Security Evaluation of Pattern Classifiers under Attack

Battista Biggio, *Member, IEEE,* Giorgio Fumera, *Member, IEEE,* Fabio Roli, *Fellow, IEEE*

Abstract—Pattern classification systems are commonly used in *adversarial* applications, like biometric authentication, network intrusion detection, and spam filtering, in which data can be purposely manipulated by humans to undermine their operation. As this adversarial scenario is not taken into account by classical design methods, pattern classification systems may exhibit vulnerabilities, whose exploitation may severely affect their performance, and consequently limit their practical utility. Extending pattern classification theory and design methods to adversarial settings is thus a novel and very relevant research direction, which has not yet been pursued in a systematic way. In this paper, we address one of the main open issues: evaluating at design phase the *security* of pattern classifiers, namely, the performance degradation under potential attacks they may incur during operation. We propose a framework for *empirical* evaluation of classifier security that formalizes and generalizes the main ideas proposed in the literature, and give examples of its use in three real applications. Reported results show that security evaluation can provide a more complete understanding of the classifier's behavior in adversarial environments, and lead to better design choices.

Index Terms—Pattern classification, adversarial classification, performance evaluation, security evaluation, robustness evaluation.

◆

1 INTRODUCTION

Pattern classification systems based on machine learning algorithms are commonly used in security-related applications like biometric authentication, network intrusion detection, and spam filtering, to discriminate between a "legitimate" and a "malicious" pattern class (e.g., legitimate and spam emails). Contrary to traditional ones, these applications have an intrinsic adversarial nature since the input data can be purposely manipulated by an intelligent and adaptive *adversary* to undermine classifier operation. This often gives rise to an arms race between the adversary and the classifier designer. Well known examples of attacks against pattern classifiers are: submitting a fake biometric trait to a biometric authentication system (*spoofing* attack) [1], [2]; modifying network packets belonging to intrusive traffic to evade intrusion detection systems [3]; manipulating the content of spam emails to get them past spam filters (e.g., by misspelling common spam words to avoid their detection) [4]–[6]. Adversarial scenarios can also occur in intelligent data analysis [7] and information retrieval [8]; e.g., a malicious webmaster may manipulate search engine rankings to artificially promote her[1] web site.

It is now acknowledged that, since pattern classification systems based on classical theory and design

methods [9] do not take into account adversarial settings, they exhibit vulnerabilities to several potential attacks, allowing adversaries to undermine their effectiveness [6], [10]–[14]. A systematic and unified treatment of this issue is thus needed to allow the trusted adoption of pattern classifiers in adversarial environments, starting from the theoretical foundations up to novel design methods, extending the classical design cycle of [9]. In particular, three main open issues can be identified: (i) analyzing the vulnerabilities of classification algorithms, and the corresponding attacks [11], [14], [15]; (ii) developing novel methods to assess classifier security against these attacks, which is not possible using classical performance evaluation methods [6], [12], [16], [17]; (iii) developing novel design methods to guarantee classifier security in adversarial environments [1], [6], [10].

Although this emerging field is attracting growing interest [13], [18], [19], the above issues have only been sparsely addressed under different perspectives and to a limited extent. Most of the work has focused on application-specific issues related to spam filtering and network intrusion detection, e.g., [3]–[6], [10], while only a few theoretical models of adversarial classification problems have been proposed in the machine learning literature [10], [14], [15]; however, they do not yet provide practical guidelines and tools for designers of pattern recognition systems.

Besides introducing these issues to the pattern recognition research community, in this work we address issues (i) and (ii) above by developing a framework for the *empirical* evaluation of classifier security at design phase that extends the model selection and performance evaluation steps of the classical design cycle of [9].

In Sect. 2 we summarize previous work, and point out three main ideas that emerge from it. We then formalize

The authors are with the Department of Electrical and Electronic Engineering, University of Cagliari, Piazza d'Armi, 09123 Cagliari, Italy
Battista Biggio: e-mail battista.biggio@diee.unica.it, phone +39 070 675 5776
Giorgio Fumera: e-mail fumera@diee.unica.it, phone +39 070 675 5754
Fabio Roli (corresponding author): e-mail roli@diee.unica.it, phone +39 070 675 5779, fax (shared) +39 070 675 5782

1. The adversary is hereafter referred to as feminine due to the popular interpretation as "Eve" or "Carol" in cryptography and computer security.

and generalize them in our framework (Sect. 3). First, to pursue security in the context of an arms race it is not sufficient to *react* to observed attacks, but it is also necessary to *proactively anticipate* the adversary by *predicting* the most relevant, potential attacks through a what-if analysis; this allows one to develop suitable countermeasures *before* the attack actually occurs, according to the principle of *security by design*. Second, to provide practical guidelines for simulating realistic attack scenarios, we define a general model of the adversary, in terms of her goal, knowledge, and capability, which encompasses and generalizes models proposed in previous work. Third, since the presence of carefully targeted attacks may affect the distribution of training and testing data separately, we propose a model of the data distribution that can formally characterize this behavior, and that allows us to take into account a large number of potential attacks; we also propose an algorithm for the generation of training and testing sets to be used for security evaluation, which can naturally accommodate application-specific and heuristic techniques for simulating attacks.

In Sect. 4 we give three concrete examples of applications of our framework in spam filtering, biometric authentication, and network intrusion detection. In Sect. 5, we discuss how the classical design cycle of pattern classifiers should be revised to take security into account. Finally, in Sect. 6, we summarize our contributions, the limitations of our framework, and some open issues.

2 BACKGROUND AND PREVIOUS WORK

Here we review previous work, highlighting the concepts that will be exploited in our framework.

2.1 A taxonomy of attacks against pattern classifiers

A taxonomy of potential attacks against pattern classifiers was proposed in [11], [15], and subsequently extended in [14]. We will exploit it in our framework, as part of the definition of attack scenarios. The taxonomy is based on two main features: the kind of *influence* of attacks on the classifier, and the kind of *security violation* they cause. The influence can be either **causative**, if it undermines the learning algorithm to cause subsequent misclassifications; or **exploratory**, if it exploits knowledge of the trained classifier to cause misclassifications, without affecting the learning algorithm. Thus, causative attacks may influence both training and testing data, or only training data, whereas exploratory attacks affect only testing data. The security violation can be an **integrity** violation, if it allows the adversary to access the service or resource protected by the classifier; an **availability** violation, if it denies legitimate users access to it; or a **privacy** violation, if it allows the adversary to obtain confidential information from the classifier. Integrity violations result in misclassifying malicious samples as legitimate, while availability violations can also cause legitimate samples to be misclassified as malicious. A third feature of the taxonomy is the *specificity*

of an attack, that ranges from **targeted** to **indiscriminate**, depending on whether the attack focuses on a single or few specific samples (e.g., a specific spam email misclassified as legitimate), or on a wider set of samples.

2.2 Limitations of classical performance evaluation methods in adversarial classification

Classical performance evaluation methods, like k-fold cross validation and bootstrapping, aim to estimate the performance that a classifier will exhibit during *operation*, by using data $\mathcal{D}$ collected during classifier *design*.[2] These methods are based on the *stationarity* assumption that the data seen during operation follow the same distribution as $\mathcal{D}$. Accordingly, they resample $\mathcal{D}$ to construct one or more pairs of training and testing sets that ideally follow the same distribution as $\mathcal{D}$ [9]. However, the presence of an intelligent and adaptive adversary makes the classification problem highly non-stationary, and makes it difficult to predict how many and which kinds of attacks a classifier will be subject to during operation, that is, how the data distribution will change. In particular, the testing data processed by the trained classifier can be affected by both exploratory and causative attacks, while the training data can only be affected by causative attacks, if the classifier is retrained online [11], [14], [15].[3] In both cases, during operation, testing data may follow a *different* distribution than that of training data, when the classifier is under attack. Therefore, security evaluation can not be carried out according to the classical paradigm of performance evaluation.[4]

2.3 Arms race and security by design

Security problems often lead to a "reactive" arms race between the adversary and the classifier designer. At each step, the adversary analyzes the classifier defenses, and develops an attack strategy to overcome them. The designer reacts by analyzing the novel attack samples, and, if required, updates the classifier; typically, by retraining it on the new collected samples, and/or adding features that can detect the novel attacks (see Fig. 1, *left*). Examples of this arms race can be observed in spam filtering and malware detection, where it has led to a

---

2. By *design* we refer to the classical steps of [9], which include feature extraction, model selection, classifier training and performance evaluation (testing). *Operation* denotes instead the phase which starts when the deployed classifier is set to operate in a real environment.

3. "Training" data refers both to the data used by the learning algorithm during classifier design, coming from $\mathcal{D}$, and to the data collected during operation to retrain the classifier through online learning algorithms. "Testing" data refers both to the data drawn from $\mathcal{D}$ to evaluate classifier performance during design, and to the data classified during operation. The meaning will be clear from the context.

4. Classical performance evaluation is generally not suitable for non-stationary, time-varying environments, where changes in the data distribution can be hardly predicted; e.g., in the case of *concept drift* [20]. Further, the evaluation process has a different goal in this case, i.e., to assess whether the classifier can "recover" quickly *after* a change has occurred in the data distribution. Instead, security evaluation aims at identifying the most relevant attacks and threats that should be countered *before* deploying the classifier (see Sect. 2.3).

Fig. 1. A conceptual representation of the *arms race* in adversarial classification. *Left:* the classical "reactive" arms race. The designer reacts to the attack by analyzing the attack's effects and developing countermeasures. *Right:* the "proactive" arms race advocated in this paper. The designer tries to anticipate the adversary by simulating potential attacks, evaluating their effects, and developing countermeasures if necessary.

considerable increase in the variability and sophistication of attacks and countermeasures.

To secure a system, a common approach used in engineering and cryptography is *security by obscurity*, that relies on keeping secret some of the system details to the adversary. In contrast, the paradigm of *security by design* advocates that systems should be designed from the ground-up to be secure, without assuming that the adversary may ever find out some important system details. Accordingly, the system designer should *anticipate* the adversary by simulating a "proactive" arms race to (i) figure out the most relevant threats and attacks, and (ii) devise proper countermeasures, *before* deploying the classifier (see Fig. 1, *right*). This paradigm typically improves security by delaying each step of the "reactive" arms race, as it requires the adversary to spend a greater effort (time, skills, and resources) to find and exploit vulnerabilities. System security should thus be guaranteed for a longer time, with less frequent supervision or human intervention.

The goal of security evaluation is to address issue (i) above, i.e., to *simulate* a number of realistic attack scenarios that may be incurred during operation, and to assess the impact of the corresponding attacks on the targeted classifier to highlight the most critical vulnerabilities. This amounts to performing a *what-if* analysis [21], which is a common practice in security. This approach has been implicitly followed in several previous works (see Sect. 2.4), but never formalized within a general framework for the empirical evaluation of classifier security. Although security evaluation may also suggest specific countermeasures, the design of *secure* classifiers, i.e., issue (ii) above, remains a distinct open problem.

