Towards a Theory of Intrusion Detection*

Giovanni Di Crescenzo, Abhrajit Ghosh, and Rajesh Talpade

Telcordia Technologies, Piscataway, NJ, USA
{giovanni, aghosh, rrt}@research.telcordia.com

Abstract. We embark into theoretical approaches for the investigation of intrusion detection schemes. Our main motivation is to provide rigorous security requirements for intrusion detection systems that can be used by designers of such systems. Our model captures and generalizes well-known methodologies in the intrusion detection area, such as anomaly-based and signature-based intrusion detection, and formulates security requirements based on both well-known complexity-theoretic notions and well-known notions in cryptography (such as computational indistinguishability).

Under our model, we present two efficient paradigms for intrusion detection systems, one based on nearest neighbor search algorithms, and one based on both the latter and clustering algorithms. Under formally specified assumptions on the representation of network traffic, we can prove that our two systems satisfy our main security requirement for an intrusion detection system. In both cases, while the potential truth of the assumption rests on heuristic properties of the representation of network traffic (which is hard to avoid due to the unpredictable nature of external attacks to a network), the proof that the systems satisfy desirable detection properties is rigorous and of probabilistic and algorithmic nature. Additionally, our framework raises open questions on intrusion detection systems that can be rigorously studied. As an example, we study the problem of arbitrarily and efficiently extending the detection window of any intrusion detection system, which allows the latter to catch attack sequences interleaved with normal traffic packet sequences. We use combinatoric tools such as time and space-efficient covering set systems to present provably correct solutions to this problem.

1 Introduction

Informally, an Intrusion Detection system is a system for raising attention towards potential misbehaviors of the system caused by external adversaries. We could think of a 'burglar alarm' in the real world as the physical analogue of an intrusion detection system in the computerized world. (Just as a burglar alarm in the real world, Intrusion Detection only deals with discovering that an intrusion might have happened into a network. A number of additional aspects related to intrusions, such as intrusion avoidance; that is, augmenting systems so to have a lower likelihood of an external attacker that successfully performs an intrusion; or intrusion tolerance; that is, augmenting systems

* The research was supported by Telcordia and NSA/ARDA under AFRL Contract F30602-03-C-0239. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of NSA/ARDA.

S. De Capitani di Vimercati et al. (Eds.): ESORICS 2005, LNCS 3679, pp. 267–286, 2005.

so that the intended system behavior does not change even after an intrusion; are the subject of study of different research areas.)

Intrusion Detection is a very active and important research area in the Security literature. We won't attempt to survey or categorize the research in this area, but we note that the origin of the problem is often attributed to [1] and several taxonomies and surveys can be found, for instance, in [3,14,15,16]. Often all techniques in known intrusion detection systems are abstracted as falling under two important principles: *anomaly detection*, according to which traffic significantly different from normal ones can be interpreted as likely to be an attack, and *signature detection*, (also called *misuse detection* or *rule-based detection*), according to which traffic significantly similar to known attack traffic can be interpreted as likely to be the same attack. Both principles offer advantages and disadvantages, and many recent systems combine the two principles, rather than specifically choosing one of them.

Despite the large amount of research in this area, no established common framework exists for the design and analysis of intrusion detection systems. A typical research paper in the area proceeds describing some new ideas for detecting intrusions and justifies their validity by describing a specific implementation experience where both the rate of 'false positives' and the rate of 'false negatives' are low. A notable exception is the seminal paper of [7], which does provide a number of valid and formal guidelines for the design and tools for the analysis of intrusion detection systems. In particular, several papers attribute to [7] the introduction of the anomaly-based detection principle.

OUR MODEL. In this paper we put forward a theoretical framework for a rigorous investigation of intrusion detection systems. Our main motivation is to provide security requirements for intrusion detection systems that can be used to accompany simulation-based approaches in their design and increase the number of properties that can be rigorously proved for such systems. Our framework captures and generalizes the notions of anomaly-based and signature-based intrusion detection. Our security requirements are formulated using cryptographic notions such as computational (in)distinguishability, and analysis tools from probability and complexity theory. Specifically, we define two requirements: sensitivity and detection. The first requirement, "sensitivity", says that a *fixed window* of network traffic entering a system can be alternatively represented so that the output of the representation algorithm behaves quite differently according to whether this traffic comes from normal traffic or from a potentially unknown attack. (We remark that this representation algorithm alone is not sufficient to build an intrusion detection system for a few reasons that we later discuss.) The second requirement, "detection", says that if the representation algorithm satisfies the sensitivity requirement, then a data structure and a classification algorithm should allow to *constructively* detect with high probability any attack among a potentially infinite set of *new attacks* or *variations of known attacks*, and in an *arbitrarily large traffic window*. The difficulty in turning a representation algorithm into data structure and classification algorithms is due to the emphasized text in the previous sentence. According to our model, proving both requirements, possibly under some additional assumption, for a proposed system should give mathematical guarantees that the system is a "satisfactory" Intrusion Detection (ID) system. When coupled with simulation-based investigations on the sensitivity of fixed-window network traffic representations and on the estimation of anomaly-type

or signature-type parameters, our framework promises to give a valuable methodology to allow ID designers to increase their claimed properties about their ID systems. Effectively, our model assumes that simulation-based investigations guarantee certain properties about both fixed-window traffic representation and parameter estimation, are satisfied. After this assumption, however, the detection requirement can be formally proved for a given system. We also provide several validations for our model, including the fact that well-known ID systems very often used in practice (most notably, SNORT [22]) can be easily cast into our formalization; and results from a satisfactory implementation experience.

OUR ID SYSTEMS. Under this framework, we obtain two efficient paradigms for intrusion detection systems, one based on nearest neighbor search algorithms, and one based on both the latter and clustering algorithms. Under formally specified assumptions (both stronger than the sensitivity property, one being more applicable than the other), we can prove that our two systems satisfy our detection requirement for an intrusion detection system. (Due to lack of space, we only briefly discuss our second system.)

OPEN QUESTIONS. We believe that our framework raises a number of important open questions on intrusion detection that can be studied using mathematical and/or algorithmic approaches. As an important example, we study the problem of arbitrarily and efficiently extending the detection window of intrusion detection systems, which allows the latter to catch attack sequences interleaved with normal traffic packet sequences (which was not detected in the previously discussed two systems). We present a construction that works for *any* intrusion detection system and is based on particular versions of known combinatorial tools (Covering Set Systems).

ORGANIZATION OF THE PAPER. In Section 2 we present our new framework and all formal definitions. (Validations of the model are in Appendix A.) In Section 3 we present our ID scheme based on Nearest Neighbor Search algorithms and briefly discuss an extension based on Clustering algorithms. In Section 4 we formulate and study the problem of extending the detection window of intrusion detection schemes.

2 Model and Formal Definitions

In this section we present our formal model and definitions for intrusion detection schemes. We start by presenting the system and attack model, including the scenario, the mechanics and the algorithms involved in an execution of such systems, and then describe the requirements that we would like an intrusion detection to satisfy. Although we concentrate on *network* intrusion detection, our definitions are applicable to *host* instrusion detection, where the traffic analyzed is entering the particular host.

2.1 System and Attack Model

SCENARIO, CONNECTIVITY, ACTION. The scenario we consider is that of a large network, also called *autonomous system* (AS), which may have many points of entry for network traffic, also called the *border gateways* (BG) of the AS. The traffic is generated by external users, and without loss of generality, each user can send traffic to each BG.

We write network traffic as a sequence of atomic *packets*, where each packet can be abstracted as a tuple $p = (sid, time, poe, pl)$, where sid is the identity of the sender, $time$ is a timestamp of the action, poe is the point of entry and pl is the payload. At any time the action in an AS system can be described as a stream of packets entering AS through any of its BG (we will assume for simplicity that all traffic enters through a single BG), where each packet in this stream can trigger an event in the AS.

ATTACK MODEL. Informally, an attack can be any sequence of c packets, for some $c \geq 1$, that successfully alters the state of machines in an AS in order to achieve a specific (malicious) goal. If by Φ_t we denote the state of the AS at time t (this may include items such as available bandwidth resources and the internal state of all hosts within the AS) we can then define a polynomial time computable predicate $\rho(1^n, t, \Phi_t)$, where n is a security parameter (later we clarify how to choose it). More generally, we can then define an *attack* as an efficiently samplable probability distribution A over all packet sequences $ps = (p_1, \ldots, p_l)$, where l is the length of A's first input, and such that the probability that experiment $E(A)$ is not successful, is negligible, (that is, smaller than $1/p(n)$, for all positive polynomials p and all sufficiently large n); and, for any distribution D, the probability experiment $E(D)$ is defined as follows.

