Poisoning adaptive biometric systems

Battista Biggio, Giorgio Fumera, Fabio Roli, and Luca Didaci

Department of Electrical and Electronic Engineering, University of Cagliari
Piazza d'Armi 09123, Cagliari, Italy
{battista.biggio, fumera, roli, luca.didaci }@diee.unica.it
WWW home page: http://prag.diee.unica.it/

Abstract. Adaptive biometric recognition systems have been proposed to deal with natural changes of the clients' biometric traits due to multiple factors, like *aging*. However, their adaptability to changes may be exploited by an attacker to compromise the stored templates, either to impersonate a specific client, or to deny access to him. In this paper we show how a carefully designed attack may gradually poison the template gallery of some users, and successfully mislead a simple PCA-based face verification system that performs self-update.

Keywords: Biometric recognition, Adaptive biometric systems, Template self-update, Principal component analysis, Poisoning attack.

1 Introduction

Adaptive biometric recognition systems have been proposed to deal with changes of the clients' biometric traits over time, like aging. Biometric data acquired over time during system operation can be exploited to account for the natural temporal variations of biometric traits. One of the proposed approaches, inspired by semi-supervised learning techniques, is template self-update. It consists of periodically updating the template gallery of a user, using samples assigned with high confidence to the corresponding identity during operation. Adaptation may allow a biometric system to maintain a good performance over time. However, an *attacker* may exploit it to compromise the stored templates, either to impersonate a specific client or to deny access to him, violating system security.

In this paper we present a preliminary investigation on how to exploit the above discussed vulnerability in the context of adaptive biometric systems, using as a case study a simple PCA-based face verification system that performs self-update. We show that an attacker can submit a carefully designed set of fake faces to the camera while claiming the identity of another user (i.e., the *victim*), to gradually compromise the stored templates of the victim. The fake faces can be obtained by printing a face image on paper. This is a well-known procedure in the literature of *spoofing* of biometric traits (see, e.g., [2]). The goal of the attacker is to eventually be able to impersonate the victim without presenting any fake face to the sensor, i.e., to include at least one of her templates into the victim's gallery.

We derive the optimal attack, i.e., the one that minimizes the number of fake faces to be submitted to the sensor, under two distinct template update policies. To this end, we exploit the results reported in [4] about poisoning attacks against a different, but related application (online anomaly detection). Our results show that an attacker may effectively compromise the system with relatively small effort, i.e., by submitting a few, carefully designed fake faces. We also highlight a trade-off between the ability of a system to adapt to changes, and its security.

In Sect. 2 we summarize background concepts on adaptive biometric systems. Poisoning attacks and our application example are described in Sect. 3 and 4, respectively. Conclusions are drawn in Sect. 5.

2 Adaptive biometric recognition systems

One of the issues that affect the performance of biometric systems in real operational scenarios is that biometric data can exhibit a large intra-class variability due to multiple factors, like illumination changes, pose variations and aging. This can make the templates stored for each client during enrollment not representative of the biometric traits submitted during verification (or identification) [10]. In particular, it is very difficult to deal with *temporal* changes of biometric patterns, like the ones due to aging. To this end, the exploitation of biometric data acquired over time during system operation has recently been proposed [3, 8, 7]. The reason is that such data stream naturally contains temporal variations of the considered biometric trait, which may allow one to implement *adaptive* systems that improve with use. In the following, we focus on the template self-update technique, that will be considered in the rest of this work.

Template self-update. Template self-update is a semi-supervised learning technique that can be easily implemented in many biometric recognition systems, to enable adaptation to temporal changes. It consists of updating the stored templates of each enrolled client over time, exploiting unlabelled biometric data acquired during system operation [7]. In this work we consider a simple biometric verification system that stores one template for each client, computed by averaging the set of n enrolled images of the same client. It will thus be referred to as *centroid*. Denoting the feature vectors of the enrolled images of a given client c as $\{\mathbf{x}_{c,1}, \dots, \mathbf{x}_{c,n}\}$, their centroid is $\mathbf{x}_c = \frac{1}{n}\sum_{k=1}^{n} \mathbf{x}_{c,k}$. During verification, a user submits a sample $\mathbf{x}$ and claims an identity c. A matching score $s(\mathbf{x}, \mathbf{x}_c)$ is then computed, e.g.:

$$s(\mathbf{x}, \mathbf{x}_c) = 1/(1 + \|\mathbf{x} - \mathbf{x}_c\|) \ , \tag{1}$$