2.4 Previous work on security evaluation

Many authors implicitly performed security evaluation as a what-if analysis, based on empirical simulation methods; however, they mainly focused on a specific application, classifier and attack, and devised ad hoc security evaluation procedures based on the exploitation of problem knowledge and heuristic techniques [1]–[6], [15], [22]–[36]. Their goal was either to point out a previously unknown vulnerability, or to evaluate security against a known attack. In some cases, specific countermeasures were also proposed, according to

a proactive/security-by-design approach. Attacks were simulated by manipulating training and testing samples according to application-specific criteria only, without reference to more general guidelines; consequently, such techniques can not be directly exploited by a system designer in more general cases.

A few works proposed *analytical* methods to evaluate the security of learning algorithms or of some classes of decision functions (e.g., linear ones), based on more general, application-independent criteria to model the adversary's behavior (including PAC learning and game theory) [10], [12], [14], [16], [17], [37]–[40]. Some of these criteria will be exploited in our framework for *empirical* security evaluation; in particular, in the definition of the adversary model described in Sect. 3.1, as high-level guidelines for simulating attacks.

2.5 Building on previous work

We summarize here the three main concepts more or less explicitly emerged from previous work that will be exploited in our framework for security evaluation.

1) Arms race and security by design: since it is not possible to predict how many and which kinds of attacks a classifier will incur during operation, classifier security should be proactively evaluated using a what-if analysis, by *simulating* potential attack scenarios.

2) Adversary modeling: effective simulation of attack scenarios requires a formal model of the adversary.

3) Data distribution under attack: the distribution of testing data may differ from that of training data, when the classifier is under attack.

3 A FRAMEWORK FOR EMPIRICAL EVALUATION OF CLASSIFIER SECURITY

We propose here a framework for the empirical evaluation of classifier security in adversarial environments, that unifies and builds on the three concepts highlighted in Sect. 2.5. Our main goal is to provide a quantitative and general-purpose basis for the application of the *what-if analysis* to classifier security evaluation, based on the definition of potential attack scenarios. To this end, we propose: (i) a model of the adversary, that allows us to define any attack scenario; (ii) a corresponding model of the data distribution; and (iii) a method for

generating training and testing sets that are representative of the data distribution, and are used for empirical performance evaluation.

3.1 Attack scenario and model of the adversary

Although the definition of attack scenarios is ultimately an application-specific issue, it is possible to give general guidelines that can help the designer of a pattern recognition system. Here we propose to specify the attack scenario in terms of a conceptual model of the adversary that encompasses, unifies, and extends different ideas from previous work. Our model is based on the assumption that the adversary acts *rationally* to attain a given *goal*, according to her *knowledge* of the classifier, and her *capability* of manipulating data. This allows one to derive the corresponding optimal *attack strategy*.

Adversary's goal. As in [17], it is formulated as the optimization of an objective function. We propose to define this function based on the desired security violation (*integrity*, *availability*, or *privacy*), and on the attack *specificity* (from *targeted* to *indiscriminate*), according to the taxonomy in [11], [14] (see Sect. 2.1). For instance, the goal of an indiscriminate integrity violation may be to maximize the fraction of misclassified malicious samples [6], [10], [14]; the goal of a targeted privacy violation may be to obtain some specific, confidential information from the classifier (e.g., the biometric trait of a given user enrolled in a biometric system) by exploiting the class labels assigned to some "query" samples, while minimizing the number of query samples that the adversary has to issue to violate privacy [14], [16], [41].

Adversary's knowledge. Assumptions on the adversary's knowledge have only been qualitatively discussed in previous work, mainly depending on the application at hand. Here we propose a more systematic scheme for their definition, with respect to the knowledge of the single components of a pattern classifier: (k.i) the training data; (k.ii) the feature set; (k.iii) the learning algorithm and the kind of decision function (e.g., a linear SVM); (k.iv) the classifier's decision function and its parameters (e.g., the feature weights of a linear classifier); (k.v) the feedback available from the classifier, if any (e.g., the class labels assigned to some "query" samples that the adversary issues to get feedback [14], [16], [41]). It is worth noting that realistic and minimal assumptions about what can be kept fully secret from the adversary should be done, as discussed in [14]. Examples of adversary's knowledge are given in Sect. 4.

Adversary's capability. It refers to the control that the adversary has on training and testing data. We propose to define it in terms of: (c.i) the attack influence (either *causative* or *exploratory*), as defined in [11], [14]; (c.ii) whether and to what extent the attack affects the class priors; (c.iii) how many and which training and testing samples can be controlled by the adversary in each class; (c.iv) which features can be manipulated, and to what extent, taking into account application-specific constraints (e.g., correlated features can not be modified independently, and the functionality of malicious samples can not be compromised [1], [3], [6], [10]).

Attack strategy. One can finally define the *optimal* attack strategy, namely, how training and testing data should be *quantitatively* modified to optimize the objective function characterizing the adversary's goal. Such modifications are defined in terms of: (a.i) how the class priors are modified; (a.ii) what fraction of samples of each class is affected by the attack; and (a.iii) how features are manipulated by the attack. Detailed examples are given in Sect. 4.

Once the *attack scenario* is defined in terms of the adversary model and the resulting attack strategy, our framework proceeds with the definition of the corresponding data distribution, that is used to construct training and testing sets for security evaluation.

3.2 A model of the data distribution

We consider the standard setting for classifier design in a problem which consists of discriminating between legitimate (L) and malicious (M) samples: a learning algorithm and a performance measure have been chosen, a set $\mathcal{D}$ of n labelled samples has been collected, and a set of d features have been extracted, so that $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i$ denotes a d-dimensional feature vector, and $y_i \in \{\text{L}, \text{M}\}$ a class label. The pairs $(\mathbf{x}_i, y_i)$ are assumed to be i.i.d. samples of some unknown distribution $p_{\mathcal{D}}(\mathbf{X}, Y)$. Since the adversary model in Sect. 3.1 requires us to specify how the attack affects the class priors and the features of each class, we consider the classical generative model $p_{\mathcal{D}}(\mathbf{X}, Y) = p_{\mathcal{D}}(Y)p_{\mathcal{D}}(\mathbf{X}|Y)$.[5] To account for the presence of attacks during operation, which may affect either the training or the testing data, or both, we denote the corresponding training and testing distributions as p_{tr} and p_{ts}, respectively. We will just write p when we want to refer to either of them, or both, and the meaning is clear from the context.

When a classifier is not under attack, according to the classical stationarity assumption we have $p_{\text{tr}}(Y) = p_{\text{ts}}(Y) = p_{\mathcal{D}}(Y)$, and $p_{\text{tr}}(\mathbf{X}|Y) = p_{\text{ts}}(\mathbf{X}|Y) = p_{\mathcal{D}}(\mathbf{X}|Y)$. We extend this assumption to the components of p_{tr} and p_{ts} that are not affected by the attack (if any), by assuming that they remain identical to the corresponding distribution $p_{\mathcal{D}}$ (e.g., if the attack does not affect the class priors, the above equality also holds under attack).

The distributions $p(Y)$ and $p(\mathbf{X}|Y)$ that are affected by the attack can be defined as follows, according to the definition of the attack strategy, (a.i-iii).

Class priors. $p_{\text{tr}}(Y)$ and $p_{\text{ts}}(Y)$ can be immediately defined based on assumption (a.i).

Class-conditional distributions. $p_{\text{tr}}(\mathbf{X}|Y)$ and $p_{\text{ts}}(\mathbf{X}|Y)$ can be defined based on assumptions (a.ii-iii). First, to account for the fact that the attack may not

---

5. In this paper, for the sake of simplicity, we use the lowercase notation $p(\cdot)$ to denote any probability density function, with both continuous and discrete random variables.
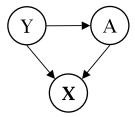
Fig. 2. Generative model of the distributions p_{tr} and p_{ts}.

modify all training or testing samples, according to (a.ii), we model $p(\mathbf{X}|Y)$ as a mixture controlled by a Boolean random variable A, which denotes whether a given sample has been manipulated ($A = \mathrm{T}$) or not ($A = \mathrm{F}$):

$$
\begin{aligned}
p(\mathbf{X}|Y) &= p(\mathbf{X}|Y, A = \mathrm{T})p(A = \mathrm{T}|Y) + \\
&\quad p(\mathbf{X}|Y, A = \mathrm{F})p(A = \mathrm{F}|Y).
\end{aligned}
\tag{1}
$$

We name the samples of the component $p(\mathbf{X}|Y, A = \mathrm{T})$ *attack samples* to emphasize that their distribution is different from that of samples which have not been manipulated by the adversary, $p(\mathbf{X}|Y, A = \mathrm{F})$. Note that $p(A = \mathrm{T}|Y = y)$ is the probability that an attack sample belongs to class y, i.e., the percentage of samples of class y controlled by the adversary. It is thus defined by (a.ii).

For samples which are unaffected by the attack, the stationarity assumption holds:

$$
p(\mathbf{X}|Y, A = \mathrm{F}) = p_{\mathcal{D}}(\mathbf{X}|Y).
\tag{2}
$$

The distribution $p(\mathbf{X}|Y, A = \mathrm{T})$ depends on assumption (a.iii). Depending on the problem at hand, it may not be possible to analytically define it. Nevertheless, for the purpose of security evaluation, we will show in Sect. 3.3 that it can be defined as the *empirical* distribution of a set of fictitious attack samples.

Finally, $p(\mathbf{X}|Y, A = \mathrm{F})$ can be defined similarly, if its analytical definition is not possible. In particular, according to Eq. (2), it can be defined as the empirical distribution of $\mathcal{D}$.

The above generative model of the training and testing distributions p_{tr} and p_{ts} is represented by the Bayesian network in Fig. 2, which corresponds to factorize $p(\mathbf{X}, Y, A)$ as follows:

$$
p(\mathbf{X}, Y, A) = p(Y)p(A|Y)p(\mathbf{X}|Y, A).
\tag{3}
$$

Our model can be easily extended to take into account concurrent attacks involving $m > 1$ different kinds of sample manipulations; for example, to model attacks against different classifiers in multimodal biometric systems [1], [2]. To this end, one can define m different Boolean random variables $\mathbf{A} = (A_1, \ldots, A_m)$, and the corresponding distributions. The extension of Eq. 3 is then straightforward (e.g., see [32]). The dependence between the different attacks, if any, can be modeled by the distribution $p(\mathbf{A}|Y)$.

