# Video injection attacks on remote digital identity verification solution using face recognition

**Kévin CARTA**
CLR-Labs
La Ciotat, France

**Claude BARRAL**
Bactech
Aix-en-Provence, France

**Nadia EL MRABET**
Ecole des Mines de St Etienne
Gardanne, France

**Stéfane MOUILLE**
CLR-Labs
La Ciotat, France

## ABSTRACT

The COVID 19 pandemic has fuelled the acceleration of the use of remote services as, for example, video conferences or digital identity verification solutions. Due to videoconferences or social medias, attackers have access to rich biometric sources and therefore make it possible to carry out high quality attacks such as videos of deepfakes, or morphing, against face recognition system. These kind of video attacks allow the attacker to fool face recognition even when these systems are secured by challenge-based liveness detection by presenting them. In order to prevent against these kind of attacks, adding an artefact detection to these systems could be a good solution. However, we will see in this paper that the development of remote digital identity verification tools on mobile application or on a computer (through a web app) opens the path to video injection attacks which bypass all of these security systems, namely a face recognition system secured with both challenge-based liveness detection and artefact detection.

**Keywords**: Video injection attack, image injection attack, face recognition, liveness detection, mobile application hacking, deepfake, morphing.

## 1. INTRODUCTION

As of today, too many automated systems are unable to identify and verify the person itself. This bring the issue that they cannot distinguish between an authorised person and an intruder who can gain access to the system fraudulently. Biometric systems are able to authentify an individual and are more convenient to use because an access can be secured without the burden of remembering passwords or carrying any personal device.

Nethertheless, if biometric systems offer great advantages, they are vulnerable to potential attacks and should not be used as a stand-alone authentication solution but need to be associated to another authentication's factor (a password or an access card). As presented in the international standard ISO / IEC 30107-1: 2016, there are 9 different points of attack divided in two types of attacks [8]:

**Direct attacks**. These are attacks that do not require any specific knowledge on how the system works, such as the matching algorithm in use, the feature vector format, etc. Only type 1 attack, so-called presentation attack, is a direct attack. To perform a presentation attack, an attacker presents to the biometric sensor a biometric trait, usually called Presentation Attack Instrument (PAI), which could be artificial, so-called artefact (like a 3D mask of a person's face), or natural (a coerced person is an example of natural PAI). Presentation attacks are the most famous attacks against biometric systems and the most accessible ones for attackers who try to steal the identity of someone else (impostors).

**Indirect attacks.** Unlike direct attacks, these are attacks for which information about the inner workings of the authentication system is needed for an attack to succeed.

Figure 1 presents the 9 different points of attack on biometric systems [14].

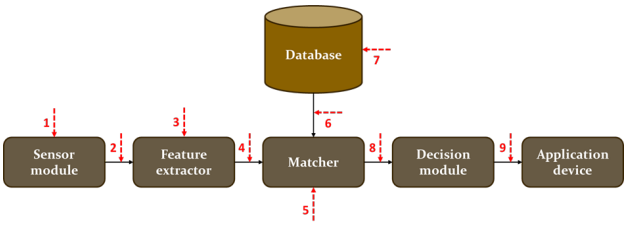


**Fig. 1**: Different points of attack on biometric systems

As presentation attacks are the major threat for biometric systems, they are usually complemented by presentation attack detection systems. One of those systems is challenge-based liveness detection modules. This feature is able to detect if the challenges asked randomly are correctly done by the user and thus if the user is a human being.

In the work [2], the authors detail the 9 points of attacks and show that there were many tools to bypass liveness detection solutions based on challenge-response for face recognition. While some tools do not require an important biometric source from the victim for the impostor, it is shown that video conferences offer to impostors the opportunity to use bona fide videos of the victim or to create superior quality attacks, in particular thanks to deep learning algorithms.

However, this latter referenced article states that it is possible

ISBN: 978-1-950492-61-9
ISSN: 2771-5914



https://doi.org/10.54808/IMCIC2022.02.92

to countermeasure with these presentation attacks by associating this solution based on challenges with other security features: either basic such as counters or replay detection solutions, or artefact detection solutions which are able to detect videos displayed on screen, or both.

