# For Your Eyes Only – Biometric Protection of PDF Documents

**J. Siciarek**[1], **M. Smiatacz**[1], **and B. Wiszniewski**[1]
[1]Dept. of Intelligent Interactive Systems, Faculty of ETI, Gdansk University of Technology, Gdansk, Poland

**Abstract**— *The paper introduces a concept of a digital document content encryption/decryption with facial biometric data coming from a legitimate user. Access to the document content is simple and straightforward, especially during collaborative work with mobile devices equipped with cameras. Various contexts of document exchange are presented with regard to the next generation pro-active digital documents proposed by authors. An important point is that documents developed in the project referred in the paper use common formats, it this particular case PDF.*

**Keywords:** Face recognition, biometric password, encryption

## 1. Introduction

Documents play a key role in any organized structure invented by human civilization, as they are both *information units* and *interface units* to collaborating people. This phenomenon is particularly common in knowledge based organizations, whose members, often called *knowledge workers*, cooperate by exchanging documents to make decisions, discover facts or accumulate knowledge.

The MENAID project [1] addresses these issues by approaching collaborative computing and virtual cooperation with next generation pro-active digital documents, which are *mobile*, *interactive*, *executable* and *intelligent*. Before going to the main theme of the paper let us refer to two basic concepts of such documents developed during the project, namely a *Mobile Interactive Document (MIND)* architecture of documents capable of traveling through Internet as autonomous agents, and an *Interactive Open Document Architecture (IODA)* enabling implementation of executable papers that may be freely transferred between digital libraries. By referring to MIND and IODA we can emphasize better the issue of protecting content from unauthorized access during various document exchange scenarios. An important point is that our architectures do not require any specific representation format for the document, as any content conformant to the MIME standard may be exchanged.

### 1.1 Mobile documents

Architecture of MIND is based on a simple combination of two popular Web concepts: automatic XML data binding with Java objects, and mobile agents. Data binding allows for converting units of information contained in document components into functional objects in a computer memory, augmented next with mobility to make them autonomous objects, capable of migrating in an open distributed system [2]. Owing to this, a static representation of an electronic document is transformed into a set of dynamic objects that can migrate to remote locations, and perform actions at these locations by interacting locally with their users and services (see Figure 1).

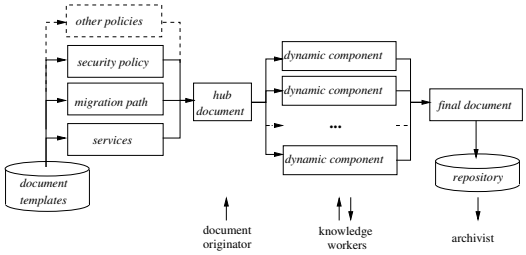


Fig. 1: Components of the MIND mobile document

Lifecycle of a MIND document is initiated by its *originator*, a worker who is responsible for designing a logical structure of a *hub document*, possibly by using a repository of *document templates*. Logical structure of the document includes in particular: a document *migration path*, and specification of *services* required by it during its journey through various locations of the collaborating workers' organization. Document *security policy* specifies access rights to various document parts between subsequent collaborators, indicated as participants of the workflow process defined by the document migration path.

Migration path is specified with a workflow definition language, in terms of *activities* to be performed by each respective worker upon document delivery, and *transitions* of documents to other workers. In our current implementation of MIND we utilize email messaging for document transitions. A lightweight email client, implemented by us for that purpose, can automatically retrieve documents attached to the email message from the receiving worker's mailbox, authenticate the worker, and make them executable objects by unmarshaling. Upon completion of one activity, the email client interprets a document migration path to determine the next activity and workers, serializes documents and sends them as email attachments.

Knowledge workers interact with dynamic components currently residing on their personal devices and contribute to their content when performing the required activity. Upon completion of the last activity specified by the migration path, document components return to their originator to be integrated into a final document and archived.

## 1.2 Executable papers

A key point about the IODA architecture [3] is that it enables paper executability without subscribing to any specific document format nor requiring implementation of document functionality in any particular programming language. It just provides a sort of a *spine* that binds common tools and services that already exist in a Web document ecosystem. A document spine is implemented as an XML file that simply specifies components of a multi-layered structure of digital objects (see Figure 2).