If one is also interested in evaluating classifier security subject to temporal, non-adversarial variations of

the data distribution (e.g., the drift of the content of legitimate emails over time), it can be assumed that the distribution $p(\mathbf{X}, Y, A = \mathrm{F})$ changes over time, according to some model $p(\mathbf{X}, Y, A = \mathrm{F}, t)$ (see, e.g., [20]). Then, classifier security at time t can be evaluated by modeling the attack distribution over time $p(\mathbf{X}, Y, A = \mathrm{T}, t)$ as a function of $p(\mathbf{X}, Y, A = \mathrm{F}, t)$.

The model of the adversary in Sect. 3.1, and the above model of the data distribution provide a quantitative, well-grounded and general-purpose basis for the application of the what-if analysis to classifier security evaluation, which advances previous work.

3.3 Training and testing set generation

Here we propose an algorithm to sample training (TR) and testing (TS) sets of any desired size from the distributions $p_{\mathrm{tr}}(\mathbf{X}, Y)$ and $p_{\mathrm{ts}}(\mathbf{X}, Y)$.

We assume that $k \geq 1$ different pairs of training and testing sets $(\mathcal{D}_{\mathrm{TR}}^i, \mathcal{D}_{\mathrm{TS}}^i)$, $i = 1, \ldots, k$, have been obtained from $\mathcal{D}$ using a classical resampling technique, like cross-validation or bootstrapping. Accordingly, their samples follow the distribution $p_{\mathcal{D}}(\mathbf{X}, Y)$. In the following, we describe how to modify each of the sets $\mathcal{D}_{\mathrm{TR}}^i$ to construct a training set TR^i that follows the distribution $p_{\mathrm{tr}}(\mathbf{X}, Y)$. For the sake of simplicity, we will omit the superscript i. An identical procedure can be followed to construct a testing set TS^i from each of the $\mathcal{D}_{\mathrm{TS}}^i$. Security evaluation is then carried out with the classical method, by averaging (if $k > 1$) the performance of the classifier trained on TR^i and tested on TS^i.

If the attack does not affect the training samples, i.e., $p_{\mathrm{tr}}(\mathbf{X}, Y) = p_{\mathcal{D}}(\mathbf{X}, Y)$, TR is simply set equal to $\mathcal{D}_{\mathrm{TR}}$. Otherwise, two alternatives are possible. (i) If $p_{\mathrm{tr}}(\mathbf{X}|Y, A)$ is analytically defined for each $Y \in \{\mathrm{L}, \mathrm{M}\}$ and $A \in \{\mathrm{T}, \mathrm{F}\}$, then TR can be obtained by sampling the generative model of $p(\mathbf{X}, Y, A)$ of Eq. (3): first, a class label y is sampled from $p_{\mathrm{tr}}(Y)$, then, a value a from $p_{\mathrm{tr}}(A|Y = y)$, and, finally, a feature vector $\mathbf{x}$ from $p_{\mathrm{tr}}(\mathbf{X}|Y = y, A = a)$. (ii) If $p_{\mathrm{tr}}(\mathbf{X}|Y = y, A = a)$ is not analytically defined for some y and a, but a set of its samples is available, denoted in the following as $\mathcal{D}_{\mathrm{TR}}^{y,a}$, it can be approximated as the empirical distribution of $\mathcal{D}_{\mathrm{TR}}^{y,a}$. Accordingly, we can sample with replacement from $\mathcal{D}_{\mathrm{TR}}^{y,a}$ [42]. An identical procedure can be used to construct the testing set TS. The procedure to obtain TR or TS is formally described as Algorithm 1.[6]

Let us now discuss how to construct the sets $\mathcal{D}_{\mathrm{TR}}^{y,a}$, when $p_{\mathrm{tr}}(\mathbf{X}|Y = y, A = a)$ is not analytically defined. The same discussion holds for the sets $\mathcal{D}_{\mathrm{TS}}^{y,a}$. First, the two sets $\mathcal{D}_{\mathrm{TR}}^{\mathrm{L,F}}$ and $\mathcal{D}_{\mathrm{TR}}^{\mathrm{M,F}}$ can be respectively set equal

---

6. Since the proposed algorithm is based on classical resampling techniques such as cross-validation and bootstrapping, it is reasonable to expect that the bias and the variance of the estimated classification error (or of any other performance measure) will enjoy similar statistical properties to those exhibited by classical performance evaluation methods based on the same techniques. In practice, these error components are typically negligible with respect to errors introduced by the use of limited training data, biased learning/classification algorithms, and noisy or corrupted data.

Algorithm 1 Construction of TR or TS.

Input: The number n of desired samples;
the distributions $p(Y)$ and $p(A|Y)$;
for each $y \in \{L, M\}, a \in \{T, F\}$, the distribution $p(\mathbf{X}|Y = y, A = a)$, if analytically defined, or the set of samples $\mathcal{D}^{y,a}$, otherwise.

Output: A data set $\mathcal{S}$ (either TR or TS) drawn from $p(Y)p(A|Y)p(\mathbf{X}|Y, A)$.

1: $S \leftarrow \emptyset$
2: **for** $i = 1, \ldots, n$ **do**
3:     sample y from $p(Y)$
4:     sample a from $p(A|Y = y)$
5:     draw a sample $\mathbf{x}$ from $p(\mathbf{X}|Y = y, A = a)$, if analytically defined; otherwise, sample with replacement from $\mathcal{D}^{y,a}$
6:     $\mathcal{S} \leftarrow \mathcal{S} \bigcup \{(\mathbf{x}, y)\}$
7: **end for**
8: **return** $\mathcal{S}$

---

to the legitimate and malicious samples in $\mathcal{D}_{\mathrm{TR}}$, since the distribution of such samples is assumed to be just $p_{\mathrm{tr}}(\mathbf{X}|Y = \mathrm{L}, A = \mathrm{F})$ and $p_{\mathrm{tr}}(\mathbf{X}|Y = \mathrm{M}, A = \mathrm{F})$ (see Eq. 2): $\mathcal{D}_{\mathrm{TR}}^{\mathrm{L,F}} = \{(\mathbf{x}, y) \in \mathcal{D}_{\mathrm{TR}} : y = \mathrm{L}\}$, $\mathcal{D}_{\mathrm{TR}}^{\mathrm{M,F}} = \{(\mathbf{x}, y) \in \mathcal{D}_{\mathrm{TR}} : y = \mathrm{M}\}$. The two sets of attack samples $\mathcal{D}_{\mathrm{TR}}^{y,\mathrm{T}}$, for $y = \mathrm{L, M}$, must come instead from $p_{\mathrm{tr}}(\mathbf{X}|Y = y, A = \mathrm{T})$. They can thus be constructed according to the point (a.iii) of the attack strategy, using any technique for simulating attack samples. Therefore, all the ad hoc techniques used in previous work for constructing fictitious attack samples can be used as methods to define or empirically approximate the distribution $p(\mathbf{X}|Y = y, A = \mathrm{T})$, for $y = \mathrm{L, M}$.

Two further considerations must be made on the simulated attack samples $\mathcal{D}_{\mathrm{TR}}^{y,\mathrm{T}}$. First, the number of distinct attack samples that can be obtained depends on the simulation technique, when it requires the use of the data in $\mathcal{D}_{\mathrm{TR}}$; for instance, if it consists of modifying *deterministically* each malicious sample in $\mathcal{D}_{\mathrm{TR}}$, no more than $|\{(\mathbf{x}, y) \in \mathcal{D}_{\mathrm{TR}} : y = \mathrm{M}\}|$ distinct attack samples can be obtained. If any number of attack samples can be generated, instead (e.g., [5], [28], [34], [36]), then the sets $\mathcal{D}_{\mathrm{TR}}^{y,\mathrm{T}}$, for $y = \mathrm{L, M}$, do not need to be constructed beforehand: a single attack sample can be generated online at step 5 of Algorithm 1, when $a = \mathrm{T}$. Second, in some cases it may be necessary to construct the attack samples incrementally; for instance, in the causative attacks considered in [11], [30], [36], the attack samples are added to the training data one at a time, since each of them is a function of the *current* training set. This corresponds to a non-i.i.d. sampling of the attack distribution. In these cases, Algorithm 1 can be modified by first generating all pairs y, a, then, the feature vectors $\mathbf{x}$ corresponding to $a = \mathrm{F}$, and, lastly, the attack samples corresponding to $a = \mathrm{T}$, one at a time.

3.4 How to use our framework

We summarize here the steps that the designer of a pattern classifier should take to evaluate its security using our framework, for each attack scenario of interest. They extend the performance evaluation step of the classical design cycle of [9], which is used as part of the model selection phase, and to evaluate the final classifier to be deployed.

1) Attack scenario. The attack scenario should be defined at the conceptual level by making specific assumptions on the goal, knowledge (k.i-v), and capability of the adversary (c.i-iv), and defining the corresponding attack strategy (a.i-iii), according to the model of Sect. 3.1.

2) Data model. According to the hypothesized attack scenario, the designer should define the distributions $p(Y)$, $p(A|Y)$, and $p(\mathbf{X}|Y, A)$, for $Y \in \{L, M\}$, $A \in \{F, T\}$, and for training and testing data. If $p(\mathbf{X}|Y, A)$ is not analytically defined for some $Y = y$ and $A = a$, either for training or testing data, the corresponding set $\mathcal{D}_{\mathrm{TR}}^{y,a}$ or $\mathcal{D}_{\mathrm{TS}}^{y,a}$ must be constructed. The sets $\mathcal{D}_{\mathrm{TR}}^{y,\mathrm{F}}$ ($\mathcal{D}_{\mathrm{TS}}^{y,\mathrm{F}}$) are obtained from $\mathcal{D}_{\mathrm{TR}}$ ($\mathcal{D}_{\mathrm{TS}}$). The sets $\mathcal{D}_{\mathrm{TR}}^{y,\mathrm{T}}$ and $\mathcal{D}_{\mathrm{TS}}^{y,\mathrm{T}}$ can be generated, only if the attack involves sample manipulation, using an attack sample simulation technique according to the attack strategy (a.iii).

3) Construction of TR and TS. Given $k \geq 1$ pairs $(\mathcal{D}_{\mathrm{TR}}^i, \mathcal{D}_{\mathrm{TS}}^i)$, $i = 1, \ldots, k$, obtained from classical resampling techniques like cross-validation or bootstrapping, the size of TR and TS must be defined, and Algorithm 1 must be run with the corresponding inputs to obtain TR^i and TS^i. If the attack does not affect the training (testing) data, TR^i (TS^i) is set to $\mathcal{D}_{\mathrm{TR}}^i$ ($\mathcal{D}_{\mathrm{TS}}^i$).

4) Performance evaluation. The classifier performance under the simulated attack is evaluated using the constructed (TR^i, TS^i) pairs, as in classical techniques.