In this article, we will pay particular attention to type 2 attacks. These attacks consist of modifying or replacing the biometric sample that is communicated between the sensor and the extractor. In the case of face recognition, this attack involves the attacker hooking the camera and injecting a different photo / video into the system other than the stream captured live by the camera.

Nowadays, we can see a brand new usage of face recognition with its implementation into remote identity verification solution, on mobile application or on web browser, in order to have secured digital identity check [6]. These kind of solutions prove a real emergence on the market and thus hold special attention from institutions, particularly in Europe, which already develop certification schemes in order to insure their relevant security level before implementing them on private or public domains [1] [3].

This type of solution consists in checking, with a sufficient level of insurance, the identity of a customer by authentifying an identity document and verifying that the one who presents the ID document is the legitimate person thanks to face recognition. More details about this kind of solutions are presented later in Section 3.1.

The advent of remote identity verification solutions on the market gives to impostors the opportunity to achieve new type of attacks in order to steal the identity of a victim. Indeed, for now biometric solutions were highly threatened by presentation attacks, which do not require any knowledge or particular access to the remote identity system. Nowadays, more and more remote solutions give to the attackers a privileged and not scrutinised access to the system. Here, we will see that type 2 attacks are more accessible than they seem to be, and we will show that the attacks prepared in the publication [2] can be injected to bypass a complete face recognition system even secured with both passive and active liveness detections. Our case study will be a mobile application with a complete and secured face recognition system for doing a digital identity that we will detail later in Section 3.

In Section 2, we will deal with liveness detection, how we can hack a mobile application in order to inject data and how to prepare video attacks. In Section 3, we will present the architecture of the targeted system in our experiments and how we prepared our new attacks. In Section 4, we will present the success rates of our attacks on the target. Eventually, in the last section, we will say some words on what should be remembered from this work and what will be our next experiments.

## 2. STATE OF THE ART

### 2-A. Liveness detection

As its name suggests, liveness detection makes it possible to establish whether the biometric trait presented to the sensor comes from a living person or an artificial artefact.

In face recognition, there are many liveness detection solutions that work on many different principles. Here is a non-exhaustive list [8]:

- Pupil dilation
- Using 3D camera for depth info
- Texture detection
- Motion detection (mouth, eyes, head)
- Challenge-response detection

Most of these solutions are based on machine learning. Among them, there is a distinction between active and passive liveness detection. We talk about ***passive*** when the liveness detection module does not require any effort from the user to determine whether it has a PAI in front or not. Most of passive liveness detection on the market are actually what is known in the litterature as artefact detection which tries to detect a presentation attack by identifying known PAIs like a printed photo, a 3D mask or a video displayed on a screen for instance [12] [13].

Otherwise, we talk about ***active*** when the liveness detection module does require an action from the user, which is the case with challenge-based approach [16]. Challenge-response detection consists in asking the user for random challenges (such as blinking or opening mouth, for example), which can counter a trivial video or stolen photo of the victim presented in front of the camera.

### 2-B. Mobile application hacking

To carry out a type 2 attack, it is necessary to be able to fraudulently inject data into the mobile application, which means hacking the mobile application [10]. In this article, we will take as an example a mobile application on Android mobile system.

In order to inject our attacks, we used the Frida tool [11]. This software is a dynamic instrumentation toolkit which lets one inject snippets of Javascript into native apk. It's a scriptable, portable and free tool which can be controled under Python language. Frida is divided into two parts: a server part which is located in the mobile phone and a client part which is located on the PC side, connected via USB to the phone, and which will notably inject the JS script.

Frida is like a debbuger. With this software, one can modify and analyse application at runtime: read/write function arguments or modify function behavior. It is also useful to bypass client-side security controls like root detection. It is particularly used by laboratories which perform pentesting, or penetration testing, on mobile applications. A pentest, or penetration test, is a method of analyzing a target by putting oneself in the shoes of an attacker.