1. A sequence $\boldsymbol{p}$ of packets is drawn from distribution D
2. sequence $\boldsymbol{p}$ is sent into the network
3. AS turns into state Φ_t
4. predicate $\rho(1^n, t, \Phi_t)$ evaluates to bit b,

and we say that $E(D)$ is successful if $b = 1$. (Here, an output 0 for ρ is intended to imply that attack A has not been succesfully carried out at time t, and 1 otherwise.)

A *class of attacks* $\mathcal{C}$ may be simply defined as a set of attacks $\{A_1, A_2, A_3, \ldots\}$.

We also define a *normal traffic distribution* (briefly, normal traffic) as an efficiently samplable probability distribution N over the set of (single) packets, such that the probability that experiment $E(N)$ is successful, is negligible.

ALGORITHMS AND ID MECHANICS. We will define an intrusion detection system as a triple of algorithms:

1. A *representation* algorithm $\mathcal{R}$ (typical actions modeled by this algorithm include data filtering, formatting, plotting, feature selection, etc.)
2. a *data structure* algorithm $\mathcal{S}$, (typical actions modeled by this algorithm include data collection, aggregation, classification; knowledge base creation, etc.)
3. a *classification* algorithm $\mathcal{C}$ (typical actions modeled by this algorithm include: detection in all forms, including pattern-based, rule-based, anomaly-based, etc.; response, refinement, information tracing, visualization, etc.).

The execution of the ID system can be divided into two phases: an *initialization phase* and a *detection phase*. Briefly speaking, algorithm $\mathcal{S}$ is run in the initialization phase and algorithm $\mathcal{C}$ is run in the detection phase; both algorithms $\mathcal{C}$ and $\mathcal{S}$ use algorithm $\mathcal{R}$ as a subroutine. Specifically, in the initialization phase, the data structure algorithm uses the representation algorithm to process a stream of data obtained from normal traffic distribution or known attack distributions; the returned output is some data structure that will help in the detection phase. Here we note that the initialization phase assumes that the traffic generated according to such distributions is not subject to an attack, with the

possible exception of simulated known attacks. In the detection phase, the classification algorithm is run on input the data structure and a sequence of traffic packets (possibly subject to a known or new attack), and returns an assessment of whether the input sequence of packets contains an attack (and if so, if this is a new attack or not) or only normal traffic. (We note that this output can be generalized to contain additional information such as an estimate of the probability of either event, etc.) Algorithm $\mathcal{R}$, informally, maps a sequence of data packets entering the AS into a fixed-length tuple, having a more compact form (e.g., a point in a high-dimension space).

2.2 Requirements

REQUIREMENTS. Let n be a security parameter; let N be a normal traffic distribution and let $A_1, \ldots, A_t$ be (known) attack distributions such that $N, A_1, \ldots, A_t$ are all efficiently samplable and with pairwise disjoint supports. We define an *intrusion detection system* IDS as a triple of polynomial time algorithms $\mathcal{R}, \mathcal{S}, \mathcal{C}$ with the following syntax.

1. On input 1^n and a sequence of rw packets $\boldsymbol{p}$, algorithm $\mathcal{R}$ returns a d-tuple $\boldsymbol{r}$.
2. On input 1^n and distributions $N, A_1, \ldots, A_t$ algorithm $\mathcal{S}$ returns a data structure ds of size at most $m[int]$.
3. On input 1^n, a data structure ds, a sequence of $m[det]$ packets $\boldsymbol{p}$, a detection window dw and a class of attacks C, algorithm $\mathcal{C}$ returns a classification value out.

Here, rw is a parameter indicating the window of packets used in a single execution of $\mathcal{R}$ (which we will also call the *representation window* and is normally considered a small value); $m[init]$ is a parameter indicating the length of the stream of packets used in the initialization phase; $m[det]$ is a parameter indicating the length of the stream of packets used in the detection phase, to be classified by $\mathcal{S}$ (which is normally considered an arbitrarily large, but polynomial in n and rw, value), and dw is a parameter indicating the maximum distance between the first and last packet of an attack sequence within the stream of packets used in the detection phase. In general, rw, d, $m[init]$, $m[det]$ and dw are all bounded by a polynomial in n; a typical setting would be $rw = O(n)$, $d = O(1)$, $m[init] = n^a$, $m[det] = n^b$, $rw \leq dw \leq m[det]$, for potentially large constants $a, b > 1$. Furthermore, IDS can satisfy the following two requirements of *sensitivity* and *detection*.

Sensitivity. Informally, we would like the output tuple of the representation algorithm to capture differences between normal traffic and attack traffic in its small input packet sequence. Capturing these differences is formalized using the notion of computational distinguishability (a particular strong negation of the notion of computational indistinguishability of [12,24], a notion very frequently used in Cryptography), and specifically by requiring distinguishability with respect to a single sample of the distributions.

Formally, we first recall (an adaptation of) the definition of computational distinguishability: Let t, q be positive integers and $\epsilon \in [0, 1]$. We say that two distributions A, B are (t, q, ϵ)-*distinguishable* if there exists a probabilistic algorithm E running in time t such that $|p_A - p_B| \geq \epsilon$, where, for $C = A, B$, it holds that $p_C = \{x_1, \ldots, x_q \leftarrow C : E(x_1, \ldots, x_q) = 1\}$.

Now, let n be a security parameter. An asymptotic formulation of this definition can be obtained by considering t and q as functions smaller than some polynomial in n.

(By noticeable we mean that it is larger than $1/p(n)$, for some polynomial p and all sufficiently large n.) Specifically, assume $A = \{A_n\}$ and $B = \{B_n\}$ are families of distributions; we say that A and B are *computationally distinguishable* if there exists a probabilistic polynomial (in n) time algorithm E such that for any polynomial (in n) q, it holds that $|p_A - p_B| \geq \epsilon(n)$, where $\epsilon(n)$ is noticeable in n and for $C = A, B$, it holds that $p_C = \{x_1, \ldots, x_q \leftarrow C : E(x_1, \ldots, x_q) = 1\}$.

In practice, we recommend running simulation experiments to determine convenient values for $\epsilon(n)$ and therefore for a security parameter n such that the above inequality $|p_A - p_B| \geq \epsilon(n)$ holds.

We recall that an important result, often used in Cryptography, states that two families of distributions are computationally indistinguishable if and only if they are *single-sample* computationally indistinguishable; that is, they satisfy the latter definition for $q(n) = 1$. In our scenarios, the families of distributions will be normal traffic or attack distributions, and therefore, in general, the algorithm E may not have access to an arbitrary number of of samples from these distributions, especially the attack ones (consider the case of an attacker that only tries her attack once). Therefore, our sensitivity definition only considers distinguishability with respect to one sample.

Definition 1. Let A be an attack distribution and N be a normal traffic distribution; also, let t, rw be a positive integers and $\sigma \in [0, 1]$. We say that a representation scheme $\mathcal{R}$ is (t, σ, A)-*sensitive* if distributions D_N, D_A are $(t, 1, \sigma)$-distinguishable, where:

$$D_N = \{p_1, \ldots, p_{rw} \leftarrow N(1^{rw}); r \leftarrow \mathcal{R}(p_1, \ldots, p_{rw}) : r\}$$
$$D_A = \{(a_1, \ldots, a_{rw}) \leftarrow A(1^{rw}); r \leftarrow \mathcal{R}(a_1, \ldots, a_{rw}) : r\}$$

Furthermore, let C be a class of distributions. We say that a representation scheme $\mathcal{R}$ is (t, σ, C)-*sensitive* if it is (t, σ, A)-sensitive for all distributions A in class C.

