Probability and Time in Measuring Security

Anatol Slissenko [1]
Laboratory of Algorithmics, Complexity and Logic (LACL), University Paris 12, France [2,3]

Abstract. The paper discusses some possible approaches to measuring security of timed and probabilistic models of systems. We discuss problems concerning the leak of information and the resistance of executions of security policies, and propose quantitative characteristics of security. Algorithmic questions related to the computation of these characteristics are formulated for finite transition models.

Keywords. Probabilistic security models. Timed security models. Information leak. Security policy resistance.

1. Introduction

This paper is an invitation to discuss some algorithmic questions concerning the analysis of computer system security. We will speak about probabilistic models and to some amount about time. We are interested in computing quantitative characteristics of system security, in particular, the quantity of leaked information or to what amount an enforcement of security policy is resistent to possible breaches, e.g., with what minimal probability of breaking a forbidden access the loss is over a given bound.

One can distinguish two aspects in this set of questions: the first one, that has been just mentioned above, is that of analysis of a given system, and the second one is that of synthesis of a system with given characteristics. We dwell upon the first question. The second one will be only mentioned.

The security is very probabilistic in its nature, and evolves in time rather rapidly. Thus, involving probability and time into models seems necessary, especially when we wish to 'measure' the security. Notice that it is a usual demand of security management [1,2] to assess and estimate risk, and it implies implicitly or explicitly to say something about probabilities and time.

The main objection against probabilities in our context is that we do not know them. I would say, more precisely, we do not know them exactly. Moreover, usually we do not have enough experimental observations that would suffice to evaluate probability distributions on the basis of the theory of statistics. Our situation is quite different, from this

[1] Partially supported by Eco-Net project No 08112WJ.
[2] Correspondence to: Informatique, Université Paris 12, 61 av. du Gén. de Gaulle, 94010 Créteil, France Tel.: +33 (0)1 4517 1663; Fax: +33 (0)1 4517 6601; E-mail:slissenko@univ-paris12.fr
[3] Also a member of St.Petersburg Institute of Informatics, Russian Academy of Sciences.

point of view, from the analysis of signal transmission where we often have stationary processes, and long series of observations are available. However, one can notice that probabilistic and timed models can be very useful if we have reasonable approximate ideas about probability and time, and this is the case when we have a sufficient experience in the domain.

If our probabilistic model is supplied with efficient procedures to compute security characteristics formulated in probabilistic terms, we can use simulation with different plausible distributions. And these simulations will give us a good material for practical conclusions. Similar for time. Moreover, if our model is supplied with efficient algorithms that can treat parametric descriptions of time and probability values, this may give an 'ideal' result: a domain of probability distributions and time parameters for which our model satisfies the desired properties.

The paper consists of two sections. Section 2 discusses a definition of quantity of information leak based on the classical notion of information. This way of exploring information flow is well known. It goes back at least to the early eighties (e.g., [3]). Later this approach was retaken again in papers [4,5] just to mention some. We consider information leaks *with respect to a given property* and concentrate on *algorithmic aspects* of its estimation. A comparison and a general framework of notions related to information flow security, not only probabilistic, can be found in insightful [6] that also gives a good list of bibliography related to the subject. The straightforward way presented here is too abstract and more simplistic as compared to [6]. However, it gives hope to develop efficient algorithms for the quantitative evaluation of security via simulation.

The approach we describe is based on abstract transition systems that has known limits as a tool of modeling. The problem of computing the information leak is discussed only for finite state transition systems. Even for this case many questions remain open. Then we briefly discuss a complexity based approach. This approach may model other aspects of information leak, theoretically closely related to the classical notion of information. On the one hand, it seems harder to develop productive models within this complexity based approach. On the other hand, such models may prove to be more adequate to the reality.

Section 3 discusses a way to characterize the *resistance* of security policy executions against violations of access. As above, the model we introduce is simple, and computational questions are discussed only for finite state transition models. The model we consider has some flavor of attack graphs (e.g., see [7]), however it is more sophisticated and permits to elaborate models of various aspects of policy evaluation. The problem of resistance rises two questions, that of analysis of a given system, and that of a synthesis of a system with a given resistance. The question of synthesis of a system with a given resistance is just mentioned.

As the paper is an invitation to discussion, it only outlines conceptual difficulties of constructing feasible and useful models.

References to the literature are reduced to minimum. There may be very relevant papers that I missed.

2. Measuring Information Leak

Consider a simple abstract state model of information flow (see, e.g., [8]). A *system* $\mathcal{S}$ is defined as a subset of the set of *traces* (finite and infinite words) over a vocabulary V. We

assume that V consists of 2 disjoint sub-vocabularies L and H that will be referred to as *low level events* and *high level events* respectively. Elements of L and H will be also called L-events and H-events respectively. For technical reason we suppose that a finite trace terminates in a special character that prevents further continuations of this trace.

A property $\mathcal{P}$ is a subset of traces of $\mathcal{S}$ (for which it is true).

We are interested in hiding a property $\mathcal{P}$ over traces from low level observations.

2.1. Entropy of a Property over a Probabilistic System

Suppose there is an observer who knows the system, knows the property and observes low level events. More precisely, if $\tau \in \mathcal{S}$ then the observer sees the *'projection'* $\tau\big|_L$ of τ onto L (there is more than one possibility to define 'projection'; we discuss it later). How this observation augments the observer's information about $\mathcal{P}(\tau)$? How to measure this information? To answer to this question we need a notion of information.

