국소화하고, 얼굴 영역 내에서 주요 얼굴 구조를 감지하는 얼굴의 ROI를 검출해내는 128D 얼굴 특징 벡터 임베딩 작업을 수행한다. 미리 훈련된 모델에서 OpenCV로 얼굴을 감지한 후 신경망을 위한 얼굴을 변형시켜 이

미지에서 눈과 입술이 동일한 위치에 나타나도록 OpenCV의 affine transformation이 적용된 dlib의 실시간 포즈를 추정한다[4].

2.2 Liveness Detection

주파수와 질감 기반 분석 기법은 얼굴 영역에서 로컬 이진 패턴(LBP)을 계산하고 SVM을 사용하여 얼굴을 실제 또는 Spoofing으로 분류하는 질감 기반 분석과 실제 얼굴과 가자 얼굴 사이의 높은 주파수 정보는 고주파의 불일치를 이용하는 기법이다[5]. 눈 깜빡임 탐지는 얼굴 랜드마크를 이용하여 EAR(Eye Aspect Ratio) 방정식을 도출하여 실시간 눈 깜빡임을 탐지하는 기법이다[6].

2.3 OTP(One-Time Password)

OTP는 파이어베이스에서 제공하는 구글 OTP를 활용하여 구현하였다. 구글 OTP는 시간 기반 일회용 비밀번호 알고리즘과 HMAC 기반 일회용 비밀번호 알고리즘을 사용하여 다 요소 인증 서비스를 구현하는 소프트웨어 토큰의 하나이다[7]. 본 논문에서 사용하는 구글 OTP는 6자리의 번호를 생성하여 애플리케이션에 생성번호를 보내준 뒤, 사용자는 전송받은 번호를 입력하여 서버로 전송한다. 서버는 번호의 동일 여부를 확인하고 번호가 일치하면 2차 인증이 완료되는 방식이다.

2.4 시장 및 기술 동향 분석

생체인식 보안의 활성화로 개인 사용자 입장에서 보안은 한층 강화되어 기업들의 생체인식 보안 도입이 늘고 있어 FIDO 생태계가 확장되고 있다[8].

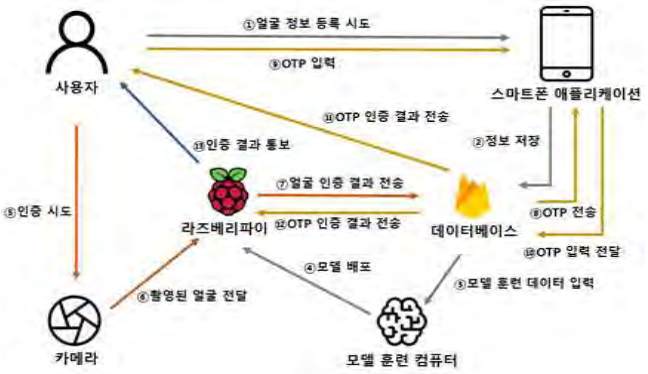
얼굴 인식 시장 규모는 점차 확대되어 2020년까지 국내 연평균 성장률은 11.8%, 세계 연평균 성장률은 13.3%를 기록할 것으로 전망된다[2].

하지만 적외선 카메라로 정면에서 촬영한 사진을 사용한 스푸핑 공격으로 마이크로 소프트의 '윈도우 헬로' 얼굴인증 시스템을 통과하였다. 또한 스키폴 국제 공항의 자동 출입국 심사대에서 가면을 쓴 사진을 카메라에 가져다 대는 것으로 자동 출입국 심사대에 인증을 시도하여 보안 허점이 있음을 보였다[9], [10].

3. 설계 및 구현

3.1 시스템 흐름도

그림 1은 본 논문의 프레임워크의 작동 흐름을 나타낸 시스템 흐름도이다.