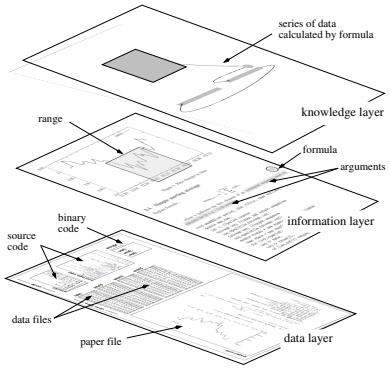


Fig. 2: Layers of the IODA executable paper

Functionality making IODA content executable may be implemented directly with standard notations used for providing functionality and interactivity of Web pages, from simple scripts, through Web- browser plug-ins available from third-party repositories, up to private or public Web services.

The base *data layer* contains a principal document in its native format, such as TIFF or JPEG for scanned analog documents, or PDF for born digital documents. Along with a principal document the data layer may contain data files, other associated documents, and services, like in the case of MIND documents. Association of documents, data and services is implicitly defined by patterns used to interpret a principal document fragment, and other components of the data layer. Interpretation patterns constitute the *information layer*. IODA does not impose any specific format to define these patterns. They are defined with metadata tags marking fragments of a principal document. Markings may take the form of a *point* (a character or a pixel), a *text range*, a rectangular *page area*, or a structure (a collection of any of the former). Tag's metadata associate each respective marking with a function (and if necessary with other objects of the data layer), which is executed upon receiving events generated by user actions. User actions performed by clicking on a marked document fragment may be any service specified by its related metadata. Services of the information layer combine respective data of the data layer to facilitate user interaction with marked fragments of IODA documents. However, user interaction scenarios, e.g. performed by reviewers, may require contexts of interpretation patterns implemented by the information layer. This is a task of the knowledge layer, which provides mechanisms for dynamic *annotations* and *links*. These operations are performed mostly by using a standard browser functionality.

# 2. Secure document delivery

Just until recently it was enough to protect an electronic document, stored in a server somewhere on the Web, by a security mechanisms built in the service granting access to the document content. But when documents are allowed to leave their safe habitat and interact with users after installing themselves on unfamiliar devices (as in the case of MIND or IODA documents described before) they are on their own – very often outside of a trusted (or even out of any) network, from which they might try to get support to protect themselves. One example might be calling authorization services of their home (originator's) server (see Figure 1). Mechanisms embedded in documents to protect access to their content against unauthorized users or tools must cope not only with the possible document theft, but also loosing an access device with the protected document. Certainly the latter is worse, as personal devices may have already a software installed that enables access to the protected document. On the other hand the protection mechanism should be easy to use and intrinsically related to the legitimate document user identity. Throughout the rest of this paper we will argue to base such mechanism on biometrics.

## 2.1 Biometric protection of data

There are at least two reasons for the increasing interest in biometric security solutions. First, a properly working biometric algorithm adds another security layer to the protected system, so brute force style password attack is less effective. Second, it can aid computers in improving their experience with users. A typical workstation is still blind today, so it keeps asking the user "is that you?". Once it is able to recognize the person in front of the screen, it may "feel safer" and stop bothering users with explicit identity checks.

### 2.1.1 Face recognition

From all biometric techniques the face recognition ones seem to be the best choice when it comes to securing access to the document content. Fingerprint readers are far less popular than cameras, identification by means of voice recognition could be cumbersome, and behavioral methods are less useful in this context. During our work on biometric security systems for mobile devices in the SART-2 project [4] we have developed a universal *Classification Framework* (CF) library [5], which can greatly simplify development and evaluation of any dedicated face recognition system. CF modules enable object localization, illumination

normalization, feature extraction, classification, etc., and can be easily plugged into any security protection system, and reconfigured, if necessary. In consequence, we have been able to combine various algorithms into one face identification library; it exports just two methods: one for training the system, and another for performing the identification. One of such configured libraries was employed in the PDF protection system described in this paper.

Our face recognition engine, used during experiments described in section 2.3, performs object detection with the popular method proposed originally by Viola and Jones [6], and proceeds similarly to the algorithm described in [7], passing the cropped and histogram-normalized image to the two feature extraction pipelines; one of them calculates *Local Binary Patterns*, and another performs *Gabor filtering*. Each pipeline creates the secondary feature space by using *Kernel Linear Discriminant Analysis* (KLDA), and then applies the multi-modal minimal-distance classifier with cosine metrics to compute the final decision. Identity verification is based on a simple voting scheme: if both pipelines agree and provide the positive result, the user is allowed to access the file, otherwise the access attempt is rejected.