The classical information theory defines the information in a probabilistic manner. We recall it in order to be self-contained. Let ξ and η be 2 random variables with a finite number of values, defined on 2 respective probabilistic spaces. Denote by k and l their respective numbers of values, by $\{p_i\}_{1\le i\le k}$ and $\{q_j\}_{1\le j\le l}$ their probability distributions, and by $\{r_{ij}\}$ their joint distribution (defined on the product of their probabilistic spaces). The quantity of information in ξ relative to η is given by the formula

$$\mathbf{I}(\xi \,\big|\, \eta) = \sum_{i=1}^{k}\sum_{j=1}^{l} r_{ij} \log \frac{r_{ij}}{p_i q_j}. \tag{1}$$

(Here all logarithms are to base 2.) A simpler version suffices for us as we consider only one distribution over traces of $\mathcal{S}$. This simpler version is $\mathbf{H}(\xi) = \mathbf{I}(\xi, \xi)$ that is the *entropy* of ξ:

$$\mathbf{H}(\xi) = -\sum_{i=1}^{k} p_i \log p_i, \tag{2}$$

(as $k = l$, $p_i = q_i$, $r_{ii} = p_i$ and $r_{ij} = 0$ for $i \ne j$).

The entropy is a measure of uncertainty: the less we know about a random variable, the higher is the entropy of this variable. Consider an example in Table 1.

Table 1. Table

Distribution	**Pr**(00)	**Pr**(01)	**Pr**(10)	**Pr**(11)	Entropy
A	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	2
B	$\frac{7}{8}$	$\frac{1}{24}$	$\frac{1}{24}$	$\frac{1}{24}$	≈ 0.77
C	1	0	0	0	0

In this example we have a random variable with 4 values 00, 01, 10, 11. Table 1 shows 3 different probabilistic distributions A, B, C. We calculate their entropy according to formula (2):

for A: $-(1/4log1/4 + 1/4log1/4 + 1/4log1/4 + 1/4log1/4) = 2$,
for B: $-(7/8log7/8 + 3(1/24log1/24)) \approx 0.75$,
for C: $-(1\log 1 + 0\log 0 + 0\log 0 + 0\log 0) = 0$ (we assume that $0\log 0 = 0$).

Distribution A in Table 1 is a uniform distribution. Imagine we observe this random variable. In a sampling its value appears with probability $\frac{1}{4}$. A priory we have no possibility to guess what will be this value. This is the maximal uncertainty, and when we see the value we get 2 bits of information. Distribution C of Table 1 is a trivial one where the value 00 appears with probability 1. In this case we know a priory that we will see this value, and thus when it appears we get no information. An intermediate case is shown by distribution B. It is harder to interpret the value of entropy in terms of bits in our context, but clearly, it measures the information and can be used to compare different distributions.

The classical information theory was motivated by the analysis of information transmission via non reliable channels. In the world of information transmission one deals usually with stationary processes, and one can have long series of observations. These data permit to really estimate probabilities on the basis of statistics, and thus, to model the reliability of channels. In the world of computing, the processes we deal with, are usually not stationary, and we do not have opportunity for sufficiently long sampling (however, one can see that the behavior of sufficiently large intensively used networks shows more and more signs of 'foreseeable behavior'). In spite of all this argumentation, we may use probabilistic models along the lines mentioned in Introduction (section1): either to take some plausible probability distributions, simulate the behavior and make a conclusion, or to consider parametric probabilities and to try to compute the domains of their values for which the system verifies the requirements.

Suppose that we have a probabilistic distribution over traces constituting $\mathcal{S}$. Moreover, we assume that the sets we will consider are measurable in our probability space. This question demands an attention, but we do not address it. For our simplest models it is easy to ensure the measurability. The distribution over $\mathcal{S}$ defines the entropy of property $\mathcal{P}$:

$$\mathbf{H}(\mathcal{P}) = -\big(p\log p + (1-p)\log(1-p)\big), \tag{3}$$

where $p = \mathbf{Pr}\{\mathcal{P}\big|\ \mathcal{S}\} = \mathbf{Pr}\{\tau \in \mathcal{S} : \mathcal{P}(\tau)\}$. (In this notation $\mathcal{P}$ is considered as a random variable over $\mathcal{S}$ with distribution **Pr**.)

One can easily see that the behavior of the function $\varphi(p) = -(p\log p + (1-p)\log(1-p))$ on $[0,1]$ is similar to the behavior of the indistinguishability function $\psi(p) = 1 - |(1-p) - p| = 1 - |1-2p|$, namely, $\varphi(0) = \psi(0) = \varphi(1) = \psi(1) = 0$, the both monotonically increase on $[0, \frac{1}{2}]$, have their maximum at $p = \frac{1}{2}$, where $\varphi(\frac{1}{2}) = \psi(\frac{1}{2}) = 1$, and monotonically decrease on $[\frac{1}{2}, 1]$.

Below we will deal with probabilities of properties, thus with random variables with two values. The just made remark says that if we can calculate the probability of the property then we can calculate its entropy, and thus can quantitatively evaluate the information.

2.2. A Measure of Quantity of Information leak

Now we are in position to introduce a measure of information leak. We have a system $\mathcal{S}$ with a probability distribution over the set of its traces and a property $\mathcal{P}$ over traces.

One can consider two types of projections of traces on low level events: without time and with time. "With time" may mean, for example, that we do not see the high level events but we can measure the time duration of these events.

Consider the first case. In this case the projection of a high level trace onto L is obtained by deleting characters of H from the trace.

To define our measure of information leak we introduce notations:

(Timeless projection) Denote by $w\big|_L$, where $w \in V$, the just mentioned projection of w onto L, i.e., the word obtained from w by deleting letters that are not in L.

(Property traces) $\mathcal{S}_\mathcal{P} =_{df} \{\tau \in \mathcal{S} : \mathcal{P}(\tau)\}$.

(Probability of the property) Recall the notation introduced in (3)
$p = \mathbf{Pr}\{\mathcal{P}\} = \mathbf{Pr}\{\mathcal{S}_\mathcal{P}\}$.

(Pre-image of an observation) $\mathcal{B}(\lambda) =_{df} \{\tau \in \mathcal{S} : \tau\big|_L = \lambda\}$.