4 APPLICATION EXAMPLES

While previous work focused on a single application, we consider here three different application examples of our framework in spam filtering, biometric authentication, and network intrusion detection. Our aim is to show how the designer of a pattern classifier can use our framework, and what kind of additional information he can obtain from security evaluation. We will show that a trade-off between classifier accuracy and security emerges sometimes, and that this information can be exploited for several purposes; e.g., to improve the model selection phase by considering both classification accuracy and security.

4.1 Spam filtering

Assume that a classifier has to discriminate between legitimate and spam emails on the basis of their textual content, and that the *bag-of-words* feature representation has been chosen, with binary features denoting the occurrence of a given set of words. This kind of classifier

Attack scenario	Sect. 4.1 Indiscrim. Exploratory Integrity	Sect. 4.2 Targeted Exploratory Integrity	Sect. 4.3 Indiscrim. Causative Integrity
$p_{\mathrm{tr}}(Y)$	$p_{\mathcal{D}}(Y)$	$p_{\mathcal{D}}(Y)$	$p_{\mathrm{tr}}(\mathrm{M})=p_{\max}$
$p_{\mathrm{tr}}(A=\mathrm{T}|Y=\mathrm{L})$	0	0	0
$p_{\mathrm{tr}}(A=\mathrm{T}|Y=\mathrm{M})$	0	0	1
$p_{\mathrm{tr}}(\mathbf{X}|Y=\mathrm{L},A=\mathrm{T})$	-	-	-
$p_{\mathrm{tr}}(\mathbf{X}|Y=\mathrm{M},A=\mathrm{T})$	-	-	empirical
$p_{\mathrm{tr}}(\mathbf{X}|Y=\mathrm{L},A=\mathrm{F})$	$p_{\mathcal{D}}(\mathbf{X}|\mathrm{L})$	$p_{\mathcal{D}}(\mathbf{X}|\mathrm{L})$	$p_{\mathcal{D}}(\mathbf{X}|\mathrm{L})$
$p_{\mathrm{tr}}(\mathbf{X}|Y=\mathrm{M},A=\mathrm{F})$	$p_{\mathcal{D}}(\mathbf{X}|\mathrm{M})$	$p_{\mathcal{D}}(\mathbf{X}|\mathrm{M})$	-
$p_{\mathrm{ts}}(Y)$	$p_{\mathcal{D}}(Y)$	$p_{\mathcal{D}}(Y)$	$p_{\mathcal{D}}(Y)$
$p_{\mathrm{ts}}(A=\mathrm{T}|Y=\mathrm{L})$	0	0	0
$p_{\mathrm{ts}}(A=\mathrm{T}|Y=\mathrm{M})$	1	1	0
$p_{\mathrm{ts}}(\mathbf{X}|Y=\mathrm{L},A=\mathrm{T})$	-	-	-
$p_{\mathrm{ts}}(\mathbf{X}|Y=\mathrm{M},A=\mathrm{T})$	empirical	empirical	-
$p_{\mathrm{ts}}(\mathbf{X}|Y=\mathrm{L},A=\mathrm{F})$	$p_{\mathcal{D}}(\mathbf{X}|\mathrm{L})$	$p_{\mathcal{D}}(\mathbf{X}|\mathrm{L})$	$p_{\mathcal{D}}(\mathbf{X}|\mathrm{L})$
$p_{\mathrm{ts}}(\mathbf{X}|Y=\mathrm{M},A=\mathrm{F})$	-	-	$p_{\mathcal{D}}(\mathbf{X}|\mathrm{M})$

TABLE 1
Parameters of the attack scenario and of the data model for each application example. 'Empirical' means that $p(\mathbf{X}|Y=y,A=a)$ was approximated as the empirical distribution of a set of samples $D^{y,a}$; and '-' means that no samples from this distribution were needed to simulate the attack.

has been considered by several authors [6], [28], [43], and it is included in several real spam filters.[7]

In this example, we focus on model selection. We assume that the designer wants to choose between a support vector machine (SVM) with a linear kernel, and a logistic regression (LR) linear classifier. He also wants to choose a feature subset, among all the words occurring in training emails. A set $\mathcal{D}$ of legitimate and spam emails is available for this purpose. We assume that the designer wants to evaluate not only classifier accuracy in the absence of attacks, as in the classical design scenario, but also its security against the well-known bad word obfuscation (BWO) and good word insertion (GWI) attacks. They consist of modifying spam emails by inserting "good words" that are likely to appear in legitimate emails, and by obfuscating "bad words" that are typically present in spam [6]. The attack scenario can be modeled as follows.

1) Attack scenario. *Goal*. The adversary aims at maximizing the percentage of spam emails misclassified as legitimate, which is an indiscriminate integrity violation.

Knowledge. As in [6], [10], the adversary is assumed to have perfect knowledge of the classifier, i.e.: (k.ii) the feature set, (k.iii) the kind of decision function, and (k.iv) its parameters (the weight assigned to each feature, and the decision threshold). Assumptions on the knowledge of (k.i) the training data and (k.v) feedback from the classifier are not relevant in this case, as they do not provide any additional information.

Capability. We assume that the adversary: (c.i) is only

7. SpamAssassin, http://spamassassin.apache.org/; Bogofilter, http://bogofilter.sourceforge.net/; SpamBayes http://spambayes.sourceforge.net/

able to influence testing data (exploratory attack); (c.ii) can not modify the class priors; (c.iii) can manipulate each malicious sample, but no legitimate ones; (c.iv) can manipulate any feature value (i.e., she can insert or obfuscate any word), but up to a maximum number $n_{\max}$ of features in each spam email [6], [10]. This allows us to evaluate how gracefully the classifier performance degrades as an increasing number of features is modified, by repeating the evaluation for increasing values of $n_{\max}$.

Attack strategy. Without loss of generality, let us further assume that $\mathbf{x}$ is classified as legitimate if $g(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i + w_0 < 0$, where $g(\cdot)$ is the discriminant function of the classifier, n is the feature set size, $x_i \in \{0,1\}$ are the feature values (1 and 0 denote respectively the presence and the absence of the corresponding term), w_i are the feature weights, and w_0 is the bias.

Under the above assumptions, the optimal attack strategy can be attained by: (a.i) leaving the class priors unchanged; (a.ii) manipulating *all* testing spam emails; and (a.iii) modifying up to $n_{\max}$ words in each spam to *minimize* the discriminant function of the classifier.

Each attack sample (i.e., modified spam) can be thus obtained by solving a constrained optimization problem. As in [10], the generation of attack samples can be represented by a function $\mathcal{A}: \mathcal{X} \mapsto \mathcal{X}$, and the number of modified words can be evaluated by the Hamming distance. Accordingly, for any given $\mathbf{x} \in \mathcal{X}$, the optimal attack strategy amounts to finding the attack sample $\mathcal{A}(\mathbf{x})$ which minimizes $g(\mathcal{A}(\mathbf{x}))$, subject to the constraint that the Hamming distance between $\mathbf{x}$ and $\mathcal{A}(\mathbf{x})$ is no greater than $n_{\max}$, that is:

$$\mathcal{A}(\mathbf{x}) = \mathrm{argmin}_{\mathbf{x}'} \quad \sum_{i=1}^{n} w_i x_i'$$
$$\text{s.t.} \quad \sum_{i=1}^{n} |x_i' - x_i| \leq n_{\max}. \qquad (4)$$

The solution to problem (4) is straightforward. First, note that inserting (obfuscating) a word results in switching the corresponding feature value from 0 to 1 (1 to 0), and that the minimum of $g(\cdot)$ is attained when all features that have been assigned a negative (positive) weight are equal to 1 (0). Accordingly, for a given $\mathbf{x}$, the largest *decrease* of $g(\mathcal{A}(\mathbf{x}))$ subject to the above constraint is obtained by analyzing all features for decreasing values of $|w_i|$, and switching from 0 to 1 (from 1 to 0) the ones corresponding to $w_i < 0$ ($w_i > 0$), until $n_{\max}$ changes have been made, or all features have been analyzed. Note that this attack can be simulated by directly manipulating the feature vectors of spam emails instead of the emails themselves.

2) Data model. Since the adversary can only manipulate testing data, we set $p_{\mathrm{tr}}(Y) = p_{\mathcal{D}}(Y)$, and $p_{\mathrm{tr}}(\mathbf{X}|Y) = p_{\mathcal{D}}(\mathbf{X}|Y)$. Assumptions (a.i-ii) are directly encoded as: (a.i) $p_{\mathrm{ts}}(Y) = p_{\mathcal{D}}(Y)$; and (a.ii) $p_{\mathrm{ts}}(A=\mathrm{T}|Y=\mathrm{M}) = 1$, and $p_{\mathrm{ts}}(A=\mathrm{T}|Y=\mathrm{L}) = 0$. The latter implies that $p_{\mathrm{ts}}(\mathbf{X}|Y=\mathrm{L}) = p_{\mathcal{D}}(\mathbf{X}|Y=\mathrm{L})$. Assumption (a.iii) is encoded into the above function $\mathcal{A}(\mathbf{x})$, which *empirically*

defines the distribution of the attack samples $p_{\mathrm{ts}}(\mathbf{X}|Y = \mathrm{M}, A = \mathrm{T})$. Similarly, $p(\mathbf{X}|Y = y, A = \mathrm{F}) = p_{\mathcal{D}}(\mathbf{X}|y)$, for $y = \mathrm{L}, \mathrm{M}$, will be approximated as the empirical distribution of the set $\mathcal{D}_{\mathrm{TS}}^{y,\mathrm{F}}$ obtained from $\mathcal{D}_{\mathrm{TS}}$, as described below; the same approximation will be made in all the considered application examples.

The definition of the attack scenario and the data model is summarized in Table 1 (first column).