We decided to use the configuration described above because our preliminary experiments [4] indicated that it performs sufficiently well. However, we can easily change it in order to adapt the system to somewhat limited capabilities of mobile devices, or to adjust the size of the biometric data that must be attached to the document sent over the network.

Any biometric system can be characterized by two important parameters: the *false acceptance rate* (FAR) representing the probability that an impostor will be able to force the blockade, and the *false rejection rate* (FRR) indicating how often the legitimate user will not be recognized properly. Increasing the security level through the reduction of FAR inevitably leads to the lower satisfaction of genuine users, as the FRR gets higher automatically. For the application presented here we opt for security rather than comfort of use. However, our face recognition library offers the possibility of tuning the FAR/FRR ratio by changing a simple parameter. Tests performed during the SART-2 project (68 cycles, each involving more than 13000 images) indicated that the current configuration provides FAR of about 4% with the acceptable comfort level. Those results were obtained for images coming from CMU-PIE database [8], which is a challenging dataset. Experiments described in section 2.3 were more limited, but carried out in typical real-life conditions.

Although we are using the general *face recognition* term, the action performed by our biometric security system shall be called *identity verification*. This means that the user attempting to open the document declares some identity to the system, and the role of the latter is to check whether the acquired feature vector $\mathbf{x}$ extracted from his/her face image is *similar enough* to the stored template. Many algorithms, however, have been designed to perform a typical *recognition*, i.e., their goal is to identify the class (person) which is *most similar* to $\mathbf{x}$. In other words, recognition is about distinguishing between known persons – it is a multi-class problem. In the case of verification there is only one class; thus, in order to use KLDA, we have to create an artificial "negative" class representing "the rest of the world". We used 66 frontal images from CMU-PIE for that, but the influence of the content of the negative set on the system's performance will be a subject of our further research.

Many face recognition systems, including various commercial applications, work incorrectly when illumination conditions are not perfect. Although recently we have developed an effective method of image normalization that is able to cope with poor lighting problem [9], we are still investigating how the different variants of this algorithm may interfere with other modules. Therefore, the normalization procedure remained switched off during the tests.

#### 2.1.2 Liveness verification

Inferior illumination conditions increase FRR and could make the system difficult to use. The main security threat, however, is related to the fact that in the case of face recognition biometric data can be obtained (or stolen) easily, e.g., from public Web pages. Any attacker may simply print the face image or even display it his/her smart-phone screen in front of the camera. This indicates that one of the most important elements of robust face recognition applications shall be the *liveness detection* module, capable of distinguishing between a real person and his/her photograph. To achieve such functionality, we have implemented an advanced eye-blinking detection mechanism combining the *Support Vector Machine* with *Conditional Random Fields* [10]. We have also proposed the special motion analysis procedure, based on the features extracted from the *optical flow field* [11]. Because we are currently working on a new version of the liveness detection algorithm, that module was not used during the tests described later in this paper.

#### 2.1.3 Emotion identification

In the extreme case yet another type of attack is possible – a legitimate user might be forced by aggressors to appear in front of the camera to unlock the document content. Although little can be done under such circumstances, the possibility of applying some anomaly detection technique that would be able to detect the unusual behavior of the person is worth investigating. Emotion recognition based on face images has been studied for a long time [12], the related algorithms, however, are usually based on the analysis of very distinctive face expressions. The symptoms of the high stress level may be subtle and vary considerably from person to person. Moreover, acquisition of experimental data may be expected to be particularly difficult in this case.

## 2.2 Secure PDF documents

PDF is one of the most popular document formats. It enables presenting documents to the end user in the same form as on author/designer desktop – documents are easy to share via Internet, to display by widely available viewers, like Acrobat Reader, Evince, or Web browsers with a proper plug-in installed.

Protection of PDF documents requires two passwords. One is for *owners*, who may have unlimited access to the document content and can change its passwords and access permissions, and another is for *users*, authorized to open the document and perform operations according to the user access permissions specified in the special *encryption dictionary* embedded in the document [13]. Access permissions may involve modifying, copying or extracting text or graphics, adding and editing text in annotations or interactive form fields, and printing. Unfortunately, there is no mechanism to force developers of PDF tools to honor access permissions set in a PDF document. This, however, has nothing to do with cracking encrypted PDF documents, which can be protected quite well with the standard Adobe's encryption handler using symmetric encryption keys [13].