Using Frida on rooted smartphone makes it easier to use. With only basic knowledge, rooting is usually simple and fast to achieve as there are lots of different tools and tutorials on the web which make it feasible by anyone. Here are the different tasks to complete in order to inject data into the mobile application:

- Rooting the smartphone and having super-user access (possibly hide root if needed)
- Install Frida server into the smartphone
- Decompile the apk code (possibly deobfuscate if needed)
- Identify the interesting functions in order to hook the camera and to know where to inject the videos and with which format
- Manipulate mobile application to desired behavior: this is the javascript step. This is where we retrieve what the application asks and where we can inject data into the application mimicking the camera thanks to our script.

### 2-C. Video attacks

The type 2 attack has the advantage of bypassing any screen detection and improves the success of the video attacks presented in [2]. Realising a video attack does require a biometric source from the victim: this one can be a photo, a video, a face scan, etc. Of course, each biometric source does not require the same effort from the attacker : while a 3D scan of the face of the victim is barely impossible to obtain, finding a photo is really simple at the age of internet and social medias. Here are different examples of video attacks which can be made from different biometric source from the victim:

- A simple face photo. With the help of a simple photo, it is notably possible to carry out these attacks:
  - Photo montage to create new photos where the victim realises the challenges and then the addition of morphing between the different photos to give a movement effect. (Photo-montage)
  - Achieving low-quality deepfake videos using an Android mobile application. A deepfake video attack consists in putting the face of a subject A on the face of the subject B during a video. The idea here is to put the face of the victim on the face of the attacker realising the challenges in a video. The resolution of the result video can then be increased using tools based on artificial intelligence available on the market. (Deepfake-LQ)

**Fig. 2**: On the left the attacker, in the middle 3 frames from the Deepfake-LQ attack (eyes completely open, mid open and closed), on the right the victim

  - The realisation of morphed photo which can then be used in all the attacks mentioned above [15]. A morphed face is created by resembling the biometric information of the two or more individuals. The created morphed face image will be successfully verified against probe samples of both contributing subjects by state-of-the-art face recognition systems. Morphing does not have to be confused with deepfake attack whose result won't be verified as both subjects used for making it. Thus, the use of a morphed face for the attacks presented above has an interest as it can allow several people to use the identity of the victim or it can make the deepfake more discreet if there was a human verification after the software verifications for example.

**Fig. 3**: On the left the attacker, in the middle the morphed face, on the right the victim

- A stolen video from the victim during a video conference. The authors in [2] show that the widespread use of video conferences, well helped by the global health situation, offers a very rich facial biometric source for an attacker. This wealth allows the attacker to develop attacks of much higher quality by using the stolen videos to train deep learning algorithms. It can help him to develop:
  - High quality deepfake attacks using various open source solutions based on deep learning. (Deepfake-HQ) [9]

**Fig. 4**: On the left the attacker, in the middle 3 frames from the Deepfake-HQ attack, on the right the victim

  - High quality morphing videos using an open source solution based on deep learning. (Morphing-video) [15] [9]

## 3. OUR EXPERIMENTS

### 3-A. Context

As precised in the introduction of this article, there is today an emergence of remote identity verification solutions in the market. These solutions have the objective to give a proof of one's identity for using a digital service with the right level of authentication. They can be deployed whether on web application or mobile application. In this article, we will only focus on mobile application solutions. They rely on two steps:

- **First step:** The user identification is done thanks to the verification of the authenticity of the identity document which is presented to the smartphone. The mobile application takes the photo and information present in the document by scanning optically the datapage thanks to the camera or reading the electronic chip of the document thanks to the NFC interface of the smartphone. It allows the application to get all information about the user (and the photo for face recognition) and to verify if the ID document is authentic.
- **Second step:** The principle is to have an acceptable level of assurance that the one who present the identity document is the legitimate person. It is processed thanks to face recognition verification between the face of the person who holds the smartphone and the face picture printed on the ID document or stored into the electronic chip.

Remote identity verification solutions have a huge interest for companies which deploy services on the web and particularly banking, border crossing, or governments services which need to have a huge confidence into the identity of their users. Thanks to these solutions, these companies have a more secured access to their services for their customers.