3) Construction of TR and TS. We use a publicly available email corpus, TREC 2007. It consists of 25,220 legitimate and 50,199 real spam emails.[8] We select the first 20,000 emails in chronological order to create the data set $\mathcal{D}$. We then split $\mathcal{D}$ into two subsets $\mathcal{D}_{\mathrm{TR}}$ and $\mathcal{D}_{\mathrm{TS}}$, respectively made up of the first 10,000 and the next 10,000 emails, in chronological order. We use $\mathcal{D}_{\mathrm{TR}}$ to construct TR and $\mathcal{D}_{\mathrm{TS}}$ to construct TS, as in [5], [6]. Since the considered attack does not affect training samples, we set $\mathrm{TR} = \mathcal{D}_{\mathrm{TR}}$. Then, we define $\mathcal{D}_{\mathrm{TS}}^{\mathrm{L},\mathrm{F}}$ and $\mathcal{D}_{\mathrm{TS}}^{\mathrm{M},\mathrm{F}}$ respectively as the legitimate and malicious samples in $\mathcal{D}_{\mathrm{TS}}$, and construct the set of attack samples $\mathcal{D}_{\mathrm{TS}}^{\mathrm{M},\mathrm{T}}$ by modifying all the samples in $\mathcal{D}_{\mathrm{TS}}^{\mathrm{M},\mathrm{F}}$ according to $\mathcal{A}(\mathbf{x})$, for any fixed $n_{\max}$ value. Since the attack does not affect the legitimate samples, we set $\mathcal{D}_{\mathrm{TS}}^{\mathrm{L},\mathrm{T}} = \emptyset$. Finally, we set the size of TS to 10,000, and generate TS by running Algorithm 1 on $\mathcal{D}_{\mathrm{TS}}^{\mathrm{L},\mathrm{F}}$, $\mathcal{D}_{\mathrm{TS}}^{\mathrm{M},\mathrm{F}}$ and $\mathcal{D}_{\mathrm{TS}}^{\mathrm{M},\mathrm{T}}$.

The features (words) are extracted from TR using the SpamAssassin tokenization method. Four feature subsets with size 1,000, 2,000, 10,000 and 20,000 have been selected using the information gain criterion [44].

4) Performance evaluation. The performance measure we use is the area under the receiver operating characteristic curve (AUC) corresponding to false positive (FP) error rates in the range $[0, 0.1]$ [6]: $\mathrm{AUC}_{10\%} = \int_0^{0.1} \mathrm{TP}(\mathrm{FP})\mathrm{dFP} \in [0, 0.1]$, where TP is the true positive error rate. It is suited to classification tasks like spam filtering, where FP errors (i.e., legitimate emails misclassified as spam) are much more harmful than false negative (FN) ones.

Under the above model selection setting (two classifiers, and four feature subsets) eight different classifier models must be evaluated. Each model is trained on TR. SVMs are implemented with the LibSVM software [45]. The C parameter of their learning algorithm is chosen by maximizing the $\mathrm{AUC}_{10\%}$ through a 5-fold cross-validation on TR. An online gradient descent algorithm is used for LR. After classifier training, the $\mathrm{AUC}_{10\%}$ value is assessed on TS, for different values of $n_{\max}$. In this case, it is a monotonically decreasing function of $n_{\max}$. The more graceful its decrease, the more robust the classifier is to the considered attack. Note that, for $n_{\max} = 0$, no attack samples are included in the testing set: the corresponding $\mathrm{AUC}_{10\%}$ value equals that attained by classical performance evaluation methods.

The results are reported in Fig. 3. As expected, the $\mathrm{AUC}_{10\%}$ of each model decreases as $n_{\max}$ increases. It drops to zero for $n_{\max}$ values between 30 and 50

8. http://plg.uwaterloo.ca/~gvcormac/treccorpus07

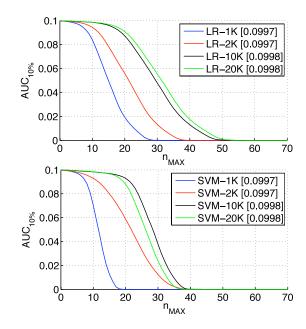
Fig. 3. $\mathrm{AUC}_{10\%}$ attained on TS as a function of $n_{\max}$, for the LR (top) and SVM (bottom) classifier, with 1,000 (1K), 2,000 (2K), 10,000 (10K) and 20,000 (20K) features. The $\mathrm{AUC}_{10\%}$ value for $n_{\max} = 0$, corresponding to classical performance evaluation, is also reported in the legend between square brackets.

(depending on the classifier): this means that all testing spam emails got misclassified as legitimate, after adding or obfuscating from 30 to 50 words.

The SVM and LR classifiers perform very similarly when they are not under attack (i.e., for $n_{\max} = 0$), regardless of the feature set size; therefore, according to the viewpoint of classical performance evaluation, the designer could choose any of the eight models. However, security evaluation highlights that they exhibit a very different robustness to the considered attack, since their $\mathrm{AUC}_{10\%}$ value decreases at very different rates as $n_{\max}$ increases; in particular, the LR classifier with 20,000 features clearly outperforms all the other ones, for all $n_{\max}$ values. This result suggests the designer a very different choice than the one coming from classical performance evaluation: the LR classifier with 20,000 features should be selected, given that it exhibit the same accuracy as the other ones in the absence of attacks, and a higher security under the considered attack.

4.2 Biometric authentication

Multimodal biometric systems for personal identity recognition have received great interest in the past few years. It has been shown that combining information coming from different biometric traits can overcome the limits and the weaknesses inherent in every individual biometric, resulting in a higher accuracy. Moreover, it is commonly believed that multimodal systems also improve security against spoofing attacks, which consist of claiming a false identity and submitting at least one

fake biometric trait to the system (e.g., a "gummy" fingerprint or a photograph of a user's face). The reason is that, to evade a multimodal system, one expects that the adversary should spoof *all* the corresponding biometric traits. In this application example, we show how the designer of a multimodal system can verify if this hypothesis holds, before deploying the system, by simulating spoofing attacks against each of the matchers. To this end, we partially exploit the analysis in [1], [2].

We consider a typical multimodal system, made up of a fingerprint and a face matcher, which operates as follows. The design phase includes the enrollment of authorized users (clients): reference templates of their biometric traits are stored into a database, together with the corresponding identities. During operation, each user provides the requested biometric traits to the sensors, and claims the identity of a client. Then, each matcher compares the submitted trait with the template of the claimed identity, and provides a real-valued matching score: the higher the score, the higher the similarity. We denote the score of the fingerprint and the face matcher respectively as x_{fing} and x_{face}. Finally, the matching scores are combined through a proper fusion rule to decide whether the claimed identity is the user's identity (genuine user) or not (impostor).

As in [1], [2], we consider the widely used likelihood ratio (LLR) score fusion rule [46], and the matching scores as its features. Denoting as $\mathbf{x}$ the feature vector $(x_{\text{fing}}, x_{\text{face}})$, the LLR can be written as:

$$LLR(\mathbf{x}) = \begin{cases} \text{L} & \text{if } p(\mathbf{x}|Y = \text{L})/p(\mathbf{x}|Y = \text{M}) \geq t, \\ \text{M} & \text{if } p(\mathbf{x}|Y = \text{L})/p(\mathbf{x}|Y = \text{M}) < t, \end{cases} \quad (5)$$

where L and M denote respectively the "genuine" and "impostor" class, and t is a decision threshold set according to application requirements. The distributions $p(x_{\text{fing}}, x_{\text{face}}|Y)$ are usually estimated from training data, while t is estimated from a validation set.

1) Attack scenario. *Goal*. In this case, each malicious user (impostor) aims at being accepted as a legitimate (genuine) one. This corresponds to a targeted integrity violation, where the adversary's goal is to maximize the matching score.

Knowledge. As in [1], [2], we assume that each impostor knows: (k.i) the identity of the targeted client; and (k.ii) the biometric traits used by the system. No knowledge of (k.iii) the decision function and (k.iv) its parameters is assumed, and (k.v) no feedback is available from the classifier.

Capability. We assume that: (c.i) spoofing attacks affect only testing data (exploratory attack); (c.ii) they do not affect the class priors;[9] and, according to (k.i), (c.iii) each adversary (impostor) controls her testing malicious samples. We limit the capability of each impostor by assuming that (c.iv) only one specific trait can be spoofed at a time (i.e., only one feature can be manipulated), and

that it is the same for all impostors. We will thus repeat the evaluation twice, considering fingerprint and face spoofing, separately. Defining how impostors can manipulate the features (in this case, the matching scores) in a spoofing attack is not straightforward. The standard way of evaluating the security of biometric systems against spoofing attacks consists of fabricating fake biometric traits and assessing performance when they are submitted to the system. However, this is a cumbersome task. Here we follow the approach of [1], [2], in which the impostor is assumed to fabricate "perfect" fake traits, i.e., fakes that produce the same matching score of the "live" trait of the targeted client. This allows one to use the available genuine scores to simulate the fake scores, avoiding the fabrication of fake traits. On the other hand, this assumption may be too pessimistic. This limitation could be overcome by developing more realistic models of the distribution of the fake traits; however, this is a challenging research issue on its own, that is out of the scope of this work, and part of the authors' ongoing work [47], [48].

Attack strategy. The above attack strategy modifies only the testing data, and: (a.i) it does not modify the class priors; (a.ii) it does not affect the genuine class, but all impostors; and (a.iii) any impostor spoofs the considered biometric trait (either the face or fingerprint) of the known identity. According to the above assumption, as in [1], [2], any spoofing attack is simulated by replacing the corresponding impostor score (either the face or fingerprint) with the score of the targeted genuine user (chosen at random from the legitimate samples in the testing set). This can be represented with a function $\mathcal{A}(\mathbf{x})$, that, given an impostor $\mathbf{x} = (x_{\text{fing}}, x_{\text{face}})$, returns either a vector $(x'_{\text{fing}}, x_{\text{face}})$ (for fingerprint spoofing) or $(x_{\text{fing}}, x'_{\text{face}})$ (for face spoofing), where x'_{fing} and x'_{face} are the matching scores of the targeted genuine user.

2) Data model. Since training data is not affected, we set $p_{\text{tr}}(\mathbf{X}, Y) = p_{\mathcal{D}}(\mathbf{X}, Y)$. Then, we set (a.i) $p_{\text{ts}}(Y) = p_{\mathcal{D}}(Y)$; and (a.ii) $p_{\text{ts}}(A = \text{T}|Y = \text{L}) = 0$ and $p_{\text{ts}}(A = \text{T}|Y = \text{M}) = 1$. Thus, $p_{\text{ts}}(\mathbf{X}|Y = \text{L}) = p_{\mathcal{D}}(\mathbf{X}|Y = \text{L})$, while all the malicious samples in TS have to be manipulated by $\mathcal{A}(\mathbf{x})$, according to (a.iii). This amounts to *empirically* defining $p_{\text{ts}}(\mathbf{X}|Y = \text{M}, A = \text{T})$. As in the spam filtering case, $p_{\text{ts}}(\mathbf{X}|Y = y, A = \text{F})$, for $y = \text{L, M}$, will be empirically approximated from the available data in $\mathcal{D}$.

The definition of the above attack scenario and data model is summarized in Table 1 (second column).