In the asymptotic formulation, n is a security parameter, A and N are families of distributions and we say that a representation scheme $\mathcal{R}$ is C-*sensitive* if the distributions D_N and D_A are single-sample computationally distinguishable for all A in class C, where:

$$D_N = \{p_1, \ldots, p_{rw} \leftarrow N_n(1^{rw}); r \leftarrow \mathcal{R}(1^n, p_1, \ldots, p_{rw}) : r\}$$
$$D_A = \{(a_1, \ldots, a_{rw}) \leftarrow A_n(1^{rw}); r \leftarrow \mathcal{R}(1^n, a_1, \ldots, a_{rw}) : r\}.$$

Finally, we say that an *intrusion detection system* $\mathrm{IDS} = (\mathcal{R}, \mathcal{S}, \mathcal{C})$ is C-*sensitive* if so is its representation algorithm $\mathcal{R}$.

For $i = 1, \ldots, rw$, let pos_i be the index $ind \in \{1, \ldots, m[det]\}$ such that $q_{ind} = a_i$, where q_{ind} is the ind-th packet received during the detection phase. We will also say that IDS has *detection window* dw if it holds that $pos_{rw} - pos_1 \leq dw$.

We remark that if a representation scheme is (t, σ, C)-sensitive for "good" parameters, this implies both that the representation has not significantly obscured the information necessary to detect attacks in class C, and that such information was originally present in the observed packet sequence (an obviously minimal feasibility assumption for intrusion detection). The algorithm E may be viewed as an ideal (perfect) analysis system for detecting attacks in class C using $\mathcal{R}$, as described later. While we will not expect

to design such an E for any attack on a given system, we will address the problem of using an estimation for such an algorithm E to detect that a given system is under a certain (known or unknown) attack.

Detection. The only property of the representation algorithm is that the *fixed-window* behavior between attack and normal traffic is different on its output, without clarifying anything about the *nature of this difference*, or any *constructive algorithm to distinguish* which of two different outputs is of which type. Instead, we would like the data structure algorithm and the classification algorithm to directly provide "good enough" detection properties on *arbitrarily large* traffic sequences as long as the representation algorithm has "good enough" sensitivity properties on *small* and *fixed* traffic sequences. This conditional detection requirement is captured by the following game. In a first phase, the data structure algorithm is given access to a stream of m packets $\boldsymbol{p}$ and can run the representation algorithm on inputs of length rw; furthermore, it is allowed to query both the normal traffic distribution N and several (known) attack distributions $A_1, \ldots, A_t$, for some t polynomial in the security parameter n. At the end of this phase, it returns a data structure ds. Now, a sequence of dw packets $\boldsymbol{q}$ are somehow generated and the classification algorithm returns an output out saying if $\boldsymbol{q}$ contains a sample from one of the known attacks $A_1, \ldots, A_t$, or a different (unknown) attack A or no attack at all. The intrusion detection system is successful if this classification is correct.

First, we define the probabilistic experiment in the initialization phase: Let $\boldsymbol{p}$ be the sequence of m packets in this phase, let $A_1, \ldots, A_t$ be known attacks and let N denote the normal traffic distribution over single packets; we can define

$$Init(1^m) \;=\; \{ds \leftarrow \mathcal{S}^{N,A_1,\ldots,A_t,\mathcal{R}}(\boldsymbol{p})\},$$

where the notation $\mathcal{S}^{D_1,\ldots,D_k}$ means that algorithm $\mathcal{S}$ can generate several independent samples from distributions $D_1, \ldots, D_k$.

Now we consider the detection phase; let $\boldsymbol{q}$ be the sequence of dw packets generated in this phase, and let $A \equiv A_0$ be a possibly unknown attack different from $A_1, \ldots, A_t$; we say that string $s = (s_0, \ldots, s_t) \in \{0,1\}^{t+1}$ is *A-correct* if $s_i = 1$ if and only if $\boldsymbol{q}$ contains a tuple of packets in the support of distribution A_i, for $i = 0, 1, \ldots, t$. We are now ready to give a formal definition of the detection property.

Definition 2. Let A be a (potentially unknown) attack, let t be a positive integer and let $\delta \in [0,1]$. We say that an intrusion detection system $\mathrm{IDS} = (\mathcal{R}, \mathcal{S}, \mathcal{C})$ is a (t, δ, A)-*detector* if for any packet sequence $\boldsymbol{q}$, it holds that $\pi(A, \boldsymbol{q}) \geq \delta$, where we define probability $\pi(A, \boldsymbol{q})$ as

$$\mathrm{Prob}\left[ds \leftarrow \mathcal{S}^{\mathcal{R}}(1^{m[init]}); out \leftarrow \mathcal{C}^{\mathcal{R}}(1^n, ds, \boldsymbol{q}, A) \;:\; out \text{ is } A\text{-correct} \right].$$

Furthermore, let C be a class of distributions. We say that an intrusion detection scheme $\mathrm{IDS} = (\mathcal{R}, \mathcal{S}, \mathcal{C})$ is a (t, δ, C)-*detector* if it is (t, δ, A)-detector for all distributions A in class C.

In the asymptotic formulation, we let n be a security parameter, and C be a class of families of distributions and we say that an intrusion detection system $\mathrm{IDS} = (\mathcal{R}, \mathcal{S}, \mathcal{C})$ is a C-*detector* if for t polynomial in n, for any $A \in C$ and any $\boldsymbol{q}$, it holds that $\pi(A, \boldsymbol{q}) \geq \delta$, for some δ noticeable in n.

We remark that an intrusion detection scheme can be considered a 'good' detector if it achieves a detection probability δ 'close enough' to the sensitivity probability σ associated with the representation algorithm. In other words, the closest δ is to σ, the highest is the detection property of the scheme.

DISCUSSION. We also remark that the sensitivity assumption on the behavior of the representation algorithm $\mathcal{R}$ is a necessary assumption, as otherwise no efficient distinguisher between a normal traffic distributions and an attack distribution exists and therefore no pair of algorithms $\mathcal{S}, \mathcal{C}$ can be a detector. Formally, this implies the following

Proposition 1. Let n be a security parameter and A be an attack distribution. Also, let $\mathcal{R}$ be a representation algorithm and assume that $\mathcal{R}$ is not (t, σ, A)-sensitive for t polynomial in n and σ noticeable in n. Then, in our model, there exist no algorithms $\mathcal{S}, \mathcal{C}$ such that the ID system $(\mathcal{R}, \mathcal{S}, \mathcal{C})$ is an $(A, \mathcal{R})$-detector.

Model validation arguments can be found in Appendix A. We do note that our approach in formulating model and security requirements has been quite minimalistic and we have made a number of simplifications. Indeed, we believe we have addressed the most basic possible variant of the intrusion detection problem. We do believe that our model will allow in the future a much easier modeling of more elaborated variants, currently studied in the Intrusion Detection literature.

ANALYSIS METHODOLOGY. Given the above definitions of sensitivity and detection, an ideal methodology to analyze an intrusion detection system in our model would prove that a given ID scheme satisfies:

1. the sensitivity requirement (for some appropriate parameter values)
2. the detection requirement (for some appropriate parameter values) under the assumption that it satisfies the sensitivity requirement.

Clearly, 1) and 2) imply that the given ID scheme satisfies the detection requirement. A mathematical proof that an intrusion detection system satisfies the sensitivity requirement seems hard to obtain, even in a formal model, due to the unpredictable nature of a generic unknown attack. Validating the sensitivity of a representation algorithm is therefore left to simulation-based analysis. However, once a heuristic representation algorithm $\mathcal{R}$ is assumed to be C-sensitive for a class C of attacks, we consider the major analysis goal in our model to formally prove that a certain classification algorithm $\mathcal{C}$ is a $(C; \mathcal{R})$-detector under this very minimal assumption. In this paper we will get very close to prove this result: specifically, we show that our two schemes are C-detectors under slightly stronger (but believable) versions of the sensitivity assumption. We stress that no simulation-based arguments are used in proving this property for our schemes.

3 An ID Scheme Based on Nearest Neighbor Search

In this section we present our first intrusion detection scheme, using algorithms for the approximate nearest neighbor search problem. We start by reviewing this problem and the properties that an algorithm for this problem has to satisfy to be applicable to our ID

scheme. Then we formulate assumptions on the normal traffic and attack distributions, on the output of the estimation algorithm and on the output returned by a representation algorithm. Finally, we present our ID scheme and observe that it satisfies the detection requirement, as defined in Section 2, under the formulated assumptions. An important property achieved using the nearest neighbor search technique is that of merging and generalizing the anomaly-based and signature-based methodologies into a setting with a well-defined metric. As an example, two traffic flows will be determined to be closer to a signature according to a well-defined distance metric, and we can therefore assign a related confidence on whether each traffic flow is a known attack or not. Analogously, in the anomaly-based case, we can assign a related confidence on whether each traffic flow is an unknown attack or a false positive.