where $\|\cdot\|$ is the Euclidean distance. The user is accepted as genuine, if $s(\mathbf{x}, \mathbf{x}_c) \geq t_c$, otherwise it is rejected as an impostor, where t_c is a predefined, client-dependent acceptance threshold. Template self-update can be implemented by updating $\mathbf{x}_c$ using $\mathbf{x}$, if $s(\mathbf{x}, \mathbf{x}_c) \geq \theta_c$, where θ_c is an update threshold, usually more conservative, i.e., $\theta_c > t_c$. The centroid $\mathbf{x}_c$ can be updated to $\mathbf{x}'_c$ according to different policies, more or less adaptive. We will consider two policies discussed in [4], which can be expressed as:

$$\mathbf{x}'_c = \mathbf{x}_c + (\mathbf{x} - \mathbf{x}_c)/n \ . \tag{2}$$

The *infinite window* policy updates $\mathbf{x}_c$ without discarding any of the past n samples [4, 6]. Thus, n increases by 1 *before* each update, and the impact of new samples reduces as n grows. A more adaptive policy is *finite window (average-out)*, that discards the current centroid at each iteration, and keeps n fixed to its initial value.

3 Poisoning attacks

Biometric recognition is an example of the use of *machine learning in adversarial environments*, in which a human "adversary" can be interested in subverting a recognition system, e.g., to impersonate a given client [5, 1]. In particular, if the adversary has some degree of control on training data (e.g., in scenarios like template self-update), she may "contaminate" it by adding carefully designed attack samples. This attack, known as *poisoning*, has been investigated in [4, 6] for online anomaly detection tasks. Since their results apply also to biometric template self-update, we summarize them below.

In template self-update, a poisoning attack exploits adaptation to gradually compromise the template $\mathbf{x}_c$ of the targeted client, until it is replaced by a sample $\mathbf{x}_a$ chosen by the attacker. To this end, the adversary may be required to iteratively submit to the system a carefully designed sequence of attack samples. We consider the case when the attacker aims to gain access with the identity of user c without using any fake trait. In this case, $\mathbf{x}_a$ must be a representative sample of the attacker's biometric trait. The type and number of attack samples depend on the template update policy, and on the capability and knowledge of the attacker. The analysis of [4, 6] was made under the worst-case assumption that the attacker perfectly knows the targeted system, which is typical in security problems. In our case, this amounts to knowing the feature vector representation of samples, the initial template gallery of the targeted client and the template updating policy, the matching score function $s(\cdot, \cdot)$, and the thresholds t_c and θ_c of the victim. The optimal attack can be derived, in terms of the minimum number of attack samples required to replace $\mathbf{x}_c$ with $\mathbf{x}_a$, as well as a lower bound on such number, depending on the update policy.

The optimal poisoning attack against the policies mentioned in Sect. 2 is depicted in Fig. 1. At iteration i, it amounts to place the attack sample $\mathbf{x}_a^{(i)}$ on the line joining $\mathbf{x}_a$ and the initial centroid $\mathbf{x}_c$, in the so-called *attack direction* $\mathbf{a} = \frac{\mathbf{x}_a - \mathbf{x}_c}{\|\mathbf{x}_a - \mathbf{x}_c\|}$, at the maximum distance from the current centroid $\mathbf{x}_c^{(i)}$ that satisfies the update condition $s(\mathbf{x}_a^{(i)}, \mathbf{x}_c^{(i)}) \geq \theta_c$. Given the matching score of Eq. 1, this distance is $d_c(\theta_c) = \|\mathbf{x}_a^{(i)} - \mathbf{x}_c^{(i)}\| = 1/\theta_c - 1$. This leads to: $\mathbf{x}_a^{(i)} = \mathbf{x}_c^{(i)} + d_c \cdot \mathbf{a}$.

The minimum number of attack samples needed to replace $\mathbf{x}_c$ with $\mathbf{x}_a$, for the *infinite* and *finite window* policies, is respectively lower bounded by:

$$n \left[\exp\left(\|\mathbf{x}_a - \mathbf{x}_c\|/d_c\right) - 1\right], \qquad n \left(\|\mathbf{x}_a - \mathbf{x}_c\|/d_c\right) \ . \tag{3}$$

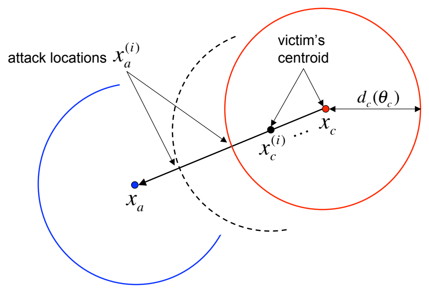

Fig. 1. Illustration of a poisoning attack, similar to [4].