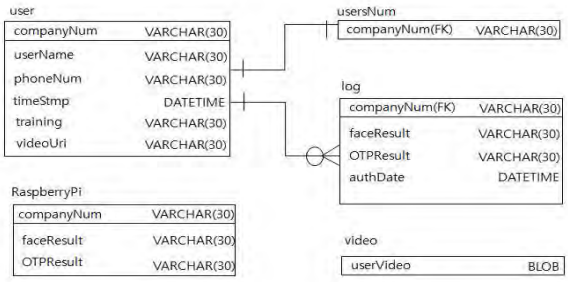


(그림 1) 시스템 흐름도

사용자는 애플리케이션에 개인정보와 본인 얼굴 동영상을 등록한다. 등록한 정보는 데이터베이스에 저장되어 얼굴인식 모델 훈련에 사용된다. 훈련된 모델은 인증 단말기로 배포된다. 사용자가 인증을 진행할 시 카메라로 얼굴을 촬영하고, 이를 인증 단말기로 전송한다. 얼굴인증을 수행한 뒤 인증 결과를 데이터베이스로 전송한다. 얼굴인증은 얼굴 인식, Liveness Detection, 눈 깜빡임 감지 기능 순으로 실행된다. OTP 인증은 얼굴인증 결과에 따라 진행한다. 사용자는 데이터베이스가 전송한 OTP 인증 번호를 애플리케이션에 입력하여 두 번째 인증 단계를 수행한다. 두 번째 인증 단계가 완료된 뒤 인증 단말기를 통해 사용자에게 최종 인증 성공 여부를 통보한다.

3.2 DB 설계

그림 2는 본 논문의 프레임워크 구현을 위해 사용한 데이터베이스의 관계를 나타낸 관계도이다.



(그림 2) DB ERD (Entity Relationship Diagram)

본 논문에서는 총 다섯 개의 테이블을 사용하고 있으며, 이 중 user, usersNum, log 세 개의 테이블이 서로 관계를 맺고 있다.

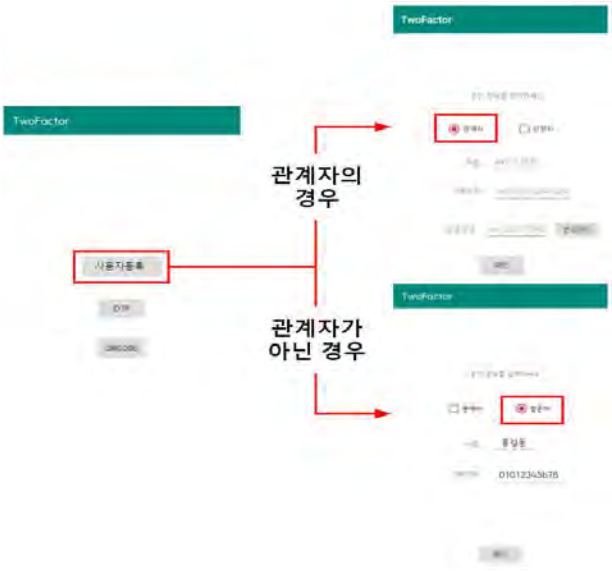
user 테이블의 구성은 모든 사용자의 성명과 개인 식별 번호, 훈련용 데이터셋 구축 여부, 타임스탬프,

video 파일 경로명을 포함하고 있다. usersNum 테이블은 사용자 중 관계자임을 확인하기 위한 관계자용 개인 식별 번호로 구성되어 있다. 일회성 혹은 단기성 사용자를 제외한 사용자는 usersNum 테이블을 통하여 본인이 관계자임을 확인하여야 다음 단계로의 진행이 가능하다.