(Local observational probability) The conditional probability to have the property $\mathcal{P}$ when observing a low level event $\lambda \in \mathcal{S}\big|_L =_{df} \{\lambda \in L^* : \exists\, \tau \in \mathcal{S}\,(\tau\big|_L = \lambda)\}$:

$$p_\lambda =_{df} \mathbf{Pr}\{\,\mathcal{P}\,\big|\,\lambda\} = \frac{\mathbf{Pr}\{\tau \in \mathcal{B}(\lambda) :\, \mathcal{P}(\tau)\}}{\mathbf{Pr}\{\mathcal{B}(\lambda)\}}. \tag{4}$$

(To avoid technical problems, we suppose that $\mathbf{Pr}\{\mathcal{B}(\lambda)\}$ is always positive.)

(Minimum and maximum of local observational probabilities)

$$p_{min} =_{df} \inf\{p_\lambda : \lambda \in \mathcal{S}\big|_L\}, \quad p_{max} =_{df} \sup\{p_\lambda : \lambda \in \mathcal{S}\big|_L\}. \tag{5}$$

(Information leak in terms of probability: indistinguishability)

$$\max\{|p - p_{min}|, |p - p_{max}|\}. \tag{6}$$

(Information leak in terms of entropy: acquired information)

$$\sup_{\lambda \in \mathcal{S}\big|_L} \{|\mathbf{H}(\mathcal{P}) - \mathbf{I}(\mathcal{P}\,\big|\,\lambda)|\}. \tag{7}$$

This formula (7) can be rewritten in terms of **Pr** and Boolean operations over $\mathcal{S}_\mathcal{P}$ and $\mathcal{B}(\lambda)$.

□

2.3. Computing Information Leak for Finite State Transition Models

In the general setting not only computability but even correctness of the definitions above is not evident as we can meet non measurable sets. Consider models where everything is represented by finite transition systems, moreover, where the traces are finite. We assume that in our setting the measurability is ensured for all the events that appears. Even this case is not simple to analyze, and we only sketch some possibilities to go on, and difficulties that we meet. The case of infinite traces in the same framework of finite transition systems can be studies along similar lines.

As a system $\mathcal{S}$ we take a Markov chain with transitions labeled by letters of V. This chain has an absorbtion state that indicates the end of a trace. A property $\mathcal{P}$ is represented by a deterministic finite automaton. The set of traces of $\mathcal{S}$ for which $\mathcal{P}$ is true can be represented in terms of a product of these two finite transition systems. If we make a product, each transition will be labeled by a pair of letters and by a probability. We leave only pairs with coinciding letters. To normalize probabilities for a given state we introduce a 'trap' state — an absorbing state where the process goes to with the probability that complements to 1 the sum of probabilities of the kept transitions. Denote the obtained system $\mathcal{SP}$.

Thus, we can compute p defined in (3) as the probability to reach appropriately defined final states of $\mathcal{SP}$.

Along similar lines one can compute p_λ defined by (4).

To compute the leak (6) we need p_{min} and p_{max} (5). I do not know how to calculate these values exactly in the general case. Consider particular cases.

All $\mathcal{B}(\lambda)$ has the same probability. In this case we can take into account only $\widetilde{p}(\lambda) = \mathbf{Pr}\{\tau \in \mathcal{B}(\lambda) : \mathcal{P}(\tau)\}$. Replace in $\mathcal{SP}$ all high level labels by a new letter, say $\top$. Denote the obtained system $\mathcal{SP}^\top$. We look at $\mathcal{SP}^\top$ as at a Markov Decision Process (MDP) with actions (decisions) defined by labels, i.e., by letters of $(L \cup \{\top\})$. Minimal and maximal policies are Markov policies. Such a policy gives for each state the same action to execute independently of the previous ones. An action is a letter of $(L \cup \{\top\})$. A sequences of actions gives a word whose projection onto L represent a low level observation. However a minimal or maximal policy does not give a concrete word in general as it describes a limit behavior. It is well known that one can efficiently (with the complexity of general linear programming that is polytime) compute $\widetilde{p}_{min}$ and $\widetilde{p}_{max}$, as well as a minimal and maximal policy (the total number of such policies can be exponential in the worst theoretical case).

'Small' number of different probabilities of $\mathcal{B}(\lambda)$. Suppose that the set $\mathcal{S}\big|_L$ of low level observations of traces of $\mathcal{S}$ is divided into ν classes $\mathcal{S}L_i$ such that all λ from a same class have the same probability of $\mathcal{B}(\lambda)$ and each $\bigcup_{\lambda \in \mathcal{S}L_i} \mathcal{B}(\lambda)$ can be represented by a finite transition system. We can apply the construction of the previous case ν times to find $p_{min}(\lambda)$ and $p_{min}(\lambda)$.

Computing $p_{min}(\lambda)$ and $p_{max}(\lambda)$ for Markov policies. I do not know whether Markov policies are sufficient to compute $p_{min}(\lambda)$ and $p_{max}(\lambda)$ in the general case. It is evident that for Markov policies the values under discussion are computable because there are only finite number of Markov policies, more precisely, the upper bound is the number of states powered by the number of possible actions.

Computing approximate values of $p_{min}(\lambda)$ and $p_{max}(\lambda)$. If in the general case there is no maximal or minimal Markov policy to compute $p_{min}(\lambda)$ and $p_{max}(\lambda)$, then one can try to calculate them approximately. Il is feasible, for example on the basis of a technique of linear inequalities similar to the technique that is used to describe maximal strategies. The problem is to arrive at a reasonable complexity (this study is a work in progress).

Coming back to the first case, when all $\mathcal{B}(\lambda)$ have the same probability, we see the following advantage dealing only with Markov policies. The values $p_{min}(\lambda)$ and $p_{max}(\lambda)$ are defined by some linear programming problem. This means that their calculation can be reduced to solving a system of linear inequalities. We can treat them sym-

bolically. In particular we can put in this system symbolic or, in other words, parametric probabilities. A question of practical interest is to describe the domain of probabilities for which the leak is smaller than a given ε. This question can be described, even with ε as parameter, as a formula of Tarski algebra. A quantifier elimination procedure gives a quantifier free description of admissible values of parameters. This description is of practical value — see Introduction.