It is worth noting that: (i) in both cases the number of attack samples scales linearly with the initial number of averaged samples n; (ii) in the *infinite window* case, the number of attack samples increases exponentially as $\|\mathbf{x}_a - \mathbf{x}_c\|$ grows; (iii) in the *finite window* case, such number scales linearly with $\|\mathbf{x}_a - \mathbf{x}_c\|$. Therefore, although more adaptive to changes, the latter policy may be misled by a poisoning attack with a significantly lower number of attack samples. This quantifies the intuitive trade-off between the ability of the system to adapt to changes and its *security* to poisoning attacks. To better characterize this trade-off in the context of biometric systems, one should also evaluate the probability for an attacker (and the victim) to be accepted as the targeted victim, as the attack proceeds. However, this can be only done empirically, as shown in the next section.

4 Application example

In this section we describe the case study, related to PCA-based face verification, that we used to investigate the vulnerability of template self-update techniques.

PCA-based face verification. The standard PCA-based face recognition method works as follows [9, 11]. During *enrollment*, a set of face images is acquired for each user and pre-processed (e.g., the background is removed by applying a specific mask to each image, and face images are normalized to have the same size and eye position). Each image is then stored as a column vector of d pixels to constitute the training set $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\} \in R^{d \times n}$, and the PCA is applied as follows. (i) The average face image and the covariance matrix are respectively computed as $\mathbf{z}_\mu = \frac{1}{n}\sum_{k=1}^{n} \mathbf{z}_k$, and $C = (Z - \mathbf{z}_\mu)(Z - \mathbf{z}_\mu)^{\mathrm{T}}$. (ii) The eigenvalues and eigenvectors of C can be more efficiently computed from the matrix $K = (Z - \mathbf{z}_\mu)^{\mathrm{T}}(Z - \mathbf{z}_\mu)$ instead of C, as explained in [11, 9]. Usually only a subset of them (those associated to the highest eigenvalues) is retained for computational efficiency. (iii) Samples in Z can be now projected onto the eigenspace as $\mathbf{x}_i = V^{\mathrm{t}}(\mathbf{z}_i - \mathbf{z}_\mu)$, $i = 1, \ldots, n$, where V is the matrix of the eigenvectors (one per column). (v) For each user c, a face template $\mathbf{x}_c$ is stored. Such template is often computed as the mean (or centroid) of the projected faces of

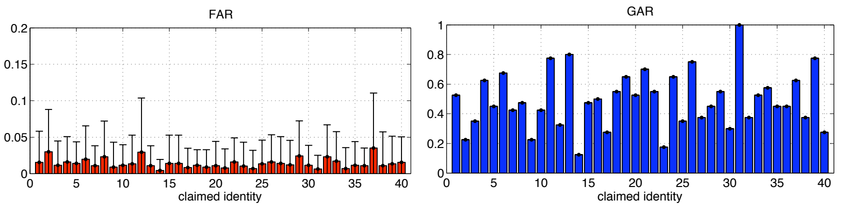

Fig. 2. FAR and GAR for each client under no attack. The FAR is averaged over all possible attackers. Standard deviation is also shown as error bars.

that user. During *verification*, the input image $\mathbf{z}$ is pre-processed and projected onto the eigenspace as $\mathbf{x} = V^{\mathrm{t}}(\mathbf{z} - \mathbf{z}_\mu)$. Then, it is compared to the centroid of the claimed identity through the matching score $s(\mathbf{x}, \mathbf{x}_c)$, either to accept or reject the user (see Sect. 2).[1] Template self-update was implemented as explained in Sect. 2, using the two update policies of Sect. 3. In our experiments we set the initial template gallery size to $n = 10$. We also assume that the PCA projection is not updated during verification, since it is too computationally expensive [7].

Data set. We collected a data set consisting of 40 different clients with 60 images each, for a total of 2,400 face images. The face images of each client were collected into two sessions, using a commercial webcam, with a time interval of about two weeks between them, under different lighting conditions and facial expressions. This induced a high intra-class variability of the face images, which makes face recognition particularly challenging. The data set is available under request to the authors, and it was also used in [2].