### 2.2.1 Standard PDF security handler

The strength of PDF encryption is determined by the combination of both: the length and complexity of a textual password invented by the document owner, and the length of the encryption key supported by the PDF version used. With increasing computing power of CPUs , and yet more radical of GPUs, the RC4 algorithm with a 40-bit key used by older PDF documents yields them today practically defenseless against brute force or dictionary attacks. Publicly available tools, like HashCat [14] for example, can find a 40-bit key, i.e. crack any password, regardless of its length and complexity (complete key-space used), in less than 24 hours. With longer encryption keys the situation gets better, so that for eight or more characters from a complete key-space calculation of MD5 passwords is still a matter of months.

Certainly, taking an advantage of the full length of 32 character passwords and 128-bit (PDF 1.6) or 256-bit (PDF 1.7) keys is a must for secure PDF documents. But even long, a password may be weak and easy to guess or steal, so it is important to aid users in inventing complex and long passwords, especially when they have to handle many documents at the same time. The solution should be automatically generated passwords that are hard to crack, but do not have to be memorized nor can be lost.

### 2.2.2 Biometric password

We propose to exploit the standard PDF password mechanism mentioned above. The idea, shown in Figure 3, is to provide a password generating application with a (training) set of photos of a legitimate user, from which a *biometric definition* DEF of him/her is derived and used to generate a textual password. The document can be encrypted then with a standard algorithm of level 2 or 3, as specified by the value of parameter $V$ in a PDF document encryption dictionary mentioned before [13].

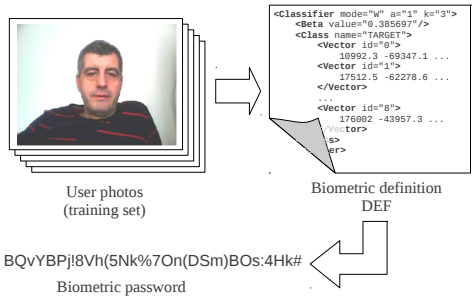


Fig. 3: Biometric password generation scheme

Upon receiving an encrypted document the user has just to face the camera of his/her computer or personal device. If the face in question matches the one required, the document can be decrypted. A dedicated piece of FYEO software to interpret biometric data can be attached to the document data layer – if implemented as an executable paper (see Figure 2), or plugged in a document browser – if implemented otherwise.

Prior to sending any FYEO protected document its potential recipient has to register at the *system training point*, shown in Figure 4. Each user is required to take a series of $n = 7, .., 25$ face shots for the training set. The set of images is processed by the *image to biometric data encoder*, which generates each respective user's face definition DEF in a form of a textual (XML) data file. Logical structure of that file is really simple, and may be seen in Figure 3; it consists of several `<Vector>` elements, each one containing a record of less than one hundred decimal floating point numbers. Upon creation, a DEF file is stored in a *biometric data database* and (optionally) personal user identifier PID is generated.

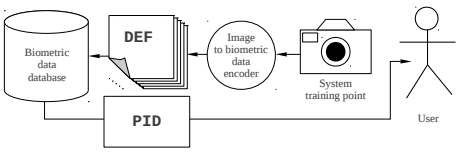


Fig. 4: FYEO Registration process

When a biometrically protected document is about to be sent to a registered user, the FYEO encryption process (shown in Figure 5) is started by the document originator, who picks the document from the *library of unsecured documents* (or simply creates a new one).

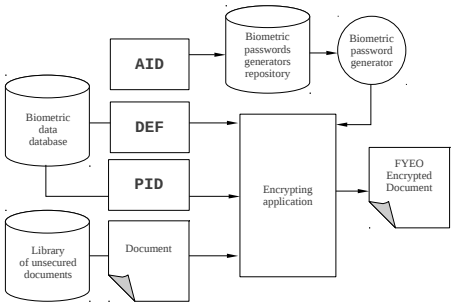


Fig. 5: FYEO encryption process

The respective recipient's DEF and PID are read from the *biometric data database*, and a specific password generation algorithm is chosen from the *biometric password generators repository*. A *biometric password generator* is an application that works in a plug-in mode, which enables usage of various algorithms – each one identified by its unique *algorithm identifier* AID. Given the size and diversity of the DEF file content, arbitrarily many password generation algorithms may be implemented and stored in the repository. Each time, however, only the algorithm whose status is currently set to *valid* (for example has not expired) may be chosen. The *encrypting application* encodes a document based on the password generated by the *biometric password generator* from the user's DEF data and (optionally) a personal identifier PID provided by the user. DEF and AID data may be added as PDF's internal objects or just appended to the document's file (what is the case of our prototype). After that the encrypted document may be delivered to any remote user as an email attachment or uploaded to the user's device.