For this experiment, we will consider an Android mobile application for performing biometric verification by remote face recognition with the support of an identity document (passport, ID card or citizen ID card). As the script for injecting data depends on each use case, we will not enter into details about the javascript used for our specific target.

### 3-B. Mobile application description

The objective of this mobile application is to collect the elements composing a secure digital identity: it identifies an individual using his identity document by reading the chip of the ID document using the smartphone and then authenticates the individual in order to ensure that the holder of the identity document is legitimate by using the face recognition matching.

For this biometric phase, the mobile application uses the smartphone camera to perform biometric verification by face recognition. This subsystem is composed of three essential blocks:

- An active liveness detection based on the challenge-response principle: the individual is asked to perform various challenges at random. This solution asks the user to open or close his eyes and open or close his mouth in random order.
- A passive liveness detection (artefact detection): the purpose of this tool is to detect the presence of known PAIs in the field: presentations of paper-printed face, videos presented on a screen, 3D masks, etc.
- A face recognition algorithm: it will perform a matching between the person facing the camera during the realisation of the biometric verification and the photo present in the identity document compliant with the international standard ISO / IEC 19794 -5 format [7]. In this article, we will not focus on the identity document and will take the hypothesis that it was stolen from the victim.

The mobile application operates under the server-client scheme. The application is the client side that allows taking images and reading the electronic document, but all security operations are performed on the remote server side:

- The server communicates the challenges to be performed to the application
- The mobile application captures images of challenges made by the user and communicates them to the server
- The server performs the various checks (liveness detections and matching) and communicates the result to the mobile application

The architecture of the biometric verification achieved by the solution that we will consider in our experiments is depicted in figure 5.

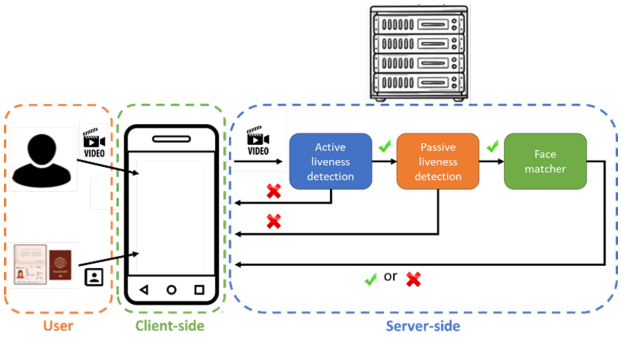


**Fig. 5**: Targeted solution architecture

### 3-C. Our attack scenario

We have considered a biometric impostor who wants to challenge the complete biometric solution : both liveness detection system and biometric matching. Thus the attacker will try to steal the identity of a particular victim by bypassing the liveness detection and the biometric matching. This means that the impostor will have to fool the active liveness detection module (i.e. being able to realise the different challenges asked by the system), the passive liveness detection module (i.e. being able to realise the different challenges without using any presentation attacks, such as 3D mask or video displayed on a screen, which will be detected by this module) and the biometric matcher (i.e. doing the attack with the face of the victim to be recognised as the victim). The attacks presented in [2], and summarised in Section 2.3, would be detected by the passive liveness detection module as the videos are displayed on a screen.

Type 2 attacks, which relies on hooking the camera, make sense for the attacker as the passive liveness detection module will see the attack video as a bona fide video filmed by the smartphone camera. The attack will thus consist in preparing fake videos of the victim realising the different challenges orders (such as those presented in Section 2.3.) and injecting them into the hacked mobile application as described in the next subsection.

If the solution would rely onto a single image, and not a video (which means a system without active liveness detection module thus less secured than the one depicted in this paper), injecting a single bona fide image of the victim may require less skill and effort than injecting a video sequence that contains the correct response to any challenges orders. However, into real-life market, most of these type of solutions do implement both active and passive liveness detection modules. The most simple attacks such as bona fide images or raw videos of the victim would not fool such systems.

Eventualy, we can note that another attack scenario would be to use physical or digital fraudulent identity document (for exemple a video of a fake identity card of the victim with the photo of the impostor which would be injected into the hacked application) in order to steal the identity of the victim. Yet, this scenario relies on document fraud which is a less accessible domain for the general public. This particular scenario will be the subject of future researches.