Experimental setup. We split the face images as follows. We randomly selected 10 images for each client as training data, to compute the PCA eigenvectors and the clients' centroids. A further set of 10 images for each client was used as validation data, to tune the acceptance threshold t_c and update threshold θ_c for each client. We set θ_c by computing the 0% FAR operational point for the corresponding client, and t_c to a less conservative value, namely, at the 1% FAR operational point. The remaining 40 images per client were retained as testing set. We observed that randomly choosing different data splits do not substantially affect our results. For the sake of ease of interpretation, we thus chose not to average them on different data splits.

Performance under no attack. We first computed the performance of the considered face verification system on the testing set, when no attack is considered, in terms of the Genuine Acceptance Rate (GAR) and False Acceptance Rate (FAR), namely, the fraction of clients correctly verified and of impostors wrongly accepted as genuine clients. For each client (i.e., claimed identity), the

[1] Note that more than one template per user may be also used, to better capture the high intra-class variability typical of biometric images. This would however slightly complicate the verification process, and the consequent poisoning attack; thus, we only consider here the simplest case in which only one centroid per user is stored.

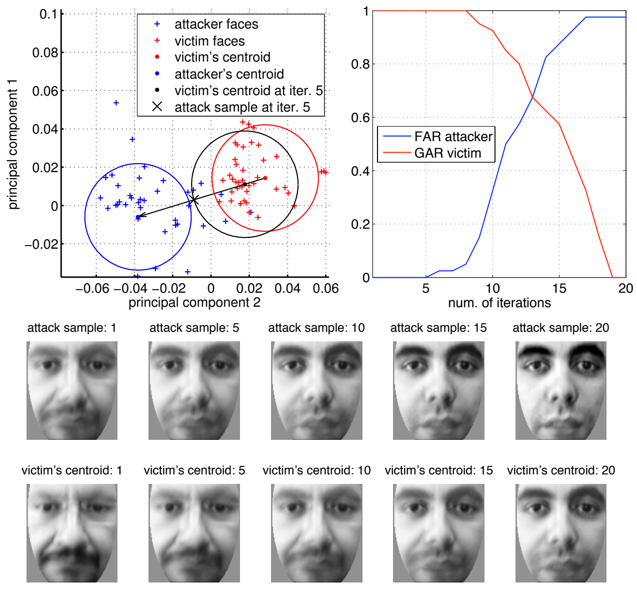

Fig. 3. Poisoning attack under finite window (average-out) update policy. *Top-left plot:* illustration of the attack in the space spanned by the first two principal components. *Top-right plot:* Variation of the attacker's FAR and victim's GAR during the attack progress. *Bottom plot:* attack samples and victim's centroids during the attack progress, at iterations 1, 5, 10, 15, 20.

corresponding GAR and FAR are shown in Fig. 2. The FAR of each client was estimated as the average FAR of the other 39 users, considered as attackers. As expected from the choice of t_c, the average FAR is around 1% for most of the clients. A rather low GAR is attained for several users (lower than 50%), due to the high-intra class variability. Further, we observed that for most clients no update actually occurred, due to the very conservative choice of θ_c.

Poisoning under the finite window (average-out) update policy. We implemented the poisoning attack as described in Sect. 3. We simulated the simplest scenario in which the template gallery of the targeted client (victim) is updated by a sequence of attack samples only, in a given period of time.[2]

In Fig. 3, we report the results attained when considering a specific attacker (user 13) and victim (user 31). The attack progress is depicted in the top-left plot, where it can be noted how the victim's centroid drifts toward the attacker's centroid. In particular, we report the initial victim's and attacker's centroids, and the drifted victim's centroid after 5 iterations (i.e., after submitting 5 at-

[2] The scenario when the gallery is updated with interleaved attack and genuine samples (i.e., samples coming from genuine verification attempts by the targeted client) can be however investigated in a similar manner, as done in [4].