Adding time. One model with a timed flavor is a model where projection of a high level event keeps a trace of this high level event as a new symbol, say ι. The symbol ι is the same for all H-events. Projection of $\tau \in \mathcal{S}$ is obtained from τ by replacing each letter of H by ι. In fact, this is a rather informative model as it permits to count the number of H-events. From computational viewpoint one finds a setting similar to the timeless one discussed above. One may argue that the possibility to count high level events is a too strong possibility.

Another way, that seems more adequate, is to take into account the duration of events as an observable value. To represent the duration of events we attribute to each event v an interval of time $\sigma_v = [\sigma_v^l, \sigma_v^r]$. If time is discrete we can eliminate it by introducing intermediate states because the number of events is finite. However, if the intervals are long, such an elimination augments the complexity too much. One can imagine other ways to represent time abstractions in timeless models. For example, all durations are partitioned into a finite number of classes, and we introduce for each class a special letter to refer to it. But it is not a metric time that permits to speak about time distances and to manipulate them.

If we use continuous time, this brings us to a domain whose computational complexity has not been sufficiently studied as far as I know. However, continuous time is adequate and it looks feasible.

Suppose that time is interpreted as $\mathbb{R}_{\geq 0}$. Ascribe to each event an interval of continuous time. Formally we speak about real valued time. But the concrete values that we can deal with, depend on admitted constants and arithmetical operations. If the constants are rational numbers, and as operations we have addition and unary multiplications by rational numbers, we are in the theory of real addition and deal only with rational numbers. If we admit binary multiplication, we are in the theory of real addition and multiplication (Tarski algebra), and deal with algebraic numbers. Some interesting probabilistic distributions involve exponential function; in this case we may confront a theory whose decidability is a hard open question.

Suppose that, as before, we deal with finite transition systems. In a probabilistic model each state s has a stationary probability density $p_v(s, s', x)$ to go to state s' at relative time x, where v is the event labeling the transition (s, s'). "Relative" means the following. If the system is in s at an instant t_0 then with probability density $p_v(s, s', x)$ the system passes to s' at instant $(t_0 + x)$ for $x \in \sigma_v$. Thus, the probability to arrive from s to s' via this transition by time instant $t_1 \in (t_0 + \sigma_v)$ is $\int_0^T p_v(s, s', t)dt$, where $T = t_1 - \sigma_v^l$. Formally speaking the interval σ_v is irrelevant as it means that $p_v(s, s', x) = 0$ for $x \notin \sigma_v$, however it is important for calculations. For example, if we know that all σ_v are separated from zero and are finite (i.e., $0 < \sigma_v^l \leq \sigma_v^r < \infty$), we have at least some decidability that is not evident otherwise. In particular, with such a separation we have no Zeno paths. In the finite automata representation of systems that we used above, each transition (s, s') has only one label v, so the subscript v of p_v is redundant in this case.

A system just described above defines a stochastic process (if the measurability is ensured, that we suppose). In the simplest case, when the probability of transition depends only on the state (and not on time) we get a continuous time Markov chain. This notion is too restrictive in our context as in this case we arrive at some kind of Poisson distribution. A larger class that is of interest in our context is the class of semi-Markov processes. Recall that a stochastic process $X(t)$ with finite or enumerable set of states is semi-Markov if it has stepwise trajectories such that each sequence of jumps defines a Markov chain. More precisely, for some probability distribution $F_{ij}(x)$ if the jumps are at instants $0 < t_1 < t_2 < \dots$ then

$\mathbf{Pr}\{t_n - t_{n-1} \le x \wedge X(t_n) = j \big| X(t_{n-1}) = i\} = p_{ij} \cdot F_{ij}(x)$,

where $p_{ij} = \mathbf{Pr}\{X(t_n) = j \big| X(t_{n-1}) = i\}$.

These kind of processes, their generalizations and the respective decision processes were studied from computational viewpoint (e.g., [9,10,11]), however many questions related to computational complexity in our context remain open.

2.4. Complexity Based Approach to Measuring Information Leak.

Another way to model the quantity of information can be based on a notion of epsilon-entropy of compact spaces introduced by Kolmogorov ([12]). This notion is related to partition based view on entropy, and thus to the classical one (e.g., see [13]). We outline an inference complexity approach based on the notion of entropy from [14,15] that was inspired by epsilon-entropy and has also a flavor of Kolmogorov complexity [16].

The inference complexity is defined with respect to an inference system that is assumed to be of very general nature: an inference rule transforms a proof into another proof. Let $\mathcal{F}$ be such an inference system. A proof is a list of formulas that are marked as assumptions and conclusions. Each conclusion contains information of its origin. A proof in $\mathcal{F}$ will be also called *$\mathcal{F}$-proof*.

The length of a word W is denoted by $|W|$. For a finite set E of words, $|E|$ denotes the sum of the lengths of the elements of E. If S is a set of algorithms then we mean that its elements are represented as words, and we treat $|S|$ according to this representation.

By $Assump(L)$ and $Concl(L)$ we denote respectively the set of formulas of a list L marked as assumptions and the set of formulas of L marked as conclusions. These sets may intersect, for example, an axiom may be marked as assumption and conclusion; this gives a zero-length proof of the axiom. And as a measure of size of a proof we will use $lc(L) = |Concl(L) \backslash Assump(L)|$.

Let Φ and Γ be finite sets of formulas. Denote by $Proofs_S(\Phi \mid \Gamma)$ the set of all $\mathcal{F}$-proofs D such that $\Phi \subseteq Concl(D)$ and $Assump(D) \subseteq \Gamma$.