### 3-D. Injection's script role

Type 2 biometric attack will allow a hook to be made at the level of the smartphone camera. In our example, this will consist of overwritting what the camera is filming during verification and instead sending a pre-made video by the attacker to the various liveness detection modules and the matching algorithm.

As presented previously in this article, it is necessary to work with a rooted phone, to have deobfuscated the application (if the code is obfuscated) and to have decompiled the application code. Once these steps are done, the attacker is able to identify: a/ which functions of the application call the camera, b/ where the images captured images are stored, and c/ the image format.

The attacker can then write a code in javascript format in order to modify the behavior of the application and obtain the various requests from the server:

- learn the server requests (order of challenges) before it is displayed on the screen of the smartphone.
- re-route, or wipe, information taken by the camera.

- call up the pre-made videos according to the server's request. The videos of the attack are in fact previously stored in the rooted filesystem as a list of images, in the specific image format of the phone in use (adequate resolution and Android raw format) which will be recompiled on the fly by our script into a video file, mimicking the video taken by the camera, before being sent to the server.

Note that it is possible for the attacker to reconstruct his attack in real time. Indeed, he can prepare in advance each different possible challenge order (for example, open mouth and closed eyes then closed mouth and open eyes, is a possible challenge order) that he stores in the application memory. Alternatively, it is also possible for him to prepare only the possible challenges (4 different challenges) and then to reconstruct the final video in real time, according to the order of the challenges requested by the server, using the Javascript file. This solution requires less preparation on the attacker side, however it has the drawback of incorporating "cuts" in the final video (as the position of the head in the different challenge videos is not exactly the same), which would be visible to a human operator in certain use cases or detected by a dedicated countermeasure.

To summarise, the type 2 attack allows the attacker to interfere here in the architecture of the application in order to modify the video stream that is sent to the server, depicted in figure 6.

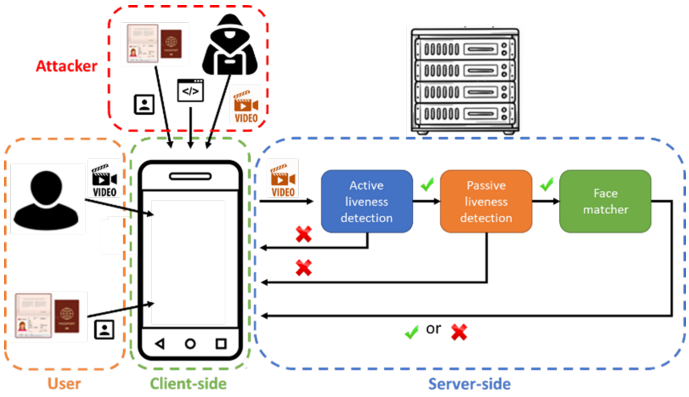


**Fig. 6**: Camera hooked in the targeted solution architecture

### 3-E. Video injection

We have used all the video attacks presented in Section 2.3 in our work. Moreover, we developed another attack which mixed the tools used for Deepfake-LQ and Deepfake-HQ attacks. We realised a video of the victim doing the challenges with only one photo thanks to the methodology of the Deepfake-LQ attack and we have used this video in order to train the deep learning algorithm of the Deepfake-HQ attack. This allowed us to develop a better quality deepfake than the Deepfake-LQ attack with the same and easily accessible biometric source: a simple photo. We named this attack Deepfake-LQ+HQ.

**Fig. 7**: On the left the attacker, in the middle 3 frames from the Deepfake-LQ+HQ attack representing 3 different challenges, on the right the victim

As presented in the previous subsection, once all the video attacks were prepared, at the right video format of the smartphones we used (Samsung Galaxy S8 and S9), we decompiled all the videos into a set of images that we stored into the application-dedicated area in the Android filesystem. All we had to do then was the injection thanks to our script that we called with Frida from our computer, connected via USB to the rooted smartphone, when the mobile application asked us to realise the face recognition (challenges).