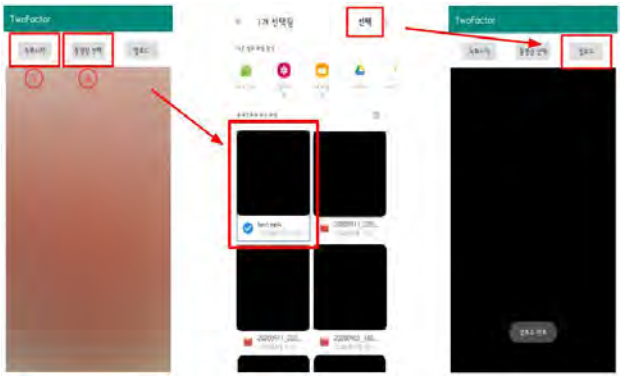
본 논문의 프레임워크는 모든 사용자의 인증과정을 구현하고, 사용자 중 관계자의 진위 여부를 구별하기 위해 사용자의 인증 성공·실패 여부와는 상관없이 모든 결괏값과 이에 해당하는 타임스탬프를 Log 테이블에 기록하도록 데이터베이스를 구현하였다. 이 결과로 관리자는 log 테이블을 통하여 부적절한 사용자의 접근을 찾아낼 수 있고, 주기적인 점검을 통해 사전적 예방의 효과 또한 기대할 수 있다.

3.3 구현

그림 3, 4는 인증을 위한 사용자 등록 페이지를 구현한 화면이다. 관계자 또는 방문자인 경우를 고려하여 개인식별번호, 이름, 전화번호를 입력하고 완료되면 얼굴 녹화를 하여 사용자 등록을 한다.



(그림 3) 사용자 등록화면의 정보입력화면 구현



(그림 4) 사용자 등록화면의 얼굴녹화화면 구현

관리자는 사용자가 등록한 정보와 얼굴 이미지를 사용하여 128D 특징을 추출하여 저장한 csv파일과 LivenessNet으로 훈련한 liveness.model, le.pickle을 인증 장치에 배포한다[11].

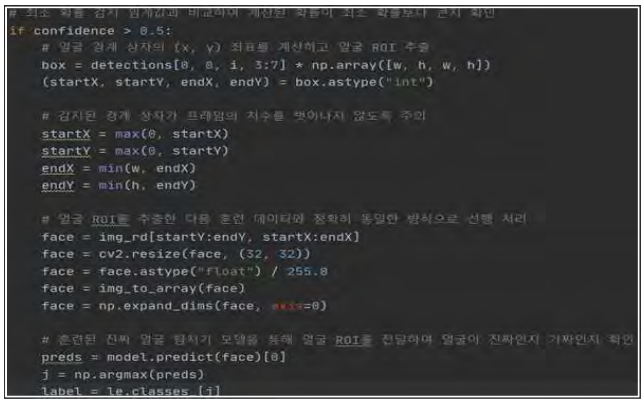
그림 5, 6은 배포 받은 파일로 Face Recognition과 Liveness Detection을 구현한 코드이다. 사용자가 카메라로 얼굴인증을 시도하면 인증 장치는 Face Recognition 기능을 먼저 수행하고 Liveness 모델과 눈 깜빡임으로 스푸핑 감지를 하여 Liveness Detection 기능 순으로 수행한다. 두 기능들이 완료되면 얼굴인증 성공으로 간주한다. 하나의 기능이 실패할 시에는 인증 실패 여부를 시각화하여 사용자에게 알려준다.

```
#인식된 사람의 수가 변하지 않을 때
if self.current_frame_faces_cnt == self.last_frame_faces_cnt:
    print("      >>>>>> scene 1: no faces cnt changes in this frame!!!")
    # 사람이 안 잡 없을 때
    if self.current_frame_faces_cnt != 0:
        #얼굴 위치들 가져온다
        for k, d in enumerate(faces):
            #얼굴 범위 박스 계산
            height = (d.bottom() - d.top())
            width = (d.right() - d.left())
            hh = int(height / 2)
            ww = int(width / 2)

            cv2.rectangle(img_rd,
                          tuple([d.left() - ww, d.top() - hh]),
                          tuple([d.right() + ww, d.bottom() + hh]),
                          (255, 255, 255), 2)

            print("      >>>>>> self.current_frame_face_names_list[k]:      ",
                  self.current_frame_face_names_list[k])
            print("      >>>>>> self.current_frame_face_pos_list[k]:      ",
                  self.current_frame_face_pos_list[k])
```