The FYEO decryption process is performed on the client device, as shown in Figure 6.

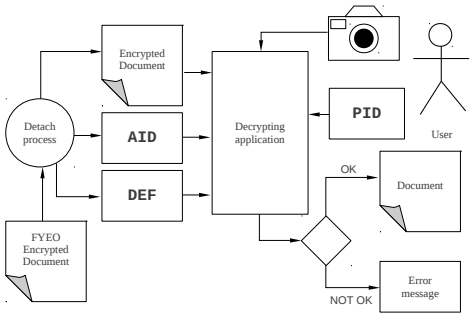


Fig. 6: FYEO decryption process (client device)

When the receiving user tries to open a document protected with a biometric password a *detach process* separates a biometric definition data DEF from the document, checks if a decoding plug-in with a proper AID is available and its state is valid. If these conditions are met the user is prompted to take a face shot. Thus obtained image is processed by the *decrypting application*, which compares data from the image with the biometric DEF definition separated from the received document. If the *biometric data classifier* embedded in the application decides that the image matches the received biometric definition, it calls its *biometric password generator* for the password to decrypt the encrypted PDF document. From now on it can be displayed with any PDF document viewer.

### 2.2.3 Combined biometric password and personal key

In order to make the authentication process more secure the FYEO encrypted document may ask the user for his/her personal PID key, assigned during the registration process shown in Figure 4. In such a case the key is passed to the *biometric password generator* shown in Figure 3, and used as an additional element in the encryption key generation process, in the fashion similar to the *salt* used in encryption processes.

## 2.3 Experiments

As mentioned before, exhaustive tests of the CF library have been performed during the SART-2 project to assess the optimal FAR/FRR ratio determining acceptable comfort levels of using the implemented algorithms in a general biometric application. Presenting them here is out of the scope of this paper. Instead we present results of several *ad hoc* experiments performed by us to demonstrate the possible attacks on the FYEO architecture. First we have prepared three training sets of photos shown in Figure 7.

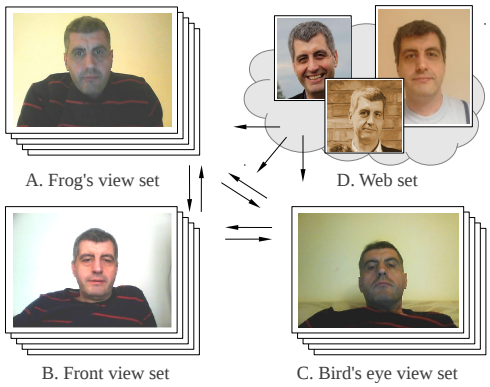


Fig. 7: Sets of photos used in experiments

They contain images of the same person, all recorded with the same camera built in the user's personal laptop; the respective sets of photos were taken when the person was standing and looking from above his laptop (A), sitting in front of it (B), and lying on his back and looking from below it (C). Photographs in respective sets A, B, and C were

very similar, as they had been taken during a short period of time during a regular use of the laptop by that person.

Next, three DEF files have been generated (see Figure 4), each one for a specific subset of A, B and C; these subsets formed *training sets*. For each training set a 32-character password has been generated with randomly selected password generators using the complete key-space (see Figure 5):

```
A: 8Lt%5VmW6Sj%2To:AGpX0Rv$6Uj%0HtX
B: BQvYBPj!8Vh(5Nk%7On(DSm)BOs:4Hk#
C: 5Nj:4SvZ1Ul)DPh!6Li:EIg(ESs:5Or!
```

For testing purposes we have also collected set D of photographs of the same person, which we were able to find on the Web. Note that photographs in D contain images of the person in various contexts (inclined, diverted, smiling, sober, etc.), also that person's age varies.

The remaining photos from A, B, C (not used in training sets), and the Web set D photos were used as *testing sets*. For each tested combination (marked respectively in Figure 7 with arrows) only three photos of the sitting user have been *false accepted* by FYEO trained with photos of the standing user. This was due to the camera angle, because the user was holding a laptop on his knees, so when taking them the angle was sometimes more like the frog's view than the front view. Each time FYEO was trained with respective subsets of A, B, or C, the remaining photos from the same set used as the testing set, have been *true accepted*. Moreover all D photos have always been *false rejected* by FYEO, if trained respectively with A, B, or C.