APPROXIMATE NEAREST NEIGHBOR SEARCH. Let VS be a vector space of dimension d and let Δ be some distance function defined over VS. Given a set S of n d-component vectors in VS, an error parameter ϵ, and a d-component vector $q \in VS$, we define the $(1+\epsilon)$-*approximate nearest neighbor* of q as the vector v in S such that $\Delta(q,v) \leq (1+\epsilon) \cdot \Delta(q,w)$, for any $w \in S$. A solution to the *approximate nearest neighbor search* problem is a pair of algorithms $(Init, Search)$ as follows. First, algorithms $Init$ and $Search$ have the following syntax: on input an n-size set S of d-length vectors and parameters ϵ, μ, algorithm $Init$ returns a data structure ds; on input data structure ds, a vector v and parameter ϵ, algorithm $Search$ returns a vector w. Then the problem requires that with probability at least μ the following holds: 1) $w \in S$, and 2) w is a $(1+\epsilon)$-approximate nearest neighbor of v. We note that we impose *efficiency requirements* on algorithms for approximate nearest neighbor search that can be of interest for our constructions of ID schemes. In particular, we will require that algorithm $Init$ runs in time polynomial in n and d, and that algorithm $Search$ runs in time polynomial in d and $\log n$. (This is because of the fact that algorithm $Init$ will be used in off-line mode in the initialization phase while algorithm $Search$ will be used in on-line mode in the detection phase). We also note that the performance of algorithm $Search$ is required to be significantly faster than $\Theta(dn)$, which is the performance of the naive, brute-force, and exact search algorithm.

Although any efficient solution for the approximate nearest neighbor search problem can be used for the design of our ID scheme, for concreteness, we will use the following result from [13].

Lemma 1. [13] There exists (constructively) a pair of algorithms (Init,Search) that solve the approximate nearest neighbor search problem for $VS = \{0,1\}^d$ and Δ equal to the Hamming distance, and has the following efficiency property: $Init$ runs in time $\epsilon^{-2} \cdot poly(dn)$ and $Search$ runs in time $\Theta(\epsilon^{-2} \cdot d \cdot poly(\log(dn)))$.

A SET OF ASSUMPTIONS. We now describe assumptions on the normal traffic and attack distributions, on the output of the estimation algorithm and on the output returned by the representation algorithm. The assumptions about the normal traffic and attack distributions generalize the usual assumptions underlying the basic principles of anomaly detection (for the normal traffic and unknown attack distributions) and signature detection (for the known attack distributions). The assumption about the estimation algorithm is stating that the estimation of the parameters in the previous assumptions is

correct with some (somewhat high) probability. The assumption about the behavior of the representation algorithm is at least as strong as the assumption that the representation algorithm is sensitive, in the sense that if a representation algorithm satisfies this new assumption then it also satisfies the sensitivity definition, as in Section 2 (while it is unclear whether the converse is true). Informally, these assumptions postulate that the representation algorithm returns, given a fixed-length sequence of packets as input, a point in a high-dimensional space, such that any two points belonging to the same distribution, being it normal traffic, a known attack, or a new attack, have 'small' distance, while any two points coming from different distributions have 'large' distance. We now define these assumptions more formally.

Assumption 1. Let N be a normal traffic distribution, let $A_1, \ldots, A_t$ be (known) attack distributions and let A be an (unknown) attack distribution.

A representation algorithm is defined as an algorithm that, on input 1^k and a sequence of at most rw packets $\boldsymbol{p}$, where rw is polynomial in k, returns a d-tuple $\boldsymbol{r}$. We say that *distributions* $N, A_1, \ldots, A_t, A$ *are* $(\delta_n, \delta_a, \delta_1, \ldots, \delta_t)$-*oversensitive*if there exists a vector space VS of dimension d, a distance function Δ over VS and a representation algorithm $\mathcal{R}$ such that, for any $\boldsymbol{p}_1, \boldsymbol{p}_2$, denoting as $r_1, r_2 \in VS$ the values such that $\mathcal{R}(\boldsymbol{p}_1) = r_1$ and $\mathcal{R}(\boldsymbol{p}_2) = r_2$, it holds that $\Delta(r_1, r_2)$ is:

1. $\leq \delta_n$ if and only if $\boldsymbol{p}_1, \boldsymbol{p}_2$ were both returned by distribution N
2. $\leq \delta_a$ if $\boldsymbol{p}_1, \boldsymbol{p}_2$ were returned by distribution A
3. $\leq \delta_i$ if $\boldsymbol{p}_1, \boldsymbol{p}_2$ were returned by distribution A_i, for $i = 1, \ldots, t$.

An estimation algorithm is defined as an algorithm returning $(\delta'_n, \delta'_a, \delta'_1, \ldots, \delta'_t)$ when given as input $(1^k, N, A_1, \ldots, A_t, A, VS, \Delta)$. We say that an estimation algorithm ES is μ-correct if it holds that $|\delta_n - \delta'_n| \leq \mu$, $|\delta_a - \delta'_a| \leq \mu$, and $|\delta_i - \delta'_i| \leq \mu$, for $i = 1, \ldots, t$.

We say that a representation scheme $\mathcal{R}$ is $(A, VS, \Delta, \delta'_n, \delta'_a, \delta'_1, \ldots, \delta'_t)$-*oversensitive* if for any $\boldsymbol{p}_1, \boldsymbol{p}_2$, denoting as $r_1, r_2 \in VS$ the values such that $\mathcal{R}(\boldsymbol{p}_1) = r_1$ and $\mathcal{R}(\boldsymbol{p}_2) = r_2$, it holds that $\Delta(r_1, r_2)$ is:

1. $\leq \delta'_n$ if and only if $\boldsymbol{p}_1, \boldsymbol{p}_2$ were both returned by distribution N
2. $\leq \delta'_a$ if $\boldsymbol{p}_1, \boldsymbol{p}_2$ were returned by distribution A
3. $\leq \delta'_i$ if $\boldsymbol{p}_1, \boldsymbol{p}_2$ were returned by distribution A_i, for $i = 1, \ldots, t$.

Finally, the assumption requires that there exists a μ-correct estimation algorithm for the oversensitivity parameters of distributions $N, A_1, \ldots, A_t, A$, and that the representation algorithm $\mathcal{R}$ is $(A, VS, \Delta, \delta'_n, \delta'_a, \delta'_1, \ldots, \delta'_t)$-*oversensitive*, where $\delta'_n, \delta'_a, \delta'_1, \ldots, \delta'_t$ are the parameters returned by the estimation algorithm.

We note that item 1 in the oversensitivity assumptions is an 'if and only if' as we would like that any point generated from an attack distribution, known or unknown, to have distance larger than parameter δ_n (or δ'_n) from a point generated from a normal traffic distribution.

OUR FIRST ID SCHEME. Let $\delta'_n, \delta'_a, \delta'_1, \ldots, \delta'_t$ be estimations, validated by simulation-based studies, of the parameters $\delta_n, \delta_a, \delta_1, \ldots, \delta_t$ in Assumption 1. Also, let $\mathcal{R}$ be an $(A, VS, \Delta, \delta'_n, \delta'_a, \delta'_1, \ldots, \delta'_t)$-oversensitive representation algorithm with representation window rw, where the oversensitivity assumption is also validated by simulation-

based studies. (Note that this assumption implies the assumptions that distributions $N, A_1, \ldots, A_t, A$ are $(\delta_n, \delta_a, \delta_1, \ldots, \delta_t)$-oversensitive and therefore we do not need to clearly state the latter assumption below.) Moreover, let (Init,Search) be a pair of algorithms for the NNS problem, satisfying Lemma 1. Specifically, on input a set S of d-length vectors and parameter ϵ, algorithm $Init$ returns a data structure ds; on input data structure ds, a vector v and parameter ϵ, algorithm $Search$ returns (with high probability) a vector $w \in S$ such that w is a $(1+\epsilon)$-approximate nearest neighbor of v. We now describe algorithms $\mathcal{S}$ and $\mathcal{C}$ for our first ID scheme IDS_1 (for simplicity, we assume that the detection window satisfies $dw = rw$).