A set of formulas or a formula Φ is *provable in $\mathcal{F}$* or *$\mathcal{F}$-provable* from formulas (assumptions) Γ if $Proofs_{\mathcal{F}}(\Phi \mid \Gamma) \neq \emptyset$, and Φ is *$\mathcal{F}$-provable* if it is provable with empty Γ.

The principal notion for the setting under consideration is *proof complexity* of a set of formulas Φ under assumptions Γ:

$d_{\mathcal{F}}(\Phi \mid \Gamma) = \min\{lc(D) \colon D \in Proofs_{\mathcal{F}}(\Phi \mid \Gamma)\}, \quad d_{\mathcal{F}}(\Phi) = d_{\mathcal{F}}(\Phi \mid \emptyset)$.

Here we assume that $\min \emptyset = \infty$.

The function $d_{\mathcal{F}}$ can be treated as a one-directional metrics; a simple symmetrization like $d_{\mathcal{F}}(\Phi \mid \Psi) + d_{\mathcal{F}}(\Psi \mid \Phi)$ gives a genuine *metrics* on the set of finite sets of formulas. One can see that $d_{\mathcal{F}}$ is a generalization of known characteristics of complexity. An

appropriate choice of $\mathcal{F}$ can give, for example, time complexity as well as Kolmogorov complexity.

One can prove the following properties of $d_{\mathcal{F}}$:
$d_{\mathcal{F}}(\Phi \,|\Gamma) = 0, \;\; d_{\mathcal{F}}(\Phi \,|\Phi) = 0,$
$\Phi \subseteq \Psi \Rightarrow d_{\mathcal{F}}(\Phi \,|\Gamma) \leq d_{\mathcal{F}}(\Psi \,|\Gamma), \;\; \Gamma \subseteq \Delta \Rightarrow d_{\mathcal{F}}(\Phi \,|\Gamma) \geq d_{\mathcal{F}}(\Phi \,|\Delta),$
$\max\{d_{\mathcal{F}}(\Phi \,|\Gamma),\, d_{\mathcal{F}}(\Psi \,|\Gamma)\} \leq d_{\mathcal{F}}(\Phi \cup \Psi \,|\Gamma) \leq d_{\mathcal{F}}(\Phi \,|\Gamma) + d_{\mathcal{F}}(\Psi \,|\Gamma).$

Entropy of a Set of Formulas. We assume that a reasonable notion of *inconsistency* of an inference system is introduced. Two inference systems are *consistent* if their union is consistent. Since any formula can be treated as an inference rule, i.e. as an axiom, these notions can be extended to sets of formulas (seen as inference systems).

Let Φ be a finite set of formulas, S and V be inference systems, and ξ be a natural number. The notion of entropy introduced below has a flavor similar to the notion of epsilon-entropy of compact metric spaces:

$entr_{\mathcal{F}}(\Phi \,|V, \xi) = \min\{|U|\colon U \text{ is consistent with } \mathcal{F} \text{ and } \forall\, F \in \Phi\, (d_{V \cup U}\, (F) \leq \xi)\},$
$entr_{\mathcal{F}}(\Phi, \xi) = entr_{\mathcal{F}}(\Phi \,|\emptyset,\, \xi).$
(To be coherent with classical notions one should take $\log$ of $entr_{\mathcal{F}}$; however, it is not clear a priory what normalization to choose as the situation is different from those where the classical notions of entropy are used.)

One can prove that the entropy is not decreasing in Φ, not increasing in V and semi-additive in Φ.

A system $\mathcal{F}$ is said to be *optimal* for Φ and a given ξ if $entr_{\mathcal{F}}(\Phi \mid \mathcal{F}, \xi) = 0$ and $entr_{\mathcal{F}}(\Phi, \xi) = |\mathcal{F}|$.

To compare this notion of entropy with the traditional one we normalize it:
$\mathbb{H}_{\mathcal{F}} = \dfrac{entr_{\mathcal{F}}(\Phi, \xi)}{card(\Phi)}$, where $card(\Phi)$ is the number of elements of Φ.

Take as en example a random predicate P over $\{0,1\}^n$. As Φ we take the set $\{P(w) = \beta\}_{w \in \{0,1\}^n}$ whose cardinality is 2^n. It can be represented by a word $\widehat{P}$ of length 2^n over $\{0,1\}$. This representation can be considered as a formal system: given a word $w \in \{0,1\}^n$ we apply a rule that calculates its place in $\widehat{P}$, and this gives a proof for $P(w) = \beta$, where β is the value $P(w)$. And this formal system is optimal (or almost optimal if we permit very powerful algorithmic rules). The length of a proof in this system can be evaluated as $O(1)$ or even 1. For the appropriate ξ (the just mentioned length of proof) we have $\mathbb{H}_{\mathcal{F}}(\Phi, \xi) = 1$ that corresponds to the classical entropy of a random predicate.

This example shows that the dependence of $\mathbb{H}$ on $\mathcal{F}$ may be very weak. It is not very instructive as in our context we never deal with random predicates. In fact, the example uses 'almost' Kolmogorov complexity. If we admit as inference rules any total algorithms, and do not care about computational complexity of such rules, then we arrive at Kolmogorov complexity. But this complexity is hard to evaluate for the predicates we are interested in.

A way that seems productive is to measure the introduced complexity in a simple inference system that formalizes our intuition of "to be similar to". An example is editing distance with respect to one-character transformations. Let ξ be fixed. Suppose that we have chosen among all traces of our system a finite subset $\mathcal{E}$. Intuitively, this subset approximately represents the patterns of behavior we are interested in. We say this subset $\mathcal{E}$ is ξ-network for property $\mathcal{P}$, if for each $\tau \in \mathcal{S}$ there exists a pattern $\tau_0 \in \mathcal{E}$ at a distance

not greater than ξ from τ such that $\mathcal{P}(\tau) \approx \mathcal{P}(\tau_0)$. Here $\approx$ may mean $=$ or equality with some probability. If such a network is smaller for a $\mathcal{B}(\lambda)$ than for the entire $\mathcal{S}$ then we have an information leak that can be measured on the basis of comparison of the sizes of the networks (better of their $\log$'s).