3) Construction of TR and TS. We use the NIST Biometric Score Set, Release 1.[10] It contains raw similarity scores obtained on a set of 517 users from two different face matchers (named 'G' and 'C'), and from one fingerprint matcher using the left and right index finger. For each user, one genuine score and 516 impostor scores are available for each matcher and each modality, for a total of 517 genuine and 266,772 impostor samples. We use the scores of the 'G' face matcher and the ones

---

9. Further, since the LLR considers only the ratio between the class-conditional distributions, changing the class priors is irrelevant.
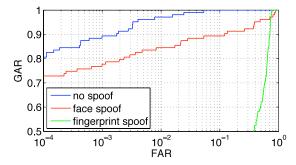
Fig. 4. ROC curves of the considered multimodal biometric system, under a simulated spoof attack against the fingerprint or the face matcher.

of the fingerprint matcher for the left index finger, and normalize them in $[0,1]$ using the min-max technique. The data set $\mathcal{D}$ is made up of pairs of face and fingerprint scores, each belonging to the same user. We first randomly subdivide $\mathcal{D}$ into a disjoint training and testing set, $\mathcal{D}_{\text{TR}}$ and $\mathcal{D}_{\text{TS}}$, containing respectively 80% and 20% of the samples. As the attack does not affect the training samples, we set TR= $\mathcal{D}_{\text{TR}}$. The sets $\mathcal{D}_{\text{TS}}^{\text{L,F}}$ and $\mathcal{D}_{\text{TS}}^{\text{M,F}}$ are constructed using $\mathcal{D}_{\text{TS}}$, while $\mathcal{D}_{\text{TS}}^{\text{L,F}} = \emptyset$. The set of attack samples $\mathcal{D}_{\text{TS}}^{\text{M,T}}$ is obtained by modifying each sample of $\mathcal{D}_{\text{TS}}^{\text{M,F}}$ with $\mathcal{A}(\mathbf{x})$. We finally set the size of TS as $|\mathcal{D}_{\text{TS}}|$, and run Algorithm 1 to obtain it.

4) Performance evaluation. In biometric authentication tasks, the performance is usually measured in terms of genuine acceptance rate (GAR) and false acceptance rate (FAR), respectively the fraction of genuine and impostor attempts that are accepted as genuine by the system. We use here the complete ROC curve, which shows the GAR as a function of the FAR for all values of the decision threshold t (see Eq. 5).

To estimate $p(\mathbf{X}|Y = y)$, for $y = \text{L}, \text{M}$, in the LLR score fusion rule (Eq. 5), we assume that x_{fing} and x_{face} are conditionally independent given Y, as usually done in biometrics. We thus compute a maximum likelihood estimate of $p(x_{\text{fing}}, x_{\text{face}}|Y)$ from TR using a product of two Gamma distributions, as in [1].

Fig. 4 shows the ROC curve evaluated with the standard approach (without spoofing attacks), and the ones corresponding to the simulated spoofing attacks. The FAR axis is in logarithmic scale to focus on low FAR values, which are the most relevant ones in security applications. For any operational point on the ROC curve (namely, for any value of the decision threshold t), the effect of the considered spoofing attack is to increase the FAR, while the GAR does not change. This corresponds to a shift of the ROC curve to the right. Accordingly, the FAR under attack must be compared with the original one, for the same GAR.

Fig. 4 clearly shows that the FAR of the biometric system significantly increases under the considered attack scenario, especially for fingerprint spoofing. As far as this attack scenario is deemed to be realistic, the system

designer should conclude that the considered system can be evaded by spoofing only one biometric trait, and thus does not exhibit a higher security than each of its individual classifiers. For instance, in applications requiring relatively high security, a reasonable choice may be to chose the operational point with FAR=10^{-3} and GAR=0.90, using classical performance evaluation, i.e., without taking into account spoofing attacks. This means that the probability that the deployed system wrongly accepts an impostor as a genuine user (the FAR) is expected to be 0.001. However, under face spoofing, the corresponding FAR increases to 0.10, while it jumps to about 0.70 under fingerprint spoofing. In other words, an impostor who submits a perfect replica of the face of a client has a probability of 10% of being accepted as genuine, while this probability is as high as 70%, if she is able to perfectly replicate the fingerprint. This is unacceptable for security applications, and provides further support to the conclusions of [1], [2] against the common belief that multimodal biometric systems are intrinsically more robust than unimodal systems.

4.3 Network intrusion detection

Intrusion detection systems (IDSs) analyze network traffic to prevent and detect malicious activities like intrusion attempts, port scans, and denial-of-service attacks.[11] When suspected malicious traffic is detected, an alarm is raised by the IDS and subsequently handled by the system administrator. Two main kinds of IDSs exist: misuse detectors and anomaly-based ones. *Misuse detectors* match the analyzed network traffic against a database of signatures of known malicious activities (e.g., `Snort`).[12] The main drawback is that they are not able to detect never-before-seen malicious activities, or even variants of known ones. To overcome this issue, *anomaly-based* detectors have been proposed. They build a statistical model of the normal traffic using machine learning techniques, usually one-class classifiers (e.g., `PAYL` [49]), and raise an alarm when anomalous traffic is detected. Their training set is constructed, and periodically updated to follow the changes of normal traffic, by collecting unsupervised network traffic during operation, assuming that it is normal (it can be filtered by a misuse detector, and should be discarded if some system malfunctioning occurs during its collection). This kind of IDS is vulnerable to causative attacks, since an attacker may inject carefully designed malicious traffic during the collection of training samples to force the IDS to learn a wrong model of the normal traffic [11], [17], [29], [30], [36].

11. The term "attack" is used in this field to denote a malicious activity, even when there is no deliberate attempt of misleading an IDS. In adversarial classification, as in this paper, this term is used to specifically denote the attempt of misleading a classifier, instead. To avoid any confusion, in the following we will refrain from using the term "attack" with the former meaning, using paraphrases, instead.

12. http://www.snort.org/

Here we assume that an anomaly-based IDS is being designed, using a one-class ν-SVM classifier with a radial basis function (RBF) kernel and the feature vector representation proposed in [49]. Each network packet is considered as an individual sample to be labeled as normal (legitimate) or anomalous (malicious), and is represented as a 256-dimensional feature vector, defined as the histogram of byte frequencies in its payload (this is known as "1-gram" representation in the IDS literature). We then focus on the model selection stage. In the above setting, it amounts to choosing the values of the ν parameter of the learning algorithm (which is an upper bound on the false positive error rate on training data [50]), and the γ value of the RBF kernel. For the sake of simplicity, we assume that ν is set to 0.01 as suggested in [24], so that only γ has to be chosen.

We show how the IDS designer can select a model (the value of γ) based also on the evaluation of classifier security. We focus on a causative attack similar to the ones considered in [38], aimed at forcing the learned model of normal traffic to include samples of intrusions to be attempted during operation. To this end, the attack samples should be carefully designed such that they include some features of the desired intrusive traffic, but do not perform any real intrusion (otherwise, the collected traffic may be discarded, as explained above).

1) Attack scenario. *Goal*. This attack aims to cause an indiscriminate integrity violation by maximizing the fraction of malicious testing samples misclassified as legitimate.

Knowledge. The adversary is assumed to know: (k.ii) the feature set; and (k.iii) that a one-class classifier is used. No knowledge of (k.i) the training data and (k.iv) the classifiers' parameters is available to the adversary, as well as (k.v) any feedback from the classifier.

Capability. The attack consists of injecting *malicious* samples into the training set. Accordingly, we assume that: (c.i) the adversary can inject malicious samples into the training data, without manipulating testing data (causative attack); (c.ii) she can modify the class priors by injecting a maximum fraction $p_{\max}$ of malicious samples into the training data; (c.iii) all the injected malicious samples can be manipulated; and (c.iv) the adversary is able to completely control the feature values of the malicious attack samples. As in [17], [28]–[30], we repeat the security evaluation for $p_{\max} \in [0, 0.5]$, since it is unrealistic that the adversary can control the majority of the training data.

Attack strategy. The goal of maximizing the percentage of malicious testing samples misclassified as legitimate, for any $p_{\max}$ value, can be attained by manipulating the training data with the following attack strategy: (a.i) the maximum percentage of malicious samples $p_{\max}$ will be injected into the training data; (a.ii) all the injected malicious samples will be attack samples, while no legitimate samples will be affected; and (a.iii) the feature values of the attack samples will be equal to that of the malicious testing samples. To understand the latter assumption,

note that: first, the best case for the adversary is when the attack samples exhibit the same histogram of the payload's byte values as the malicious testing samples, as this intuitively forces the model of the normal traffic to be as similar as possible to the distribution of the testing malicious samples; and, second, this does not imply that the attack samples perform any intrusion, allowing them to avoid detection. In fact, one may construct an attack packet which exhibit the same feature values of an intrusive network packet (i.e., a malicious testing sample), but looses its intrusive functionality, by simply shuffling its payload bytes. Accordingly, for our purposes, we directly set the feature values of the attack samples equal to those of the malicious testing samples, without constructing real network packets.

2) Data model. Since testing data is not affected by the considered causative attack, we set $p_{\mathrm{ts}}(\mathbf{X}, Y) = p_{\mathcal{D}}(\mathbf{X}, Y)$. We then encode the attack strategy as follows. According to (a.ii), $p_{\mathrm{tr}}(A = \mathrm{T}|Y = \mathrm{L}) = 0$ and $p_{\mathrm{tr}}(A = \mathrm{T}|Y = \mathrm{M}) = 1$. The former assumption implies that $p_{\mathrm{tr}}(\mathbf{X}|Y = \mathrm{L}) = p_{\mathcal{D}}(\mathbf{X}|Y = \mathrm{L})$. Since the training data of one-class classifiers does not contain any malicious samples (see explanation at the beginning of this section), i.e., $p_{\mathrm{tr}}(A = \mathrm{F}|Y = \mathrm{M}) = 0$, the latter assumption implies that the class prior $p_{\mathrm{tr}}(Y = \mathrm{M})$ corresponds exactly to the fraction of attack samples in the training set. Therefore, assumption (a.i) amounts to setting $p_{\mathrm{tr}}(Y = \mathrm{M}) = p_{\max}$. Finally, according to (a.iii), the distribution of attack samples $p_{\mathrm{tr}}(\mathbf{X}|Y = \mathrm{M}, A = \mathrm{T})$ will be defined as the *empirical* distribution of the malicious testing samples, namely, $\mathcal{D}_{\mathrm{TR}}^{\mathrm{M,T}} = \mathcal{D}_{\mathrm{TS}}^{\mathrm{M,F}}$. The distribution $p_{\mathrm{tr}}(\mathbf{X}|Y = \mathrm{L}, A = \mathrm{F}) = p_{\mathcal{D}}(\mathbf{X}|\mathrm{L})$ will also be empirically defined as the set $\mathcal{D}_{\mathrm{TR}}^{\mathrm{L,F}}$.

The definition of the attack scenario and data model is summarized in Table 1 (third column).