Input to Algorithm $\mathcal{S}$: 1^n, distributions $N, A_1, \ldots, A_t$, algorithm $\mathcal{R}$, and parameters $\epsilon, \delta'_1, \ldots, \delta'_t, \delta'_n, \delta'_a$.

Instructions for Algorithm $\mathcal{S}$:

1. For $i = 1, \ldots, n$,
   for $j = 1, \ldots, rw$,
   uniformly and independently sample $r_{i,j}$ from D
   set $x_i = \mathcal{R}(r_{i,1}, \ldots, r_{i,rw})$
2. For $i = 1, \ldots, t$ and $j = 1, \ldots, n$,
   uniformly and independently sample s_{ij} from A_i
   set $y_{ij} = \mathcal{R}(s_{ij})$
3. Let $S = \{x_i\}_{i=1}^n \cup \{y_{1j}\}_{j=1}^n \cup \ldots \cup \{y_{tj}\}_{j=1}^n$
4. Let $ds = \mathrm{Init}(S, \epsilon)$ and set $ds = ds \cup S$
5. Return: ds.

Input to Algorithm $\mathcal{C}$: $1^n, 1^c$, data structure ds, algorithm $\mathcal{R}$, packets $p_1, \ldots, p_m$, and parameters $\epsilon, \delta'_1, \ldots, \delta'_t, \delta'_n, \delta'_a > 0$, where $m = m[det] = rw^c$

Instructions for Algorithm $\mathcal{C}$:

1. For $\ell = 0, \ldots, m - rw$,
   det $v_\ell = \mathcal{R}(p_{\ell+1}, \ldots, p_{\ell+rw})$
   let $w_\ell = \mathrm{Search}(ds, v_\ell, \epsilon)$
   let S be the set contained in ds such that
   $S = \{x_i\}_{i=1}^n \cup \{y_{1j}\}_{j=1}^n \cup \ldots \cup \{y_{tj}\}_{j=1}^n$
   set $out_h = 0$ for $h = 0, \ldots, t$
   if $w_\ell = y_{ij}$ for some $i \in \{1, \ldots, t\}$ and $j \in \{1, \ldots, n\}$ then
   if $\Delta(w_\ell, y_{ij}) \leq \delta'_i$ then set $out_i = 1$
   else set $out_i = (1, \ell)$
   if $w_\ell = x_j$ for some $j \in \{1, \ldots, n\}$ then
   if $\Delta(w_\ell, x_j) > \delta'_n$ then set $out_0 = (1, \ell)$
2. Return: $(out_0, out_1, \ldots, out_t)$ and halt.

We would like to prove that under the oversensitivity assumption on RS, the system IDS is a successful detector.

By inspection of algorithms $\mathcal{S}, \mathcal{C}$, and by assuming that algorithm $\mathcal{R}$ satisfies Assumption 1, we observe that the successful detection of algorithm $\mathcal{C}$ strictly depends on whether the point w_ℓ returned by algorithm Search is the nearest neighbor of v_ℓ and whether the estimations $\delta'_a, \delta'_n, \delta'_1, \ldots, \delta'_t$ are sufficiently close to $\delta_a, \delta_n, \delta_1, \ldots, \delta_t$.

Specifically, we observe that if point w_ℓ returned by algorithm Search is the *exact* nearest neighbor of v_ℓ and it holds that $\delta'_a = \delta_a$, $\delta'_n = \delta_n$ and $\delta'_i = \delta_i$, for $i = 1, \ldots, t$, then then the output $out = (out_0, out_1, \ldots, out_t)$ is A-correct. Therefore, the probability that out is not A-correct can be bounded, using the union bound, as at most the probability that w_ℓ is not the exact nearest neighbor of v_ℓ for at least one $\ell \in \{1, \ldots, m\}$, plus the probability that the estimations $\delta'_a, \delta'_n, \delta'_1, \ldots, \delta'_t$ are not correct. We finally note that the former probability is at most ϵ by Lemma 1 and the latter probability is at most μ by Assumption 1.

Among all performance metrics of the scheme, we stress the importance of the efficiency of the running time of algorithm $\mathcal{C}$. We then obtain the following

Theorem 1. Let A be an attack distribution, and let $\delta_n, \delta_a, \delta_1, \ldots, \delta_t, \epsilon$ be some parameters > 0, and let $\delta'_n, \delta'_a, \delta'_1, \ldots, \delta'_t$ be the output of a μ-correct estimation algorithm taking as input $(1^k, N, A_1, \ldots, A_t, VS, \Delta)$.
If $\mathcal{R}$ is an $(A, VS, \Delta, \delta'_n, \delta'_a, \delta'_1, \ldots, \delta'_t)$-oversensitive representation algorithm then the scheme $\mathrm{IDS} = (\mathcal{R}, \mathcal{S}, \mathcal{C})$ is a (τ, δ, A)-detector, where $\delta = 1 - \mu - m \cdot \epsilon$, and for any $\tau =$poly(n). Moreover, scheme IDS is efficient as algorithm $\mathcal{S}$ runs in time poly$(n \cdot rw \cdot \epsilon^{-1})$ and algorithm $\mathcal{C}$ runs in time $O(\epsilon^{-2} \cdot rw \cdot$ polylog$(n \cdot rw))$. Furthermore, IDS has detection window $dw = rw$.

We consider a major open problem in the theory of intrusion detection to design ID schemes with assumptions weaker than Assumption 1. (Due to Proposition 1, the ultimate goal would be that of using the sensitivity requirement as a minimal assumption.)

OUR SECOND ID SCHEME. We only briefly mention that our first ID scheme can be generalized using Clustering algorithms and resulting in a second scheme based on a slightly weaker assumption. The idea we use here is in relaxing the assumption is in allowing several distributions (rather than a single one) for normal traffic. As a consequence, it is not true any more that any two points associated to normal traffic have 'small' distance, but it will hold that any such point has 'close' distance from at least one point generated according to at least one of the normal traffic distributions. Since our second scheme is based on weaker assumptions than our first one, the class of attacks that it can detect is strictly larger than the class of attacks of our first scheme, which points at another interesting capability allowed by our model.

4 ID Schemes with Arbitrary-Length Detection Window

In the previous sections we have studied intrusion detection schemes with detection window equal to the representation window. This restriction is, in practice, undesirable as it allows an adversary to perform simple attack strategies that would not be detected by the intrusion detection system. For instance, even for attacks consisting of two packets only, an adversary could send the second packet slightly later than the first packet (precisely, by interleaving between the two packets a number of packets at least as large as the representation window), and the detection window of the system will not contain both packets.

In this section we formally define and study the problem of extending the length of the detection window of an ID scheme. We use combinatorial techniques and apply

them to any ID scheme that satisfies the definition in Section 2. Therefore, when applied to our schemes in Section 3, we obtain ID schemes with extended-length detection window under the same assumptions on the representation algorithm.

More formally, a first formulation of this problem could be the following. Given a generic intrusion detection system $\text{IDS}_1=(\mathcal{R},\mathcal{S},\mathcal{C})$ with representation window rw_1 and detection window $dw_1 = k$, is it possible to construct an intrusion detection system IDS_2 with representation window $rw_2 = rw_1$ and detection window $dw_2 = m$, for any $m = \text{poly}(k)$?

We note that the size of the tuple returned by an attack distribution A is defined to be equal to the length of A's first input, which is set, for convenience of parameters, equal to the representation window rw. More generally, in our problem formulation we would like to capture the situation of the number of effective attack packets being equal to some ℓ such that $1 \leq \ell \leq rw$, which is closer to what expected in practice. Formally, we define an attack distribution A as *ℓ-effective* if, denoting by $\text{Supp}(A, rw)$ the support of distribution A, when run on input 1^{rw}, the following holds: for each tuple $(a_1, \ldots, a_{rw}) \in \text{Supp}(A, rw)$, there exists an ℓ-tuple of indices $i_1, \ldots, i_\ell$ such that all rw-tuples containing $a_{i_1}, \ldots, a_{i_\ell}$ are in $\text{Supp}(A, rw)$. (Here, such ℓ-tuple can be considered as the effective attack witness.)