## 4. OUR RESULTS

As we stated previously in this article, the attacker has the ability to inject his attacks in two different ways:

- Using only a few videos of the different challenges that he will reconstruct live using the script to obtain the complete video with the order of the challenges requested by the server. This solution requires less work from the attacker because the number of videos to prepare is very low but the reconstructed videos present cuts during reconstruction and therefore the video is not perfectly smooth. In our example, the attacker only prepares four videos (eyes closed or opened + mouth closed or opened).
- Using the full videos of each possible challenge order. This solution requires more work (24 different possible orders) from the attacker, but the attacks do not present any defect in fluidity and are more effective because no time is wasted reconstructing the final video, but consuming far more storage memory resources.

In order to carry out the attacks, depending on the various attacks but also according to the injection method adopted by the attacker, there is a time-memory trade-off and computation complexity trade-off (so-called time-memory-complexity trade-off). In order to understand the different impacts, it's important to consider two different parts of the attack: preparation of the attack and realisation of the attack.

When carrying out the attack, this time-memory-complexity trade-off is not of major importance in our use case because current smartphones are powerful enough and have a sufficiently large memory to offer similar realisation times on a macroscopic scale. However, it this should be taken into account in the case of exponential multiplication of challenges combinatories.

When it comes to preparing for the attack, many parameters are to be taken into account:

- The time to retrieve the biometric source which can be significantly different if it is a simple photo or a video
- The production time of the attack video which will require very different preparation times depending on the attack to be carried out but also according to the chosen injection method. Since, depending on the made choice, it can increase the number of videos to prepare (4 for the first method and 24 for the second in our use-case, but could many more).

Note that for the two injection methods, we will only take into account the preparation time for making the videos. Two weeks of work must be added to the preparation times indicated in the following tables in order to setup the injection platform.

Our two injection methods were used on different real-life market solution that implement all of the security features that we described in the previous section. Here are the results we obtained with the attacks presented in Sections 2.3 and 3.4, onto

one of these market solutions, knowing that we processed each attack 20 times per type of injection:

| Attack | Preparation time | Biometric source | Success rate |
|---|---|---|---|
| Photo-montage | 5 hours | A photo | 0/20 = 0% |
| Deepfake-LQ | 2 hours | A photo | 0/20 = 0% |
| Deepfake-LQ+HQ | 5 days | A photo | 0/20 = 0% |
| Deepfake-HQ | 5 days | A video | 2/20 = 10% |
| Morphing-video | 5 hours | A video | 1/20 = 5% |

**Table 1**: Results for attacks reconstructed in live

| Attack | Preparation time | Biometric source | Success rate |
|---|---|---|---|
| Photo-montage | 7 hours | A photo | 0/20 = 0% |
| Deepfake-LQ | 7 hours | A photo | 0/20 = 0% |
| Deepfake-LQ+HQ | 1 week | A photo | 1/20 = 5% |
| Deepfake-HQ | 1 week | A video | 4/20 = 20% |
| Morphing-video | 1 week | A video | 4/20 = 20% |

**Table 2**: Results for all challenge orders prepared in advance

As we can see, video injection poses a real threat to even top-level secure face recognition systems. In addition to the security features stated in section 3, it should be noted that the system on which we worked had additional securities dedicated to injection attacks, which explains why we obtained very poor results with the live video reconstruction method. Without these additional securities, we would expect much higher attack success rates for both injection methods.

## 5. CONCLUSION

As we have seen, the development of remote digital identity verification solution paves the way for new attacks that are more accessible for attackers because the mobile phone is under the attacker's control with no scrutiny from any authority. Moreover, the same type of solution that would be used not via a mobile application but via a web browser is even more vulnerable to this type of attack because a simple webcam simulation tool makes it possible to inject videos via the web browser (e.g. OBS Studio).

Since the smartphone or computer is under the attacker's control, and because any type of security (code obfuscation, anti-rooting detection, etc.) can be bypassed by the attacker, it is absolutely necessary that the countermeasures, faced with the new vulnerability of image injection, must be placed on the server side in order to ensure greater robustness. Additionnaly, the client-side (i.e. the smartphone) could be complemented by the implementation of so-called mutual authentication and secure channel protocol (as it exists today in smartcard-based security solutions) between the camera and the Android operating system [4] [5]. In our next work, we will focus in particular our researches for countermeasures to overcome this new breach facing face recognition.