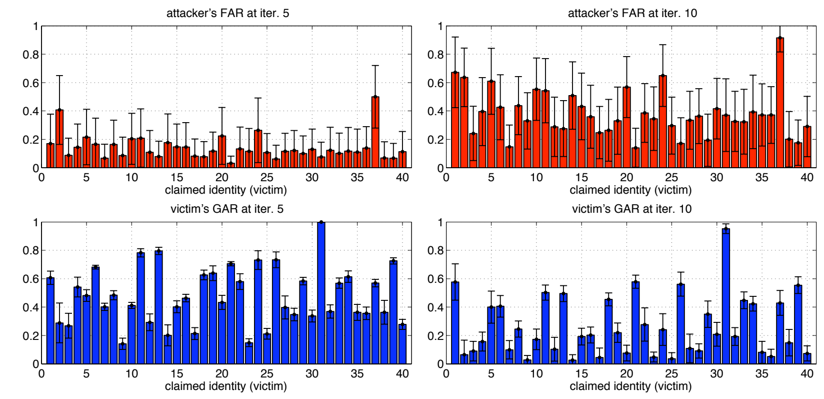

Fig. 4. Poisoning attack under the finite window (average-out) update policy.

tack samples). The top-right plot shows how the victim's GAR decreases, and, simultaneously, the FAR relative to the attacker identity increases, as a function of the number of attack iterations. Although the number of iterations required to replace the victim's centroid with the attacker's centroid is 20, the attacker's FAR raises quite quickly, being equal to 40% after only 10 iterations. The bottom plot shows some attack samples, and the corresponding change in the victim's centroid. Note how the initial victim's face (victim's centroid at iteration 1) is eventually replaced by the attacker's face (victim's centroid at iteration 20).

Fig. 4, and Fig. 5 (left plot) summarize the results obtained considering all possible pairs of attacker and victim ($39 \times 40 = 1560$). The latter depicts the average number of iterations and the standard deviation, over the 39 possible attackers, required to replace the victim's centroid with the attacker's one. Note that most of the attacks are successfully completed after 10 to 20 iterations.

Since in many cases it is not realistic for an attacker to perform more than 10 attempts without being caught, we focused on the GAR of each victim, and the FAR relative to the corresponding attacker, after 5 and 10 iterations, which are reported in Fig. 4. As in the previous case, for each victim we average the GAR and FAR with respect to all possible 39 attackers. In other words, the FAR represents the probability that a randomly chosen attacker cracks a specific victim account. Similarly, the GAR represents the probability of a specific victim being correctly accepted as a genuine user, under a poisoning attack carried out by a randomly chosen attacker. It can be seen that the FAR is relatively high even at the early stages of the attack: it ranges from 10% to 20% after only 5 iterations, and approaches 50% for most of the targeted victims after 10 iterations. The GAR remains instead almost the same after 5 iterations, but significantly decreases after 10 iterations for most of the victims. This means that an attacker may significantly increase the chance of being accepted after few iterations (e.g., from 1% FAR to 10% FAR with just 5 iterations) without causing a substantial denial of access to the victim. Lastly, note that after 10

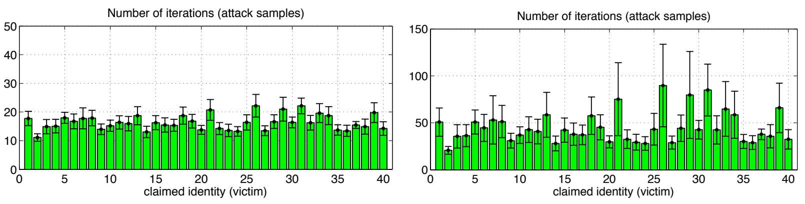

Fig. 5. Number of attack samples required to replace the victim's centroid with the attacker's centroid. For each victim, we reported the number of iterations averaged over all possible attackers. Standard deviation is also shown as error bars. *Left plot:* finite window (average-out) update policy. *Right plot:* infinite window update policy.

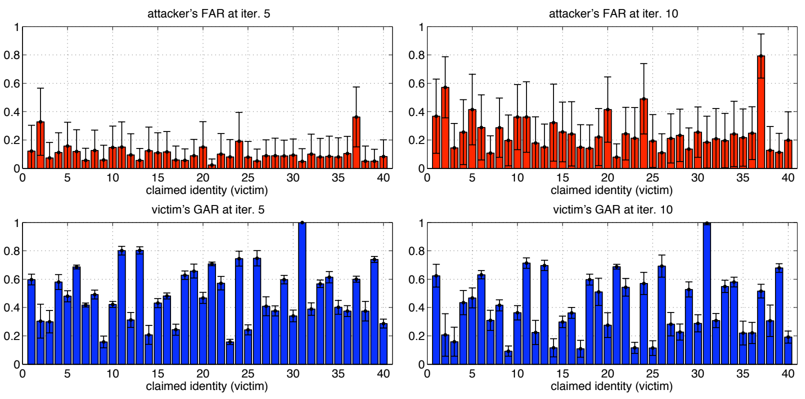

Fig. 6. Poisoning attack under the infinite window update policy.

iterations, the FAR is much higher than the GAR, although most of the poisoning attacks are not complete at this stage.