The just described framework is 'less' abstract than the previous one. In the first framework the only way of extracting information is 'guessing'. In the second model one may hope to 'deduce' something. Though theoretically this deduction is not an advantage, practically, it could be better as it is easier extendable with additional knowledge. To go on in this direction the notion of trace should be concretized according to a particular application: information flow in programs (e.g, [17,18,19]), intrusion detection etc.; a survey of language related motivation can be found, e.g., in [20]).

3. Evaluating Resistance of Security Policy Executions

We are interested in the behavior of an access control security policy in a system. This is not the question about description and enforcement of this or that type of security policies. In some way we will consider a product of a policy and a system, and will call this product a system, and will try to avoid the word "policy". Notice that treating access largely, one can model in a similar manner the behavior of not only access control policies, but, for example, information flow control policies.

3.1. A Model of Policy Execution

Models we consider are based on abstract transition systems, though slightly less abstract than in the previous section.

The class of systems we introduce is defined over *universes* of:

• Components $\mathfrak{C}$; this universe consists of finite sets with relation $\in$. A component may contain other components, the latter again may contain components and so on. However, each component is finite and well founded, i.e, the tree of inclusions $\in$ is finite.

• States $\mathfrak{S}$ of components. A state is not atomic, and contains accesses as described below. It is a part of global state (see below).

• Inputs $\mathfrak{X}$; inputs are introduced just to make the systems open, and as a way to resolve non determinism.

• Actions $\mathfrak{A}$; actions model implementations of accesses, their types are concretized below.

As variables for the elements of the introduced universes we will use the following letters with indices: c for $\mathfrak{C}$, s for $\mathfrak{S}$, x for $\mathfrak{X}$ and α for $\mathfrak{A}$.

A component models, for example, a computer or an account or a set of services or a set of files etc. A component may have sub-components of similar nature. For example, an administrator manages users' accounts, programs and files. The files may have different rules of access for different users. A user may manage accesses of other users. And so on. We do not assume that components are disjoint, two components may have common sub-components.

An important element of the model is *action*. An action implements an access that may change other accesses and states.

An action has one of three types: either

$(\mathfrak{C} \times \mathfrak{S} \times \mathfrak{X}) \times (\mathfrak{C} \times \mathfrak{S}) \to \mathfrak{S} \times \mathfrak{S}$
that signifies that α acts from a component c_1 in a state s_1 with an input x onto a component c_2 in a state s_2 and changes the states of these components:
$\alpha(c_1, s_1, x, c_2, s_2) = (s'_1, s'_2)$, or

$(\mathfrak{C} \times \mathfrak{S} \times \mathfrak{X}) \times (\mathfrak{C} \times \mathfrak{S}) \to \mathfrak{S}$
that signifies that α acts from a component c_1 in a state s_1 with an input x onto a component c_2 in a state s_2, changes the state of c_1 and deletes c_2, or

$\mathfrak{C} \times \mathfrak{S} \times \mathfrak{X} \to \mathfrak{S} \times (\mathfrak{C} \times \mathfrak{S})$
that signifies that α acts from a component c_1 in a state s_1 with an input x, changes the state of c_1 and creates a component c_2 in a state s_2.

An action of the first two types may be reflexive, i.e., may act from a component onto itself.

At this high level of abstraction we introduce three actions: state change, append a component and delete a component. However, at lower levels of abstraction we may wish to have a more diverse variety of actions, taking into account inputs, reading files, writing to files, executing programs etc. Even at the present level of abstraction we structure the states.

The state s of a component c contains a set of accesses that is denoted by $Access(c, s)$. An element of $Access(c, s)$ is a tuple (c', s', α') that means that component c in state s is permitted to apply an action α' to a component c' whose state is s', in particular c' may be a new component. We presume here that the permission is valid for any input. We may introduce input as one more parameter of $Access$. However, we will refer to inputs in another way, as indicated below.

There can be various modes of execution of a system. We consider a simple one. Basically, a *transition* from one global state (see just below) to another one is determined by an action executed locally. This notion of transition is similar to the notion of transition in asynchronous distributed systems.

A *global state* $\mathfrak{g}$ of $\mathcal{S}$ is represented by a set of components $\mathcal{C} \subseteq \mathfrak{C}$ and a mapping $state : \mathfrak{C} \to \mathfrak{S}$ that attributes a state to each component. We assume that there is a function (the same for all systems) act that indicates for given c, s and x what element of $Access(c, s)$ to execute. Moreover, we limit ourselves to inputs attributed to components. This permits to avoid non determinism (for simplicity). So we suppose that an input arrives as a sequence of pairs (c_i, x_i) with $c_i \in \mathfrak{C}$ and $x_i \in \mathfrak{X}$. An input is processed sequentially, and a pair (c_i, x_i) says that the system makes transition by applying an action $act(c_i, state(c_i), x_i)$ from c_i. One may need to consider more complicated inputs, like involving also states, not ordered or permitting simultaneous actions etc.

A *run* of a system from an initial global state $\mathfrak{g}_0$ with the set of components C_0 is determined by an input. The first element (c_0, x_0) of the input says what action to apply to C_0. This application carries the system into a state $\mathfrak{g}_1$. Then the next element of the input transforms $\mathfrak{g}_1$ into $\mathfrak{g}_2$ and so on.