## 6. REFERENCES


[1] ANSSI. "**Référentiel d'exigences ANSSI - Prestataires de vérification d'identité à distance - version 1.1**". In: (2021). URL: https://www.ssi.gouv.fr/uploads/2021/03/anssi-referentiel_exigences-pvid-v1.1.pdf (visited on 03/23/2021).

[2] K. Carta *et al.* "**On the pitfalls of videoconferences for challenge-based face liveness detection**". In: *To appear in proceedings of World Multi-Conference on Systemics, Cybernetics and Informatics 2021*. 2021.

[3] ENISA. "**Remote ID Proofing**". In: (2021). URL: https://www.enisa.europa.eu/publications/enisa-report-remote-id-proofing (visited on 05/27/2021).

[4] ETSI. "**TS 102 484 : Secure channel between a UICC and an end-point terminal**". In: (2012). URL: https://www.etsi.org/deliver/etsi_ts/102400_102499/102484/11.00.00_60/ts_102484v110000p.pdf (visited on 05/28/2021).

[5] GlobalPlatform. "Secure Channel Protocol '03'". In: (2019). URL: https://globalplatform.org/wp-content/uploads/2014/07/GPC_2.3_D_SCP03_v1.1.2_PublicRelease.pdf (visited on 05/28/2021).

[6] IATA. "**One ID Concept paper**". In: (2018). URL: https://www.iata.org/contentassets/1f2b0bce4db4466b91450c478928cf83/oneid-concept-paper.pdf (visited on 05/22/2021).

[7] ISO/IEC. "**19794-5:2011 Information technology- Biometric data interchange formats - Part 5: Face image data**". In: (2011). URL: https://www.iso.org/standard/50867.html.

[8] ISO/IEC. "**30107-1:2016 Information technology - Biometric presentation attack detection - Part 1: Framework**". In: (2016). URL: https://www.iso.org/standard/53227.html.

[9] P. Korshunov and S. Marcel. "**Vulnerability assessment and detection of Deepfake videos**". In: *2019 International Conference on Biometrics (ICB)*. 2019, pp. 1–6. DOI: 10.1109/ICB45273.2019.8987375.

[10] Ahmed Mohiuddin. "**False Image Injection Prevention Using iChain**". In: *Applied Sciences* 9 (Oct. 2019), pp. 1–11. DOI: 10.3390/app9204328.

[11] B. Mueller, S. Schleier, and J. Willemsen. **Mobile Security Testing Guide**. OWASP, 2019. Chap. Android Testing Guide.

[12] K. Patel, H. Han, and A. K. Jain. "**Secure Face Unlock: Spoof Detection on Smartphones**". In: *IEEE Transactions on Information Forensics and Security* 11.10 (2016), pp. 2268–2283. DOI: 10.1109/TIFS.2016.2578288.

[13] R. Ramachandra *et al.* "**Custom silicone Face Masks: Vulnerability of Commercial Face Recognition Systems Presentation Attack Detection**". In: *2019 7th International Workshop on Biometrics and Forensics (IWBF)*. 2019, pp. 1–6. DOI: 10.1109/IWBF.2019.8739236.

[14] N. K. Ratha, J. H. Connell, and R. M. Bolle. "**Enhancing security and privacy in biometrics-based authentication systems**". In: *IBM Systems Journal* 40.3 (2001), pp. 614–634. DOI: 10.1147/sj.403.0614.

[15] U. Scherhag *et al.* "**Face Recognition Systems Under Morphing Attacks: A Survey**". In: *IEEE Access* 7 (2019), pp. 23012–23026. DOI: 10.1109/ACCESS.2019.2899367.

[16] A. K. Singh, P. Joshi, and G. C. Nandi. "**Face recognition with liveness detection using eye and mouth movement**". In: *2014 International Conference on Signal Propagation and Computer Technology (ICSPCT 2014)*. 2014, pp. 592–597. DOI: 10.1109/ICSPCT.2014.6884911.