Poisoning under the infinite window update policy. Poisoning is much harder under this policy, since it requires an exponential number of attack samples with respect to the relative displacement (see Eq. 3). We report the evaluation involving all pairs of attacker and victim, as above, in Fig. 6, and Fig. 5 (right plot). As expected, the latter plot shows that the number of iterations required to complete a poisoning attack is much higher in this case, and its effectiveness (in terms of FAR at the same number of iterations) is lower. However, Fig. 6 shows that the increase in FAR is still significant, even after few iterations.

Finally, we repeated the experiments reducing the initial number of templates per client to $n = 5$. As predicted by Eq. 3, the number of attack iterations scaled linearly with n for both policies, without substantially changing the attack effectiveness in terms of FAR and GAR. In particular, almost the same values were attained after half of the iterations.

5 Conclusions and future work

In this work, we demonstrated that adaptive biometric recognition systems can be vulnerable to poisoning attacks, namely, carefully designed attacks that exploit system adaptation. Such attacks can significantly violate system security from the early stages, i.e., with few attack iterations. To our knowledge, this is the first time that such vulnerability is highlighted in the context of biometric adaptive systems. Further, we observed that more adaptive update policies (e.g., finite window), which may be more beneficial in the standard scenario without attacks, can be more vulnerable to poisoning than less adaptive policies (e.g., infinite window). This highlights that a trade-off between security and ease of adaptation is required in adaptive biometric systems, and that it should be investigated more in detail in future work; e.g., considering different adaptive systems and more update policies.

Acknowledgment. This work has been partly supported by the project CRP-18293 funded by Regione Autonoma della Sardegna, L.R. 7/2007, Bando 2009.

References

1. Barreno, M., Nelson, B., Sears, R., Joseph, A.D., Tygar, J.D.: Can machine learning be secure? In: Proc. Symp. on Information, Computer and Comm. Sec. (ASIACCS). pp. 16–25. ACM, (2006)
2. Biggio, B., Akhtar, Z., Fumera, G., Marcialis, G.L., Roli, F.: Security evaluation of biometric authentication systems under real spoofing attacks. IET Biometrics 1(1), 11–24 (2012)
3. Jiang, X., Ser, W.: Online fingerprint template improvement. IEEE Trans. Pattern Analysis and Machine Intell. 24(8), 1121 – 1126 (2002)
4. Kloft, M., Laskov, P.: Online anomaly detection under adversarial impact. In: Proc. 13th Int'l Conf. on AI and Statistics (AISTATS). pp. 405–412 (2010)
5. Laskov, P., Lippmann, R.: Machine learning in adversarial environments. Machine Learning 81, 115–119 (2010).
6. Nelson, B., Joseph, A.D.: Bounding an attack's complexity for a simple learning model. In: Proc. 1st Workshop on Tackling Computer Systems Problems with ML Techniques (SysML) (2006)
7. Roli, F., Marcialis, G.L.: Semi-supervised PCA-based face recognition using self-training. In: Proc. Joint IAPR Int'l Conf. on Structural, Syntactic, and Statistical Pattern Rec. (S+SSPR). pp. 560–568. Springer-Verlag, Berlin, Heidelberg (2006)
8. Ryu, C., Kim, H., Jain, A.K.: Template adaptation based fingerprint verification. In: Proc. 18th Int'l Conf. Pattern Rec. vol. 04. pp. 582–585. IEEE CS, (2006)
9. Turk, M., Pentland, A.: Eigenfaces for recognition. J. Cogn. Neuroscience 3(1), 71–86 (1991)
10. Uludag, U., Ross, A., Jain, A.K.: Biometric template selection and update: a case study in fingerprints. Pattern Recognition 37(7), 1533–1542 (2004)
11. Yambor, W.S.: Analysis of PCA-based and Fisher discriminant-based image recognition algorithms. Technical Report, Colorado State University (2000)