We presume *dependencies* between accesses. That means that certain accesses imply others. A traditional dependency is transitivity: if (c_0, s_0) has an access to (c_1, s_1) via an action α, et (c_1, s_1) has an access to (c_2, s_2) via the same action then (c_0, s_0) has access to (c_2, s_2) via α. However, this derivative access can be not given explicitly in the system. We suppose that there are dependencies between accesses, for example transitivity, inheritance of access and maybe other dependencies — we suppose that they are easy

to verify. To formulate some questions of synthesis of policies one needs to distinguish between dependencies inherent to the systems and dependencies defined by the policy. For the analysis of the resistance the nature of dependencies is not so relevant.

To compare the 'power of access' of systems we need some notions. We speak about systems over the same set of universes.

A global state $\mathfrak{g}_1$ with state mapping $state_1$ is said to *include* $\mathfrak{g}_0$ with state mapping $state_0$ if each component of $\mathfrak{g}_0$ is a component of $\mathfrak{g}_1$, and for each component c_0 of $\mathfrak{g}_0$ we have $state_0(c) = state_1(c)$, $Access(c_0, state_0(c_0)) \subseteq Access(c_1, state_1(c_1))$.

A system $\mathcal{S}_1$ in state $\mathfrak{g}_1$ is an *extension* of a system $\mathcal{S}_0$ in state $\mathfrak{g}_0$ if $\mathfrak{g}_1$ includes $\mathfrak{g}_0$.

The behavior of a system $\mathcal{S}_1$ from a state $\mathfrak{g}_1$ is an *extension of the behavior* of a system $\mathcal{S}_0$ from a state $\mathfrak{g}_0$, if for any run ρ_0 of $\mathcal{S}_0$ from $\mathfrak{g}_0$ there is a run ρ_1 of $\mathcal{S}_1$ from $\mathfrak{g}_1$ that preserves the extension of states, i.e., the kth state $\rho_0(k)$ of ρ_0 is included in the kth state $\rho_1(k)$ of ρ_1.

A system $\mathcal{S}_1$ is an *extension* of $\mathcal{S}_0$ if for any state $\mathfrak{g}_0$ of $\mathcal{S}_0$ there is a state $\mathfrak{g}_1$ of $\mathcal{S}_1$ such that the behavior of $\mathcal{S}_1$ from this state is an extension of the behavior of $\mathcal{S}_0$ from $\mathfrak{g}_0$.

Two systems are *equivalent* if each one is an extension of the other.

Suppose that we have two systems $\mathcal{S}_0$ and $\mathcal{S}_1$, and $\mathcal{S}_1$ is an extension of $\mathcal{S}_0$. Suppose that their initial states are respectively $\mathfrak{g}_0$ and $\mathfrak{g}_1$, and $\mathfrak{g}_0$ is included into $\mathfrak{g}_1$. Compare the behaviors of these systems for a given input X. From the fact that act is universal, i.e., does not depend on the system, we conclude that while the run of $\mathcal{S}_0$ is defined, the run of $\mathcal{S}_1$ will be also defined and will consist of the same actions.

In addition to a description of a system we suppose that a *requirement* is also given. The requirement specifies what accesses should be *permitted* and what should be *forbidden* for all executions of the system.

With respect to given requirements, that may also specify what components must be present in the system, we look for a minimal system that verifies these requirements. This minimality means that the amount of accesses should be minimal to meet the requirements. This can be formulated in terms of the notion of extension introduced above: a system $\mathcal{S}_0$ is *minimal* for a given requirement if there is no other system $\mathcal{S}$ that satisfies the requirement and for which $\mathcal{S}_0$ is an extension.

Thus, we arrive at a question of *minimization* of a given system that preserves the requirements. To be minimal is important for the security; intuitively, the smaller is the system, the better is its resistance against possible security breaches.

3.2. Resistance to security breaches

It is hard to estimate risks and vulnerabilities. It is simpler to estimate losses from concrete security breaches (as we mentioned in Introduction, all these estimations are demanded by the methodologies of security management).

Suppose that a function Δ of *losses* is given. This function (with rational non negative) number values is defined over permitted and forbidden accesses. A loss for a forbidden access signifies a loss implied by breaching this prohibition. A loss for a permitted access means a loss implied by blocking this access.

One can represent a system as a graph, for example, in the following way. Each component is a vertex. Sub-components of a given component are connected to it by 'inclusion' arcs. Accesses are presented by 'access' arcs. Each access arc is labeled by an access. So there may be many access arcs between two components. The requirement

defines also ‘no-access’ arcs, i.e., forbidden accesses, that are labeled in a way similar to the labeling of access arcs.

The function Δ is defined over access and no-access arcs. A gain of a forbidden access may imply, via dependencies, other forbidden accesses that increase the total loss. Similar for the accesses that are blocked, blocking an access may imply blocking some other ones.

Consider a system $\mathcal{S}$ that is determined by a set $\mathfrak{S}_0$ of initial states and a set $\mathfrak{X}_0$ of inputs. Suppose that the system is consistent, i.e., satisfies the requirement which presumes, in particular, that there never appear contradictions between permitted and prohibited accesses.

Suppose that for each state there is given a distribution of probabilities of violating permissions or prohibitions. We call such a distribution *B-distribution* (*B* stands for *Breach*). It is easy to calculate the expected loss for a given state.

There another interesting problem for a given state. Suppose we partially ordered the B-distributions. So we can speak about minimal B-distribution that implies a loss greater than some given bound Λ. Such distributions characterize Λ-*resistance* of a given state. More precisely, a state is Λ-resistant for a given B-distribution if the expected loss is not greater than Λ.

What we are more interested in is the Λ-*resistance* of the system. This means that given a B-distribution for each state, whether all the states that can appear in executions of $\mathcal{S}$ are Λ-*resistant*. Even if B-distributions are described in a simple finite manner, this question is undecidable for general systems, even for rather simple systems that can grow up (one can follow the lines of the undecidability proof from [21]). However, the proofs of undecidability we know, are irrelevant to the computational reality, and thus, it is more productive to look for what is decidable.