3) Construction of TR and TS. We use the data set of [24]. It consists of 1,699,822 legitimate samples (network packets) collected by a web server during five days in 2006, and a publicly available set of 205 malicious samples coming from intrusions which exploit the HTTP protocol [51].[13] To construct TR and TS, we take into account the chronological order of network packets as in Sect. 4.1, and the fact that in this application all the malicious samples in $\mathcal{D}$ must be inserted in the testing set [27], [51]. Accordingly, we set $\mathcal{D}_{\mathrm{TR}}$ as the first 20,000 legitimate packets of day one, and $\mathcal{D}_{\mathrm{TS}}$ as the first 20,000 legitimate samples of day two, plus all the malicious samples. Since this attack does not affect testing samples, we set $\mathrm{TS} = \mathcal{D}_{\mathrm{TS}}$. We then set $\mathcal{D}_{\mathrm{TR}}^{\mathrm{L,F}} = \mathcal{D}_{\mathrm{TR}}$, and $\mathcal{D}_{\mathrm{TR}}^{\mathrm{M,F}} = \mathcal{D}_{\mathrm{TR}}^{\mathrm{L,T}} = \emptyset$. Since the feature vectors of attack samples are identical to those of the malicious samples, as above mentioned, we set $\mathcal{D}_{\mathrm{TR}}^{\mathrm{M,T}} = \mathcal{D}_{\mathrm{TS}}^{\mathrm{M,F}}$ (where the latter set clearly includes the malicious samples in $\mathcal{D}_{\mathrm{TS}}$). The size of TR is initially set to 20,000. We then consider different attack scenarios by increasing the number of attack samples in the training data, up to 40,000 samples
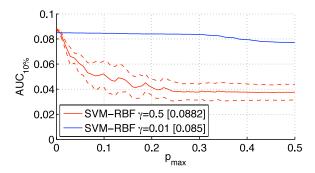
---

13. http://www.i-pi.com/HTTP-attacks-JoCN-2006/

Fig. 5. $\mathrm{AUC}_{10\%}$ as a function of $p_{\max}$, for the ν-SVMs with RBF kernel. The $\mathrm{AUC}_{10\%}$ value for $p_{\max} = 0$, corresponding to classical performance evaluation, is also reported in the legend between square brackets. The standard deviation (dashed lines) is reported only for $\gamma = 0.5$, since it was negligible for $\gamma = 0.01$.

in total, that corresponds to $p_{\max} = 0.5$. For each value of $p_{\max}$, TR is obtained by running Algorithm 1 with the proper inputs.

4) Performance evaluation. Classifier performance is assessed using the $\mathrm{AUC}_{10\%}$ measure, for the same reasons as in Sect. 4.1. The performance under attack is evaluated as a function of $p_{\max}$, as in [17], [28], which reduces to the classical performance evaluation when $p_{\max} = 0$. For the sake of simplicity, we consider only two values of the parameter γ, which clearly point out how design choices based only on classical performance evaluation methods can be unsuitable for adversarial environments.

The results are reported in Fig. 5. In the absence of attacks ($p_{\max} = 0$), the choice $\gamma = 0.5$ appears slightly better than $\gamma = 0.01$. Under attack, the performance for $\gamma = 0.01$ remains almost identical as the one without attack, and starts decreasing very slightly only when the percentage of attack samples in the training set exceeds 30%. On the contrary, for $\gamma = 0.5$ the performance suddenly drops as $p_{\max}$ increases, becoming lower than the one for $\gamma = 0.01$ when $p_{\max}$ is as small as about 1%. The reason for this behavior is the following. The attack samples can be considered outliers with respect to the legitimate training samples, and, for large γ values of the RBF kernel, the SVM discriminant function tends to overfit, forming a "peak" around each individual training sample. Thus, it exhibits relatively high values also in the region of the feature space where the attack samples lie, and this allows many of the corresponding testing intrusions. Conversely, this is not true for lower γ values, where the higher spread of the RBF kernel leads to a smoother discriminant function, which exhibits much lower values for the attack samples.

According to the above results, the choice of $\gamma = 0.5$, suggested by classical performance evaluation for $p\max = 0$ is clearly unsuitable from the viewpoint of classifier security under the considered attack, unless it is deemed unrealistic that an attacker can inject more than

1% attack samples into the training set. To summarize, the designer should select $\gamma = 0.01$ and trade a small decrease of classification accuracy in the absence of attacks for a significant security improvement.

5 SECURE DESIGN CYCLE: NEXT STEPS

The classical design cycle of a pattern classifier [9] consists of: data collection, data pre-processing, feature extraction and selection, model selection (including the choice of the learning and classification algorithms, and the tuning of their parameters), and performance evaluation. We pointed out that this design cycle disregards the threats that may arise in adversarial settings, and extended the performance evaluation step to such settings. Revising the remaining steps under a security viewpoint remains a very interesting issue for future work. Here we briefly outline how this open issue can be addressed.

If the adversary is assumed to have some control over the data collected for classifier training and parameter tuning, a filtering step to detect and remove attack samples should also be performed (see, e.g., the *data sanitization* method of [27]).

Feature extraction algorithms should be designed to be robust to sample manipulation. Alternatively, features which are more difficult to be manipulated should be used. For instance, in [52] inexact string matching was proposed to counteract word obfuscation attacks in spam filtering. In biometric recognition, it is very common to use additional input features to detect the presence ("liveness detection") of attack samples coming from spoofing attacks, i.e., fake biometric traits [53]. The adversary could also undermine feature selection, e.g., to force the choice of a set of features that are easier to manipulate, or that are not discriminant enough with respect to future attacks. Therefore, feature selection algorithms should be designed by taking into account not only the discriminant capability, but also the robustness of features to adversarial manipulation.

Model selection is clearly the design step that is more subject to attacks. Selected algorithms should be robust to causative and exploratory attacks. In particular, robust learning algorithms should be adopted, if no data sanitization can be performed. The use of robust statistics has already been proposed to this aim; in particular, to devise learning algorithms robust to a limited amount of data contamination [14], [29], and classification algorithms robust to specific exploratory attacks [6], [23], [26], [32].

Finally, a secure system should also guarantee the *privacy* of its users, against attacks aimed at stealing confidential information [14]. For instance, privacy preserving methods have been proposed in biometric recognition systems to protect the users against the so-called hill-climbing attacks, whose goal is to get information about the users' biometric traits [41], [53]. Randomization of some classifiers parameters has been also proposed to preserve privacy in [14], [54].

6 CONTRIBUTIONS, LIMITATIONS AND OPEN ISSUES

In this paper we focused on empirical security evaluation of pattern classifiers that have to be deployed in adversarial environments, and proposed how to revise the classical performance evaluation design step, which is not suitable for this purpose.

Our main contribution is a framework for empirical security evaluation that formalizes and generalizes ideas from previous work, and can be applied to different classifiers, learning algorithms, and classification tasks. It is grounded on a formal model of the adversary, and on a model of data distribution that can represent all the attacks considered in previous work; provides a systematic method for the generation of training and testing sets that enables security evaluation; and can accommodate application-specific techniques for attack simulation. This is a clear advancement with respect to previous work, since without a general framework most of the proposed techniques (often tailored to a given classifier model, attack, and application) could not be directly applied to other problems.

An intrinsic limitation of our work is that security evaluation is carried out empirically, and it is thus data-dependent; on the other hand, model-driven analyses [12], [17], [38] require a full analytical model of the problem and of the adversary's behavior, that may be very difficult to develop for real-world applications. Another intrinsic limitation is due to fact that our method is not application-specific, and, therefore, provides only high-level guidelines for simulating attacks. Indeed, detailed guidelines require one to take into account application-specific constraints and adversary models. Our future work will be devoted to develop techniques for simulating attacks for different applications.

Although the design of secure classifiers is a distinct problem than security evaluation, our framework could be also exploited to this end. For instance, simulated attack samples can be included into the training data to improve security of *discriminative* classifiers (e.g., SVMs), while the proposed data model can be exploited to design more secure *generative* classifiers. We obtained encouraging preliminary results on this topic [32].

ACKNOWLEDGMENTS

REFERENCES

[1] R. N. Rodrigues, L. L. Ling, and V. Govindaraju, "Robustness of multimodal biometric fusion methods against spoof attacks," *J. Vis. Lang. Comput.*, vol. 20, no. 3, pp. 169–179, 2009.

[2] P. Johnson, B. Tan, and S. Schuckers, "Multimodal fusion vulnerability to non-zero effort (spoof) imposters," in *IEEE Int'l Workshop on Inf. Forensics and Security*, 2010, pp. 1–5.

[3] P. Fogla, M. Sharif, R. Perdisci, O. Kolesnikov, and W. Lee, "Polymorphic blending attacks," in *Proc. 15th Conf. on USENIX Security Symp.* CA, USA: USENIX Association, 2006.

[4] G. L. Wittel and S. F. Wu, "On attacking statistical spam filters," in *1st Conf. on Email and Anti-Spam*, CA, USA, 2004.

[5] D. Lowd and C. Meek, "Good word attacks on statistical spam filters," in *2nd Conf. on Email and Anti-Spam*, CA, USA, 2005.

[6] A. Kolcz and C. H. Teo, "Feature weighting for improved classifier robustness," in *6th Conf. on Email and Anti-Spam*, CA, USA, 2009.

[7] D. B. Skillicorn, "Adversarial knowledge discovery," *IEEE Intell. Syst.*, vol. 24, pp. 54–61, 2009.

[8] D. Fetterly, "Adversarial information retrieval: The manipulation of web content," *ACM Computing Reviews*, 2007.

[9] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification.* Wiley-Interscience Publication, 2000.

[10] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, "Adversarial classification," in *10th ACM SIGKDD Int'l Conf. on Knowl. Discovery and Data Mining*, WA, USA, 2004, pp. 99–108.

[11] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *Proc. Symp. Inf., Computer and Commun. Sec. (ASIACCS).* NY, USA: ACM, 2006, pp. 16–25.

[12] A. A. Cárdenas and J. S. Baras, "Evaluation of classifiers: Practical considerations for security applications," in *AAAI Workshop on Evaluation Methods for Machine Learning*, MA, USA, 2006.

[13] P. Laskov and R. Lippmann, "Machine learning in adversarial environments," *Machine Learning*, vol. 81, pp. 115–119, 2010.

[14] L. Huang, A. D. Joseph, B. Nelson, B. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *4th ACM Workshop on Artificial Intelligence and Security*, IL, USA, 2011, pp. 43–57.

[15] M. Barreno, B. Nelson, A. Joseph, and J. Tygar, "The security of machine learning," *Machine Learning*, vol. 81, pp. 121–148, 2010.

[16] D. Lowd and C. Meek, "Adversarial learning," in *Proc. 11th ACM SIGKDD Int'l Conf. on Knowl. Discovery and Data Mining*, A. Press, Ed., IL, USA, 2005, pp. 641–647.