Note that rejection of an image of the same person, as the one for which a biometric system has been trained, is in general considered an *error*. However, Web based photos used as input should be *rejected* by FYEO, no matter what person they present. Therefore photos for the training set must be carefully selected – enabling true accept for testing photos of the user which are close to the ones of the training set, and false reject for those which are not. Our experiments indicate that manipulating the camera angle is quite effective in providing that.

Having these results in mind consider the following attacks on FYEO.

#### 2.3.1 Illegal interception

A document may be stolen or sent by mistake to an adversary, who does not know identity of its legitimate user, and is not aware of the FYEO protection. Most likely the attack would be applying *brute force* or *dictionary* methods on the password. The document is as safe as the password encrypted PDF document could be; given the fact that the password is full length and consisting of characters from the entire character plane, it would be rather unrealistic to crack it in any reasonable time (say less than hundreds of days).

#### 2.3.2 Device theft

If the user's personal device is stolen, an adversary may know identity of the user, and most likely will be able to use the FYEO client code. In such a case an impostor may look for the identified user photos on the Web, as well as in local directories of the stolen device. False reject rates estimated in our experiments indicate that photos of the same person taken at arbitrary camera angles can be effectively distinguished by CF algorithms used by FYEO, from the training set of photos, taken at a specific angle (not known to the adversary).

#### 2.3.3 Web-cam takeover

A more dangerous situation is when attackers can take over the user's web-cam, as photos taken by the latter may be close to the ones used for the training set. Getting access to the user's web-cam from outside is possible, as demonstrated in 2012 by the Weelsof virus targeting computer users in USA, Great Britain, Germany and Poland. This ransomeware, disguised as some fake law enforcement agency, displayed image of the user's face streamed from his/her connected web-cam as "recording" [15]. As a precaution FYEO users may be advised to take unusual positions, e.g. lying on back, when taking shots for the training set, and to remember what position to take when opening the document later. The attacker would not know for sure at what angle photos were created during a training session.

#### 2.3.4 Direct assault

The user may be physically forced by the attacker to open the document by looking at the camera of his/her personal device. To prevent that some additional functionality may be implemented in the FYEO software, eg. a special mimic or gesture which upon detection would destroy the document or device. More subtle face expressions related to various emotional states of the user (fear, anger, pain, etc.), informing the system about any unusual situation, can be detected by some of our algorithms.

#### 2.3.5 Hacked training point

Photos of the training set may be stolen if an adversary can get access to the FYEO system training point. Since it does not require any network connection, it should be off-line all the time, and the training photos destroyed as soon as DEF files are generated; they are not needed any longer and cannot be recovered from the DEF file content. One possibility is sabotage from inside the organization, e.g. unauthorized copying of files to the USB stick during the registration process, but this is the matter of a general security policy of the related organization using FYEO protected documents.

#### 2.3.6 Hacked server

Finally the organization server (see Figure 5) may be attacked at several points:

- *Biometric data database* does not contain any useful data unless user PIDs are kept in open text instead of a digested format.
- *Library of unsecured documents* may contain confidential data and must be protected by the existing security mechanisms of the server. An alternative is to create a document on the fly right before encrypting and sending it out.
- *Biometric passwords generators repository* may be useful to hackers only if after forcing the server's security mechanisms they would know AIDs. For more demanding security levels, a special password generator may be created on the fly, instead of using one from the repository

## 3. Conclusions

The purpose of our work reported in this paper has not been improving or replacing the existing security mechanisms in PDF. Our choice to implement FYEO biometric protection for PDF has been motivated by the encryption mechanisms built directly in the PDF document content, rather then "enveloped" around the document file, e.g. as is the case of ZIP files. Any FYEO protected PDF document is protected to the extent provided by the security handler of the revision level specified in the encryption dictionary of the original PDF document.