The first question about resistance, that is feasible, is about systems that do not grow, and that are described by finite transition systems. Suppose for simplicity that there is only one finite initial state S_0, and that the set of inputs $\mathfrak{X}_0$ of $\mathcal{S}$ is described by a finite automaton. The set of actions is supposed to be finite and fixed. In this case the set of states that can appear is finite. Thus, if a given B-distribution is stationary, i.e., does not depend on time, then Λ-resistance is computable. Similar for the inverse problem: what minimal B-distribution implies non Λ-resistance. But what is the complexity of solving these problems?

In practice the probability of breaching the security increases with time. In our model this means that the B-distribution evolves in time. For simply defined evolutions, e.g., for evolutions where the probabilities grow up with a constant speed, how to calculate the time interval where $\mathcal{S}$ remains Λ-resistant? Again this is a question of complexity of solving this problem if the probability evolution can be described by polynomials. If exponents appear it depends on particular features as we may not know the decidability of theories with exponential functions that may appear.

One can make the same framework more probabilistic. Suppose that a probability distribution over inputs is given. For the model under discussion that may mean that the inputs are defined by a Markov chain for a timeless case or by a stochastic process like semi-Markov process for continuous time.

In any case an interesting question is how to construct a ‘most dangerous’ behavior or to describe all such behaviors.

Growing systems. For growing up systems almost all interesting questions become undecidable in the general case. Thus, to be able to analyze systems with growing number of components we have to impose realistic constraints on the type of this growth to avoid the undecidability.

Growing practical systems possess this or that self-similarity. If a local vulnerable configuration, i.e., a configuration with a positive probability of security breach, can have a growing number of occurrences then clearly the loss will surpass any given bound. The time of arrival of this situation may be harder, and more interesting, to estimate. It depends on the propagation of losses in the states that may appear. If for one given breach the propagation is bounded, then the resistance depends on how this propagation may interact with other ones. At this point one can see that we touch the question of resistance of various architectures of existing systems.

Other criteria of resistance may be of interest. For example, the expected loss normalized by the size of the system. This question is also open.

Conclusion.

We have outlined simple models of quantitative estimation of some security characteristics of a given system. This framework is centered on algorithmic questions of decidability and complexity of computing these security characteristics. The question how to model evolving systems in a reasonable setting, that can be supported by efficient algorithms of analysis of security, remains open.

The question of synthesis a system with given properties is of evident interest also. For the model of security policy execution it means the following. Suppose we have a set of actions, a set of dependencies and a function of losses. We wish to construct a system with a given set of components and a given set of inputs that is Λ-resistant for a given B-distribution of probabilities of violation of accesses (better, resistant for a given class of B-distributions).

References

[1] J. Hunter. *An information security handbook*. Springer, 2001.

[2] S. Bosworth and M. E. Kabay, editors. *Computer Security Handbook*. John Wiley, 4th edition edition, 2002.

[3] D. Denning. *Cryptography and Data Security*. Addison-Wesley, 1982.

[4] J. McLean. Security models and information flow. In *Proc. IEEE Symposium on Security and Privacy*, pages 180–189, 1990.

[5] J. W. III. Gray. Toward a mathematical foundation for information flow security. In *Proc. IEEE Symposium on Research in Security and Privacy*, pages 21–34, 1991.

[6] J. Halpern and K. O’Neil. Secrecy in multiagent systems. In *Proc. of the 15th IEEE Computer Security Foundations Workshop (CSFW’02)*, pages 32–48, 2002.

[7] V. Gorodetski and I. Kotenko. Attacks against computer network: Formal grammar-based framework and simulation tool. *Lecture Notes in Computer Science*, 2516:219–238, 2002.

[8] D. McCullough. Specifications for multi-level security and a hook-up property. In *Proc. IEEE Symposium on Security and Privacy*, pages 161–166, 1987.

[9] K. Murphy. A survey of POMDP solution techniques. Technical Report, U.C. Berkeley, 2000.

[10] K. Murphy. Learning Markov processes, 2002. Submitted to Encyclopedia of Cognitive Scince, ECS-40A.

[11] H Akan L. S. Younes and Reid G. Simmons. Solving generalized semi-Markov decision processes, March 26 2004. http://citeseer.ist.psu.edu/701226.html; http://www-2.cs.cmu.edu/ lorens/papers/aaai2004.pdf.

[12] A. N. Kolmogorov and V. M. Tikhomirov. ε-entropy and ε-capacity of sets in functional spaces. *Amer. Math. Soc. Transl.*, 17:277–364, 1961.

[13] N. Martin and J. England. *Mathematical Theory of Entropy*. Addison-Wesley, 1981.

[14] A. Slissenko. On measures of information quality of knowledge processing systems. *Information Sciences: An International Journal*, 57–58:389–402, 1991.

[15] A. Slissenko. Minimizing entropy of knowledge representaion. In *Proc. of the 2nd International Conf. on Computer Science and Information Technologies, August 17–22, 1999, Yerevan, Armenia*, pages 2–6. National Academy of Sciences of Armenia, 1999.

[16] M. Li and P. Vitányi. *An introduction to Kolmogorov complexity and its applications*. Springer, 2nd edition, 1997.

[17] D. Clark, S. Hunt, and P. Malacaria. Quantified interference for a while language. QAPL'2004, 2004. Preliminary version.

[18] D. Clark, S. Hunt, and P. Malacaria. Quantitative information flow, relations and polymorphic types. *Journal of Logic and Computation*, 15, 2005.

[19] A. Di Pierro, C. Hankin, and H. Wiklicky. Approximate non-interference, October 2002. http://citeseer.ist.psu.edu/542444.html; http://www.di.unipi.it/ dipierro/papers/CSFW02.ps.

[20] A. Sabelfeld and A. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):1–15, 2003.

[21] M. Harrison, W. Ruzzo, and J. Ullman. Protection in operating systems. *Communications of the ACM*, 19(8):461–471, 1976.