[17] P. Laskov and M. Kloft, "A framework for quantitative security analysis of machine learning," in *Proc. 2nd ACM Workshop on Security and Artificial Intelligence.* NY, USA: ACM, 2009, pp. 1–4.

[18] P. Laskov and R. Lippmann, Eds., *NIPS Workshop on Machine Learning in Adversarial Environments for Computer Security*, 2007. [Online]. Available: http://mls-nips07.first.fraunhofer.de/

[19] A. D. Joseph, P. Laskov, F. Roli, and D. Tygar, Eds., *Dagstuhl Perspectives Workshop on Mach. Learning Methods for Computer Sec.*, 2012. [Online]. Available: http://www.dagstuhl.de/12371/

[20] A. M. Narasimhamurthy and L. I. Kuncheva, "A framework for generating data to simulate changing environments," in *Artificial Intell. and Applications.* IASTED/ACTA Press, 2007, pp. 415–420.

[21] S. Rizzi, "What-if analysis," *Enc. of Database Systems*, pp. 3525–3529, 2009.

[22] J. Newsome, B. Karp, and D. Song, "Paragraph: Thwarting signature learning by training maliciously," in *Recent Advances in Intrusion Detection*, ser. LNCS. Springer, 2006, pp. 81–105.

[23] A. Globerson and S. T. Roweis, "Nightmare at test time: robust learning by feature deletion," in *Proc. 23rd Int'l Conf. on Machine Learning.* ACM, 2006, pp. 353–360.

[24] R. Perdisci, G. Gu, and W. Lee, "Using an ensemble of one-class SVM classifiers to harden payload-based anomaly detection systems," in *Int'l Conf. Data Mining.* IEEE CS, 2006, pp. 488–498.

[25] S. P. Chung and A. K. Mok, "Advanced allergy attacks: does a corpus really help," in *Recent Advances in Intrusion Detection*, ser. RAID '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 236–255.

[26] Z. Jorgensen, Y. Zhou, and M. Inge, "A multiple instance learning strategy for combating good word attacks on spam filters," *Journal of Machine Learning Research*, vol. 9, pp. 1115–1146, 2008.

[27] G. F. Cretu, A. Stavrou, M. E. Locasto, S. J. Stolfo, and A. D. Keromytis, "Casting out demons: Sanitizing training data for anomaly sensors," in *IEEE Symp. on Security and Privacy.* CA, USA: IEEE CS, 2008, pp. 81–95.

[28] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia, "Exploiting machine learning to subvert your spam filter," in *Proc. 1st Workshop on Large-Scale Exploits and Emergent Threats*. CA, USA: USENIX Association, 2008, pp. 1–9.

[29] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. D. Tygar, "Antidote: understanding and defending against poisoning of anomaly detectors," in *Proc. 9th ACM SIGCOMM Internet Measurement Conf.*, ser. IMC '09. NY, USA: ACM, 2009, pp. 1–14.

[30] M. Kloft and P. Laskov, "Online anomaly detection under adversarial impact," in *Proc. 13th Int'l Conf. on Artificial Intell. and Statistics*, 2010, pp. 405–412.

[31] O. Dekel, O. Shamir, and L. Xiao, "Learning to classify with missing and corrupted features," *Machine Learning*, vol. 81, pp. 149–178, 2010.

[32] B. Biggio, G. Fumera, and F. Roli, "Design of robust classifiers for adversarial environments," in *IEEE Int'l Conf. on Systems, Man, and Cybernetics*, 2011, pp. 977–982.

[33] ——, "Multiple classifier systems for robust classifier design in adversarial environments," *Int'l Journal of Machine Learning and Cybernetics*, vol. 1, no. 1, pp. 27–41, 2010.

[34] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and F. Roli, "Bagging classifiers for fighting poisoning attacks in adversarial environments," in *Proc. 10th Int'l Workshop on Multiple Classifier Systems*, ser. LNCS, vol. 6713. Springer-Verlag, 2011, pp. 350–359.

[35] B. Biggio, G. Fumera, F. Roli, and L. Didaci, "Poisoning adaptive biometric systems," in *Structural, Syntactic, and Statistical Pattern Recognition*, ser. LNCS, vol. 7626. Springer, 2012, pp. 417–425.

[36] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proc. 29th Int'l Conf. on Machine Learning*, 2012.

[37] M. Kearns and M. Li, "Learning in the presence of malicious errors," *SIAM J. Comput.*, vol. 22, no. 4, pp. 807–837, 1993.

[38] A. A. Cárdenas, J. S. Baras, and K. Seamon, "A framework for the evaluation of intrusion detection systems," in *Proc. IEEE Symp. on Security and Privacy*. DC, USA: IEEE CS, 2006, pp. 63–77.

[39] B. Biggio, G. Fumera, and F. Roli, "Multiple classifier systems for adversarial classification tasks," in *Proc. 8th Int'l Workshop on Multiple Classifier Systems*, ser. LNCS, vol. 5519. Springer, 2009, pp. 132–141.

[40] M. Brückner, C. Kanzow, and T. Scheffer, "Static prediction games for adversarial learning problems," *J. Mach. Learn. Res.*, vol. 13, pp. 2617–2654, 2012.

[41] A. Adler, "Vulnerabilities in biometric encryption systems," in *5th Int'l Conf. on Audio- and Video-Based Biometric Person Authentication*, ser. LNCS, vol. 3546. NY, USA: Springer, 2005, pp. 1100–1109.

[42] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*. New York: Chapman & Hall, 1993.

[43] H. Drucker, D. Wu, and V. N. Vapnik, "Support vector machines for spam categorization," *IEEE Trans. on Neural Networks*, vol. 10, no. 5, pp. 1048–1054, 1999.

[44] F. Sebastiani, "Machine learning in automated text categorization," *ACM Comput. Surv.*, vol. 34, pp. 1–47, 2002.

[45] C.-C. Chang and C.-J. Lin, "LibSVM: a library for support vector machines," 2001. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[46] K. Nandakumar, Y. Chen, S. C. Dass, and A. Jain, "Likelihood ratio-based biometric score fusion," *IEEE Trans. on Pattern Analysis and Machine Intell.*, vol. 30, pp. 342–347, February 2008.

[47] B. Biggio, Z. Akhtar, G. Fumera, G. Marcialis, and F. Roli, "Robustness of multi-modal biometric verification systems under realistic spoofing attacks," in *Int'l Joint Conf. on Biometrics*, 2011, pp. 1–6.

[48] B. Biggio, Z. Akhtar, G. Fumera, G. L. Marcialis, and F. Roli, "Security evaluation of biometric authentication systems under real spoofing attacks," *IET Biometrics*, vol. 1, no. 1, pp. 11–24, 2012.

[49] K. Wang and S. J. Stolfo, "Anomalous payload-based network intrusion detection," in *RAID*, ser. LNCS, vol. 3224. Springer, 2004, pp. 203–222.

[50] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, "New support vector algorithms," *Neural Comput.*, vol. 12, no. 5, pp. 1207–1245, 2000.

[51] K. Ingham and H. Inoue, "Comparing anomaly detection techniques for http," in *Recent Advances in Intrusion Detection*, ser. LNCS. Springer, 2007, pp. 42–62.

[52] D. Sculley, G. Wachman, and C. E. Brodley, "Spam filtering using inexact string matching in explicit feature space with on-line linear classifiers," in *15th Text Retrieval Conf.* NIST, 2006.

[53] S. Z. Li and A. K. Jain, Eds., *Enc. of Biometrics*. Springer US, 2009.

[54] B. Biggio, G. Fumera, and F. Roli, "Adversarial pattern classification using multiple classifiers and randomisation," in *Structural, Syntactic, and Statistical Pattern Recognition*, ser. LNCS, vol. 5342. FL, USA: Springer-Verlag, 2008, pp. 500–509.

Battista Biggio received the M. Sc. degree in Electronic Eng., with honors, and the Ph. D. in Electronic Eng. and Computer Science, respectively in 2006 and 2010, from the University of Cagliari, Italy. Since 2007 he has been working for the Dept. of Electrical and Electronic Eng. of the same University, where he holds now a postdoctoral position. From May 12th, 2011 to November 12th, 2011, he visited the University of Tübingen, Germany, and worked on the security of machine learning algorithms to contamination of training data. His research interests currently include: secure / robust machine learning and pattern recognition methods, multiple classifier systems, kernel methods, biometric authentication, spam filtering, and computer security. He serves as a reviewer for several international conferences and journals, including Pattern Recognition and Pattern Recognition Letters. Dr. Biggio is a member of the IEEE Institute of Electrical and Electronics Engineers (Computer Society), and of the Italian Group of Italian Researchers in Pattern Recognition (GIRPR), affiliated to the International Association for Pattern Recognition.

Giorgio Fumera received the M. Sc. degree in Electronic Eng., with honors, and the Ph.D. degree in Electronic Eng. and Computer Science, respectively in 1997 and 2002, from the University of Cagliari, Italy. Since February 2010 he is Associate Professor of Computer Eng. at the Dept. of Electrical and Electronic Eng. of the same University. His research interests are related to methodologies and applications of statistical pattern recognition, and include multiple classifier systems, classification with the reject option, adversarial classification and document categorization. On these topics he published more than sixty papers in international journal and conferences. He acts as reviewer for the main international journals in this field, including IEEE Trans. on Pattern Analysis and Machine Intelligence, Journal of Machine Learning Research, and Pattern Recognition. Dr. Fumera is a member of the IEEE Institute of Electrical and Electronics Engineers (Computer Society), of the Italian Association for Artificial Intelligence (AI*IA), and of the Italian Group of Italian Researchers in Pattern Recognition (GIRPR), affiliated to the International Association for Pattern Recognition.

Fabio Roli received his M. Sc. degree, with honors, and Ph. D. degree in Electronic Eng. from the University of Genoa, Italy. He was a member of the research group on Image Processing and Understanding of the University of Genoa, Italy, from 1988 to 1994. He was adjunct professor at the University of Trento, Italy, in 1993 and 1994. In 1995, he joined the Dept. of Electrical and Electronic Eng. of the University of Cagliari, Italy, where he is now professor of computer engineering and head of the research group on pattern recognition and applications. His research activity is focused on the design of pattern recognition systems and their applications to biometric personal identification, multimedia text categorization, and computer security. On these topics, he has published more than two hundred papers at conferences and on journals. He was a very active organizer of international conferences and workshops, and established the popular workshop series on multiple classifier systems. Dr. Roli is a member of the governing boards of the International Association for Pattern Recognition and of the IEEE Systems, Man and Cybernetics Society. He is Fellow of the IEEE, and Fellow of the International Association for Pattern Recognition.