As a consequence, we will study the following problem. Let C be a class of ℓ-effective attacks. Given a C-sensitive intrusion detection system $\text{IDS}_1 = (\mathcal{R},\mathcal{S},\mathcal{C})$ with representation window rw_1 and detection window $dw_1 = k$, is it possible to construct a C-sensitive intrusion detection system IDS_2 with representation window $rw_2 = rw_1$ and detection window $dw_2 = m$, for any $m = \text{poly}(k)$?

4.1 A Solution Based on Covering Set Systems

We now recall the definition of well-studied combinatorial objects, called covering set systems, and present a generic construction of an intrusion detection system with arbitrary-length detection window from one with a fixed detection window.

Definition 3. Let ℓ, k, m be positive integers. Let S be a set of size m and let $T = \{T_1, \ldots, T_s\}$ be a set of k-size subsets of S. We say that T is an (ℓ, k, m)-*covering set system* for S if for any ℓ-size $S_i \subseteq S$, there exists a subset $T_j \in T$ such that $S_i \subseteq T_j$. The *space efficiency* of the covering set system T is defined to be the size s of T (and can be a function of ℓ, k, m). The *time efficiency* of covering set system T is defined to be the running time (as a function of ℓ, k, m) that an algorithm takes to construct T.

As an example, note that the set of all ℓ-size subsets of S is an (ℓ, k, m)-covering set system for S having both time and space efficiency $\binom{m}{\ell}$. Covering set systems have been studied in several works (see, e.g., [10,11,9,18,21] and references therein), focusing on somewhat different requirements than ours. We also note that a related and dual notion of set systems (in an area also called Turan Theory) has been applied to other areas in Cryptography, such as secret sharing [19] and secure mixnets [6] (works on this notion typically focus on covering set systems for k, m very close to ℓ). We are not aware of other applications of covering set systems in the Security area.

Construction of an IDS with arbitrary detection window. Let C be a class of ℓ-effective attacks, and let $\text{IDS}_1=(\mathcal{R}_1,\mathcal{S}_1,\mathcal{C}_1)$ be a C-sensitive intrusion detection sys-

tem with representation window rw_1 and detection window $dw_1 = k$. Also, let $T = \{T_1, \ldots, T_m\}$ be a (ℓ, k, m)-covering set system for set $S = \{1, \ldots, m\}$. We now define an intrusion detection system IDS$_2$=$(\mathcal{R}_2, \mathcal{S}_2, \mathcal{C}_2)$, with representation window rw_2 and detection window $dw_2 = m$.

Algorithms $\mathcal{R}_2, \mathcal{S}_2$ are defined as equal to $\mathcal{R}_1, \mathcal{S}_1$, respectively. Algorithm $\mathcal{C}_2$ goes as follows. On input a sequence of m packets $p_1, \ldots, p_m$, it runs s times (using independent randomness) $\mathcal{C}_1$, each time on inputs a sequence of packets $s = p_{j_1}, \ldots, p_{j_k}$, where $T_i = \{j_1, \ldots, j_k\}$; we denote as $(out_{i0}, \ldots, out_{it})$ be the output returned by this execution of $\mathcal{C}_1$. Finally, $\mathcal{C}_2$ returns $(out_0, \ldots, out_t)$, where $out_j = \vee_{i=1}^{m} out_{ij}$, for $j = 1, \ldots, t$.

The sensitivity of $\mathcal{C}_2$ can be proved by using the sensitivity of $\mathcal{C}_1$ and the definition of covering set system. (Very roughly, for each ℓ-size effective attack sequence seq, there exists at least one subset in T that will define a sequence of packets seq' that contains seq and is given as input to $\mathcal{C}_1$ that will detect it). The efficiency of IDS$_2$ depends on the efficiency of the construction for the covering set systems. We note that for $\ell = O(1)$ (which is expected in practice) or for just s polynomial in the security parameter, then algorithm $\mathcal{C}_2$ runs in time polynomial in the security parameter and then so does IDS$_2$.

We obtain the following

Theorem 2. Let C be a class of ℓ-effective attacks. Given a C-sensitive intrusion detection system IDS$_1$=$(\mathcal{R}_1, \mathcal{S}_1, \mathcal{C}_1)$ with representation window rw_1 and detection window $dw_1 = k$, and given an (ℓ, k, m)-covering set system for set $S = \{1, \ldots, m\}$ with time efficiency t and space efficiency s, it is possible to construct a C-sensitive intrusion detection system IDS$_2$ $= (\mathcal{R}_2, \mathcal{S}_2, \mathcal{C}_2)$ with representation window $rw_2 = rw_1$ and detection window $dw_2 = m$, for any $m = \text{poly}(k)$, where algorithm $\mathcal{C}_2$ runs in time $O(t + s \cdot \text{time}(\mathcal{C}_1))$.

We note that in the above theorem the efficiency of algorithm $\mathcal{C}_2$ (and therefore, of IDS$_2$) significantly depends on both time and space efficiency of the covering set system. It is then of interest to obtain covering set systems with satisfactory performance on both parameters and yet working for all choices of ℓ, k, m. (Specifically, we are willing to sacrifice optimality with respect to space efficiency in order to achieve generality and satisfactory time efficiency.) Furthermore, of additional interest is the practical requirement that the code to generate such systems is simple. Constructions of covering set systems in the combinatorics and theoretical computer science literature mostly focus on achieving space-optimality, even for possibly limited choice of parameters ℓ, k, m. In the next section we show some constructions that work for all choices of ℓ, k, m, are simple to generate, and are time and space-efficient for $\ell = O(1)$. Improving these constructions to achieve time and space-efficiency for larger values of ℓ is an interesting open problem.

4.2 Constructions of Time-Efficient Covering Set Systems

We define $C(\ell, k, m)$ as the minimum, over all (ℓ, k, m)-covering set systems T, of the space efficiency of T. We recall that a trivial upper bound of $\binom{n}{\ell}$ on $C(\ell, k, m)$ follows by defining a set T_i as an arbitrary extension of the i-th ℓ-size subset of S. Furthermore, we now recall two known lower bounds for $C(\ell, k, m)$. The first bound is simple and

follows by observing that each T_i can at most cover $\binom{k}{\ell}$ distinct subsets of size ℓ from S. The second lower bound is also well-known and due to [20].

Fact 1. It holds that

1. $C(\ell, k, m) \geq \frac{\binom{m}{\ell}}{\binom{k}{\ell}}$
2. $C(\ell, k, m) \geq \lceil \frac{m}{k} \cdot C(\ell - 1, k - 1, m - 1) \rceil$

We ideally would like to define general and time-efficient constructions of T also having space efficiency as close as possible to the above lower bounds. Assuming $\ell = O(1)$ and, for simplicity, k/ℓ equal to an integer, we now define two constructions that meet these bounds up to a constant.

Construction 1:

1. Let $S = \{1, \ldots, m\}$ and $T_1 = \emptyset$.
2. Partition S into k-size disjoint subsets $S_1, \ldots, S_{\lceil m/k \rceil}$
3. For $i = 1, \ldots, \lceil m/k \rceil$,
   partition each S_i into disjoint (k/ℓ)-size subsets $Z_{i,1}, \ldots, Z_{i,\ell}$
4. For each $i_1, \ldots, i_\ell \in \{1, \ldots, \lceil m/k \rceil\}$,
   for each $(a_1, b_1), \ldots, (a_\ell, b_\ell) \in \{(i_j, t) : j, t = 1, \ldots, \ell\}$,
   add $\cup_{i=1}^{\ell} Z_{a_i, b_i}$ to T_1,
5. Return: T_1.

Construction 2:

1. Let $S = \{1, \ldots, m\}$ and $T_2 = \emptyset$.
2. Partition S into (k/ℓ)-size disjoint subsets $S_1, \ldots, S_{\ell \cdot \lceil m/k \rceil}$
3. For each $i_1, \ldots, i_\ell \in \{1, \ldots, \ell \cdot \lceil m/k \rceil\}$,
   add $\cup_{j=1}^{\ell} S_{i_j}$ to T_2,
4. Return: T_2.