We have approached the problem of PDF security from the perspective of automatic generation of passwords from facial biometric data. Our method enables generation of passwords that may be arbitrarily long and complex, using the complete key-space, which do not have to be memorized or written down by users. In fact our FYEO approach subscribes to the concept of exploiting hard AI problems for security purposes, stated formally by the inventors of CAPTCHA [16]. In their terminology, a program delivering photo $i$ of the user attempting to access a protected document is *verifier* $V$, while the FYEO client side is *prover* $P$. $P$ receives from $V$ transformed image $t(i)$, and outputs label $\lambda(i)$, which in our case is a DEF file, identifying the user. Breaking in a document with a forged photo would require finding image transformation $i \rightarrow t(i)$, capable of making $P$ to calculate DEF that can eventually lead to document opening. This is a hard AI problem belonging to the $\mathcal{P}2$ family defined formally in [16]. Owing to the modular structure of the CF library, new algorithms making the system more resistant to impostors are planned to be added, to further complicate the problem of finding transformation $t(i)$. They will include in the first place liveness verification and emotion detection, but in the future may also include detection of gestures combined with faces, or specific visual tokens incorporated in the image.

Further extensions of the FYEO scheme may combine biometric passwords with personal keys, either built in the personal device – a FYEO document may be then decrypted only on a device with a specific ID, or at specific geographical location identified by the user's device GPS receiver, and so on, or in a form of physical tokens – USB memory sticks or eID cards plugged-in the device prior to the FYEO document opening attempt.

## Acknowledgment

This work was supported by the National Science Centre grants no. N N516 367936 and DEC1-2011/01/B/ST6/06500.

## References

[1] MeNaID project home page, "Methods and tools of future document engineering." [Online]. Available: www.menaid.org.pl
[2] M. Godlewska and B. Wiszniewski, "Distributed MIND – a new processing model based on mobile interactive documents," in *Proc. 8th Int. Conf. PPAM 2009*, ser. LNCS, vol. 6068. Springer, 2010, pp. 244–249.
[3] J. Siciarek and B. Wiszniewski, "IODA - an interactive open document architecture," *Procedia CS*, vol. 4, pp. 668–677, 2011.
[4] SART-2 project home page, "Biometric security system for mobile workstations," 2012. [Online]. Available: http://sart2.eti.pg.gda.pl/
[5] M. Smiatacz and K. Przybycien, "A framework for training and testing of complex pattern recognition systems," in *Proc. IEEE Conf. on Signal and Image Processing Applications (ICSIPA)*, 2011, pp. 198–203.
[6] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comp. Vision*, vol. 57, no. 2, pp. 137–154, 2004.
[7] X. Tan and B. Triggs, "Enhanced local texture feature sets for face recognition under difficult lighting conditions," *IEEE Trans. Image Processing*, vol. 19, no. 6, pp. 1635–1650, 2010.
[8] T. Sim, S. Baker, and M. Bsat, "The CMU pose, illumination, and expression (PIE) database," in *Proc. 5th Int. Conf. on Automatic Face and Gesture Recognition*, 2002, pp. 53–58.
[9] M. Smiatacz, "Image normalization method for face identification under difficult lighting conditions," in *Proc. 11th IEEE Int. Conf. on Information Sciences, Signal Processing and their Applications (ISSPA)*, 2012, pp. 204–209.
[10] M. Szwoch and P. Pieniążek, "Eye blink based detection of liveness in biometric authentication systems using conditional random fields," *LNCS*, vol. 7594, pp. 669–676, 2012.
[11] M. Smiatacz, "Liveness measurements using optical flow for biometric person authentication," *Metrology and Measurement Systems*, vol. 19, no. 2, pp. 257–268, 2012.
[12] G. Donato, M. S. Bartlett, J. C. Hager, P. Ekman, and T. J. Sejnowski, "Classifying facial actions," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, no. 10, pp. 974–989, 1999.
[13] *Document management – Portable document format – Part 1: PDF 1.7*, Adobe Systems Inc., 2008. [Online]. Available: http://www.adobe.com/devnet/pdf/pdf_reference.html
[14] J. Steube, "Hashcat – advanced password recovery," March 7, 2013. [Online]. Available: http://hashcat.net/hashcat/
[15] T. Meskauskas, "The FBI Federal Bureau of Investigation screen locker," July 16, 2012. [Online]. Available: http://www.pcrisk.com/removal-guides/6765-remove-the-fbi-federal-bureau-of-investigation-screen-locker
[16] L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford, "CAPTCHA: using hard AI problems for security," in *Proc. 22nd Int. Conf. on Theory and Applications of Cryptographic Techniques EUROCRYPT'03*. Springer, 2003, pp. 294–311.