(그림 5) Face Recognition 구현 코드



```
if confidence > 0.5:
    # 얼굴 경계 상자의 (x, y) 좌표를 계산하고 얼굴 ROI 추출
    box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
    (startX, startY, endX, endY) = box.astype("int")

    # 감지된 경계 상자가 프레임의 치수를 벗어나지 않도록 주의
    startX = max(0, startX)
    startY = max(0, startY)
    endX = min(w, endX)
    endY = min(h, endY)

    # 얼굴 ROI를 추출한 다음 훈련 데이터와 정확히 동일한 방식으로 전처리
    face = img_rd[startY:endY, startX:endX]
    face = cv2.resize(face, (32, 32))
    face = face.astype("float") / 255.0
    face = img_to_array(face)
    face = np.expand_dims(face, axis=0)

    # 훈련된 진짜 얼굴 탐지기 모델을 통해 얼굴 ROI를 전달하여 얼굴이 진짜인지 가짜인지 확인
    preds = model.predict(face)[0]
    j = np.argmax(preds)
    label = le.classes_[j]
```

(그림 6) Liveness Detection 구현 코드

얼굴인증 성공 시, OTP 번호가 애플리케이션에 전송이 되어 사용자는 OTP 인증을 시도한다. OTP 또한 인증 실패 여부를 시각화하여 사용자에게 알려준다.

4. 결론

본 논문에서는 변조된 얼굴을 이용한 안면인식 보안 우회를 방지하기 위한 얼굴인증 프레임워크를 설계 및 구현하였다. 얼굴인증 프레임워크는 일반적인 얼굴인식 뿐만 아니라 Liveness Detection 기능을 추가하여 실제 사람의 얼굴 여부를 감지하도록 구현해 스푸핑 공격을 방지하고자 하였다. 또한 더욱 안전한 인증을 위해 기존 단일인증의 한계를 보완하고자 OTP를 이용한 다중인증 기능을 구현하였다. 더 나아가 본 논문은 프레임워크로 기능을 제시하기 때문에 맞춤형 제작 및 도입이 가능하며 비대면 인증이 확산하는 사회 분위기에 따라 비대면 인증 환경 조성에 기여할 것으로 예상한다. 따라서 얼굴인증 프레임워크를 다양한 분야에 활용할 수 있고 더욱 안전한 본인 인증 환경 조성에 도움이 될 것으로 기대한다.

[본 논문은 과학기술정보통신부 정보통신창의인재양성사업의 지원을 통해 수행한 ICT멘토링 프로젝트 결과물입니다.]

참고문헌

[1] 과학기술정보통신부, “과학기술&ICT 정책 기술 동향”, 148호, p. 1, 2019

[2] 조상래, 김수형, 진승헌, “얼굴 인식에서의 스푸핑 공격 탐지 연구 동향”, p. 18, 2018

[3] Martin Heller, InfoWorld, “빅 브라더를 위한 인공지능, 안면 인식의 의미와 스푸핑”, 2020

[4] Vahid Kazemi, Josephine Sullivan, “One Millisecond Face Alignment with an Ensemble of Regression Trees”, 27th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), p. 1-8, 2014

[5] 김가현, 엄성민, 서재규, 김동익, 박강룡, 김재희, “Face Liveness Detection Based on Texture and Frequency Analysis”, 5th IAPR International Conference on Biometrics (ICB), New Delhi, India. p. 67-72, 2012

[6] Tereza Soukupova, Jan Cech, ”Real-Time Eye Blink Detection Using Facial Landmark“, 21st Computer Vision Winter Workshop, 2016

[7] Wikipedia, Google Authenticator, "https://ko.wikipedia.org/wiki/Google_Authenticator“

[8] 김선애, “[보안특집]차세대 인증 시장 주도하는 FIDO”, 데이터넷, 2015

[9] 윤현종, 박윤주 ,“가짜 얼굴로도 다 뚫린다는 안면인식기술… 이렇게 무너지나”, 인터비즈 , 2020

[10] 김민수, “인쇄한 얼굴 사진에도 뚫려… MS ‘윈도우 헬로’ 보안 허술” , CBS노컷뉴스 , 2017

[11] Adrian Rosebrock, “Liveness Detection with OpenCV”, pyimagesearch, 2019