The above constructions satisfy the following

Theorem 3. The above two constructions define (ℓ, k, m)-covering set systems T_1, T_2 for arbitrary positive integers ℓ, k, m, with time and space efficiency (t_1, s_1) and (t_2, s_2), respectively, where:

1. $s_1 = \binom{\lceil m/k \rceil}{\ell} \cdot \binom{\ell^2}{\ell}$ and $t_1 = \Theta(s_1)$;
2. $s_2 = \binom{\ell \cdot \lceil m/k \rceil}{\ell}$ and $t_2 = \Theta(s_2)$.

References

1. J. Anderson, *Computer Security Threat Monitoring and Surveillance*, in James P. Anderson Co., Fort Washington, Pa. 1980.
2. S. Axelsson, *The Base-Rate Fallacy and its Implication for the Difficulty of Intrusion Detection*, in Proc. of ACM CCS, 1999.
3. S. Axelsson, *Intrusion Detection Systems: A Survey and Taxonomy*, Technical Report 99-15, Depart. of Computer Engineering, Chalmers University, march 2000.
4. A. Borodin, R. Ostrovsky, and Y. Rabani, *Subquadratic Approximation Algorithms For Clustering Problems in High Dimensional Spaces*, in Proc. of The 31'st ACM Symposium on Theory of Computing (STOC-99)

5. Cisco Flow Collector Overview, http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/nfc/nfc_3_0/nfc_ug/nfcover.pdf
6. Y. Desmedt and K. Kurosawa, *How to Break a Practical Mix and Design a New One*, in Proc. of Eurocrypt 2000, LNCS vol. 1807, Springer.
7. D. E. Denning, *An Intrusion Detection Model*, in IEEE Transactions on Software Engineering, Vol. SE-13, no. 2, pp. 222-232, 1987.
8. M. Esmaili, R. Safavi Naini, and J. Pieprzyk, *Intrusion Detection: A Survey*, in Proc. of ICCC 1995.
9. D. Gordon, *La Jolla Covering Repository*, web site: http://www.ccrwest.org/cover.html.
10. D. Gordon, G. Kuperberg, and O. Patashnik, *New Constructions for Covering Designs*, Journal of Combinatorial Designs, 3 (1995), pp. 269–284.
11. D. Gordon, G. Kuperberg, O. Patashnik, and J. Spencer, *Asymptotically Optimal Covering Designs*, Journal of Combinatorial Theory A, 75 (1996), pp. 220–240.
12. S. Goldwasser, and S. Micali, *Probabilistic Encryption*, in Journal of Computer and System Sciences, vol. 28, n. 2, 1984, pp. 270–299.
13. E. Kushilevitz, R. Ostrovsky, and Y. Rabani, *Efficient Search for Approximate Nearest Neighbor in High Dimensional Spaces*, in Proc. of the 30's ACM Symposium on Theory of Computing (STOC-98)
14. W. Lee, *A Data Mining Framework for Building Intrusion Detection Models*, in Proc. of IEEE Symposium on Security and Privacy 1999.
15. T. Lunt, *Automated Audit Trail Analysis and Intrusion Detection: A Survey*, in Proc. of 11th National Computer Security Conference, 1988.
16. N. McAuliffe, D. Wolcott, L. Schaefer, N. Kelem, B. Hubbard, and T. Haley, *Is Your Computer Being Misused ? A Survey of Current Intrusion Detection System Technology*, in Proc. of 6th IEEE Computer Security Applications Conference, 1990.
17. Netflow, IETF RFC, ftp://ftp.rfc-editor.org/innotes/rfc3954.txt
18. K. Nurmela and P. Ostergard, *Upper Bounds for Covering Designs by Simulated Annealing*, Congressum Numerantium, 96:93–111, 1993.
19. R. Rees, D. R. Stinson, R. Wei and G. H. J. van Rees, *An application of covering designs: Determining the maximum consistent set of shares in a threshold scheme*, Ars Combinatoria 531 (1999), 225-237.
20. J. Schonheim, *On Coverings*, Pacific Journal of Mathematics, 14:1405-1411, 1964
21. C. Colbourn and J. Dinitz, THE CRC HANDBOOK OF COMBINATORIAL DESIGNS, CRC Press, Boca Raton, FL 1996 (see D. R. Stinson, Coverings, pp. 260-265)
22. http://www.snort.org
23. Flowtools public-domain software. http://www.splintered.net/sw/flow-tools/
24. A. Yao, *Theory and Application of Trapdoor Functions*, in Proc. of FOCS 85.
25. A. Ghosh, L. Wong, G. Di Crescenzo and R. Talpade, *Infilter: Predictive Ingress Filtering to Detect IP Spoofed Traffic*, in 2nd International Workshop on Security in Distributed Computing Systems (SDCS 2005).

A Model Validation

We have gone through a few basic steps towards validation of our model.

WELL-KNOWN PERFORMANCE METRIC OF ID SYSTEMS IN OUR MODEL. All natural performance metrics considered in the ID literature have a rigorous definition according to our model, as we discuss in detail in Appendix A. In particular, we discuss false positive rate, detection probability, detection attempt rate, time and space efficiency, data collection stability, data upgrade rate and performance.

WELL-KNOWN ID SYSTEMS IN OUR MODEL. Well-known ID systems very often used in practice can be easily cast into our formalization. We only discuss the notable case of SNORT [22] and show how its major components can be recast in forms of representation, data structure and classification algorithms. Then we discuss how analysis along the lines of Section 3 can be used to argue a number of interesting facts about one or more SNORT instantiations, even beyond just rigorously proving its detection properties. As an example, our model can be used to rigorously evaluate the tradeoff in two different SNORT instantiations between increased set of rules and efficiency performance of the system. We now proceed in slightly greater detail.

A public domain tool and perhaps the most widely deployed ID systems, SNORT [22] can be abstracted in one-line as a signature-based network intrusion detection system. A little more precisely, SNORT is a rule-based ID system, as it allows the definition and update of rules for traffic classification, and it actually provides somewhat sophisticated detection capabilities, such as information about attack 'origin' and attack 'breach type'. A high level definition of SNORT major components is as follows:

1. *Packet Capture Engine:* this uses a certain library to capture traffic datagrams.
2. *Preprocessor Plug-Ins:* they inspect packet data received from the capture engine and decide whether to analyze it or not, and, if yes, whether to generate an alert of a potential attack. They also perform some data filtering to eliminate traffic that may be malicious to the SNORT application itself.
3. *Detection Engine:* this performs basic tests according to a set of internal rules, each of them typically asking to search for a string/value associated with the rule itself and some particular piece of the packet. As for any signature-based ID system, it contains a preliminary phase of data gathering and main rules definition, and an active phase of online traffic classification.
4. *Output Plug-Ins:* they return high-level information of interest to the ID analyst.

We now show how we can simply fit SNORT into our formalization. Specifically, the representation algorithm $\mathcal{R}$ is composed with both the Packet Capture Engine and the Preprocessor Plug-Ins. The data structure algorithm $\mathcal{S}$ is composed with the rule definition part (both in the preliminary and active phase) and the preliminary phase of the Detection Engine. Finally, the classification algorithm $\mathcal{C}$ is composed with the active phase of the Detection Engine as well as the Output Plug-Ins. Technically, it is more appropriate to talk of SNORT as of an ID system suite, rather than a single ID system, as its detection success may significantly change according to how the above 4 components are instantiated. It is clear then that for each instantiation, one could prove a theorem similar in spirit to Theorem 1. One major difference, however, is that, given that the rules used

by any SNORT instantiation fall under the signature detection principle, any SNORT instantiation will only be able to detect attacks A that are among the known attacks $A_1, \ldots, A_t$ (while other schemes including the one given in Section refse-scheme1 combine and generalize the anomaly and signature detection principles.) Still, theorems in our model can be used in order to compare the advantages and disadvantages of different rule sets in different SNORT instantiations. For instances, a very basic question for which our model can provide quantitative answers, is that of evaluating the tradeoff between the convenience of enlarging the set of rules (i.e., using a weaker assumption and obtaining stronger detection results) and the degrade in certain performance metrics (such as running time, detection attempt rate and data upgrade rate).

A similar abstraction can be done for several other well-known signature-based ID systems. We remark that our formalization captures also anomaly-based ID systems (in fact, our system in Section 3 is an hybrid of both approaches: anomaly-based and signature-based).

DESIGN/ANALYSIS PLAN FOR ID SYSTEMS IN OUR MODEL. It is possible to formulate a detailed plan for the design and analysis methodology of ID systems in our model (thus, further elaborating on the discussion at the end of Section 2.2), that automatically integrates simulations and implementation experiences with theoretical analysis. In general, we will consider the following (summarized) step-by-step design and analysis methodology for ID systems:

1. Assumptions about normal traffic distributions and single attacks or attack classes distributions are rigorously formulated in terms of a set PS of parameters.
2. An algorithm ES is defined to produce a set PS' of parameters estimating the parameters in PS
3. Algorithms $\mathcal{R}, \mathcal{S}, \mathcal{C}$ are defined using estimations in PS'.
4. An assumption is made about the estimation property of algorithm ES and the assumption is validated through simulation-based studies.
5. An assumption is made about the sensitivity property of algorithm $\mathcal{R}$ and the assumption is validated through simulation-based studies.
6. The detection property of algorithms $\mathcal{S}, \mathcal{C}$ for the given attack class is mathematically proved under the assumption that $\mathcal{R}$ satisfies the sensitivity property.

Note that we could have included the estimation algorithm in the formalization above but we decided not to do so not to overburden the formalism (alternatively, estimates could be returned by the algorithm $\mathcal{R}$ itself). We underline the highly desirable modularity of this approach: an ID designer can mix-and-match representation and parameter estimation algorithms validated through simulation studies with data structure and classification algorithms that are mathematically proved correct. In the rest of this paper we will concentrate on the latter part: defining data structure and classification algorithms that are mathematically proved correct under the assumption that the associated representation algorithm is sensitive to a given attack or class of attacks. We stress that this is performed for *any* classification algorithm satisfying the sensitivity property and therefore the reader should not expect a simulation-based analysis, but rather a mathematical correctness proof for the detection property of the classification algorithm.

OUR IMPLEMENTATION EXPERIENCE. One implementation in [25] of an ID system (using the system discussed in Section 3) performs quite satisfactorily on several prac-

tical performance metrics (in addition to the desired theoretical properties established here). Specifically, in [25], together with other coauthors, we detail an implementation of a version of our ID system in Section 3, based on Nearest Neighbor Search, as a component of a larger system for the detection of IP spoofed traffic. There we run experiments designed to quantify the ability to detect various kinds of attacks (of both voluminous and stealthy nature), the detection rate, the false positive rate, and the variance with the location of attack sources. Except for pathological cases and very high attack loads, the implementation has a detection rate of about 80 $\%$ and a false positive rate of about 2 $\%$ in testbed experiments using Internet traffic and real cyberattacks. The implementation is compromised of various system level components deployed at various locations within a target network. NetFlow [17] is enabled on Border Routers (BRs) in large IP backbone networks. Flowtools [23] software modules can be deployed at various host nodes within the target network. NetFlow data is transmitted to the flow-tools modules from the BRs. Statistics generated by Flow-tools are then transferred to the analysis software module, which analyzes the data and can provide notification in case abnormal behavior is detected. A full report on some features and results of our implementation can be found in [25].

PERFORMANCE METRICS. We consider several metrics that can help in measuring the performance of an intrusion detection system receiving as input a stream of $m[det]$ packets and formally redefine them in the described model (this is, of course, non necessarily an exhaustive list); finally, we discuss values for these metrics that would imply satisfactory performance of an intrusion detection system.

False Positive Rate. Informally, a *false positive* happens when an alert for an attack is raised in correspondence of a sequence of packets that does not contain any attack. This is one of the most often considered performance metrics, especially in anomaly-based intrusion detection systems, and reducing the rate of false positives in such systems is one of the biggest areas of research for Intrusion Detection. In our formal model, a false positive can be defined as a sequence $\boldsymbol{q}$ of dw packets such that the string $out = (out_0, out_1, \ldots, out_t)$ returned by algorithm $\mathcal{C}$ when run on input $\mathcal{R}, (1^n, ds, \boldsymbol{q}, A)$, satisfies the following: there exists $i \in \{0, \ldots, t\}$ such that $out_i = 1$ and $\boldsymbol{q}$ does not contain a tuple of packets in the support of distribution A. Then the *false positive rate* of an intrusion detection system for sequences up to $m[det]$ packets, is equal to the expected value, over all sequences of length $m[det]$, of the ratio of the number of false positives to the number of sequences of dw packets having nonzero probability of occurrence. Here the probability space is over distributions $N, A, A_1, \ldots, A_t$.

Detection Probability. Informally, the detection probability is the probability that the response from the intrusion detection system is correct, and, clearly, this is the ultimately more interesting parameter. In our formal model, the detection probability with respect an attack A and a sequence $\boldsymbol{q}$ of dw packets is denoted as $\pi(A, \boldsymbol{q})$ and is formally defined in Definition 2.

Detection Attempt Rate. Informally, the detection attempt rate is the frequency with which a detection attempt is being performed. While an ideal system would check in an $m[det]$-packet sequence for every dw-packet subsequence where an attack might appear, more realistic efficiency constraints might prevent the system to do that and

therefore detection attempts would be performed less frequently. Let A be an attack distribution, rw be the representation window of the intrusion detection system and denote as $\boldsymbol{s}$ an $m[det]$-packet stream entering into the network. We define the set of $(A, rw, m[det])$-*candidate sequences* as the set of rw-packet subsequences in $\boldsymbol{s}$ that might contain a tuple in the support of distribution A. The *detection attempt rate* is then the expected value of the ratio of the number of subsequences of (A, rw, m)-candidate sequences for which the output of algorithm $\mathcal{C}$ is A-correct, to the number of all (A, rw, m)-candidate sequences. Here, again, the probability space is over distributions $N, A, A_1, \ldots, A_t$.

Initialization and Detection Time and Space Efficiency. Informally, the initialization (resp., detection) time and space efficiency are the running time and the space complexity of the intrusion detection system during the initialization phase (resp., the detection phase). In our model, we define the *initialization time efficiency* (resp., *initialization space efficiency*) as the running time (resp., storage complexity) of $\mathcal{S}$ as a function of $n, m[init], \sigma, \delta$; we then define the *detection time efficiency* (resp., *detection space efficiency*) as the running time (resp., storage complexity) of $\mathcal{C}$ as a function of $n, m[init], dw, m[det], \sigma, \delta$.

Data Collection Stability. Informally, the data collection stability parameter is the amount of storage that is necessary in the initialization phase in order to guarantee the claimed detection properties of an intrusion detection system for an $m[det]$-packet stream in the detection phase. In our model, we denoted this parameter as a free parameter and defined as the length of the output of algorithm $\mathcal{S}$; in general, it can be set as a function of other parameters $n, \sigma, \delta, dw, m[det]$.

Data Upgrade Rate. Informally, the data upgrade rate denotes how often the data structure is upgraded; at one extreme, a system could periodically discard the previously collected data and rerun the initialization phase; at the other extreme, a system could use every packet received by the network in order to update the data structure. Formally, this rate can be defined as the expected value of $1 -$ the ratio of the number of packets for which an update of ds has not occurred to the length of the packet stream $m[det]$. Here, again, the expected value is over all $m[det]$-packet sequences and the probability space is over distributions $N, A, A_1, \ldots, A_t$.

Satisfactory Performance. Clearly, one would like an intrusion detection system to optimize all the defined performance metrics. We only remark here on two metrics. In terms of detection, as we observe later, algorithm $\mathcal{C}$ cannot find attacks that are not somehow captured by algorithm $\mathcal{R}$; therefore, we would require a satisfactory detection probability to be one that minimizes the difference $\delta - \sigma$. In a complexity-theoretic sense, satisfactory time and space efficiency of an intrusion detection system could be required to be equivalent to all algorithms $\mathcal{R}, \mathcal{S}, \mathcal{C}$ running in time polynomial in the security parameter n. In a more practical setting, we note that algorithm $\mathcal{S}$ is run once and for all in an initialization phase, while algorithms $\mathcal{R}, \mathcal{C}$ are repeatedly run (in an on-line fashion) in the detection phase. Therefore, we specifically require that algorithms $\mathcal{R}, \mathcal{C}$ are significantly more efficient; for instance, that they run in time at most polynomial in $\log n$. (We note that both schemes we propose in this paper satisfy this.)