7 NONDETERMINISTIC SECURITY MODELLING

History abhors determinism but cannnot tolerate chance.
Bernard De Voto,
The Course of Empire (preface),
1952.

Science repulses the indefinite.
Claude Bernard,
Introduction à l'Étude de la Médecine Expe´rimental,
1865.

7.1. INTRODUCTION

David Sutherland's 1986 paper [Sutherland86] introduced the nondeducibility definition of security which, among its other attributes, appears to have been the first such formal definition that could be applied to nondeterministic systems. There are, however, no indications in the paper that the author's efforts had been directed specifically in that direction, or that the achievement was recognized as being of particular importance.

Since that time, nondeterminism has gained an increased following among security theorists. Nondeducibility sparked the development of many new models and modelling methods in which nondeterminism is held to be a fundamental concept, reflecting the growing conviction among some researchers that the ability to represent nondeterministic systems is essential to any modern formal definition of security.

This chapter deals with several questions about nondeterminism in security models. Foremost among these is the validity of this emerging belief in the indispensability of nondeterminism. Our thesis is that, in fact, nondeterminism is neither necessary nor desirable, and that existing nondeterministic models fail to adequately address the security modelling problems cited as evidence supporting the need for nondeterminism.

7.2. NONDETERMINISM VERSUS RANDOMNESS

The original referent of *determinism* is a philosophical doctrine that denies the existence of free will. When the word was adopted for various mathematical uses, it spawned the antonym *nondeterminism*, which acquired an interpretation slightly different from ''free will''.

Formal language theory uses mathematical structures called *nondeterministic automata*[1]. The *nondeterministic Turing machine* encountered in the study of computational complexity is a close relative. They, and their deterministic counterparts, are idealized machines which, when presented with some input, proceed through a sequence of ''moves'' (state transitions) and possibly eventually halt. In the nondeterministic automaton, the state transition function maps the current state and input symbol into a set of possible moves rather than a single move. The automaton, however, does not randomly select one of these possibilities (a machine that does is properly called *probabilistic* rather than nondeterministic). Instead, it effectively enters all of those next states, thereby simultaneously performing all possible computations on its input. This can be viewed in various ways: that the current state is really a set of states, or that the machine duplicates itself and each duplicate follows a different transition, or that it somehow ''knows'' which transition will lead to a successful computation and follows that one.

In contrast, researchers in formal security models have tended to use the term ''nondeterministic'' to refer to machines whose operation contains some elements of randomness. ''Probabilistic'' would be a better name, but for those models that represent the variations without specifying probabilities it might be less misleading to use one of ''random'', ''stochastic'', or (the author's favourite) ''aleatory''.

However, rather than trying to correct an established misuse, the current security modelling practice will be continued here. Readers familiar with the term ''nondeterministic'' as it is used elsewhere are encouraged to read it as ''probabilistic'' wherever it appears in the balance of this document.

7.2.1. The History of Nondeterminism

As mentioned in the introduction, the first formal definition of security to be applicable to nondeterministic systems was nondeducibility [Sutherland86]. Briefly, let Ω represent the set of all possible system executions. These may take the form of event traces. Let $visible_s(\omega)$ represent the information about the execution $\omega \in \Omega$ that observer s is permitted to see. Also, let $secret_s(\omega)$ be other information that s is not permitted to know. The system is nondeducibility-secure (with respect to s) if, for each pair of executions

[1] [Aho72], p. 113.

$\omega_1, \omega_2 \in \Omega$, there exists an $\omega_3 \in \Omega$ with the properties

$$visible_s(\omega_3) = visible_s(\omega_1)$$
$$secret_s(\omega_3) = secret_s(\omega_2)$$

One possible instantiation of the model is that the Ω are event traces in which each event is assigned a sensitivity. $visible_s(\omega)$ is the subsequence of events whose sensitivities are dominated by subject s's clearance, and $secret_s(\omega)$ is the subsequence of *input* events of higher or incomparable levels. The model assumes that the only source of high information is high inputs. The definition is basically a requirement that every possible observation made by s must be compatible with every conceivable secret. This prevents the observer from deducing the secret based on his observations.

Some time after the introduction of this definition, researchers began looking rather carefully at its handling of nondeterministic systems. That it permits them is clear: there is no requirement that ω_3 be unique, its visible and secret parts may be freely interleaved, and there are no constraints at all on other events (i.e. high outputs). However, it has several problems that relate to the current discussion [2].

First, nondeducibility defines security as a ''liveness'' property. This is a requirement that a system must be capable of behaving in a certain way (ω_3, to be precise), rather than a prohibition against specified behaviour. If the actual system displays only some of these modelled traces, for example by being more deterministic than its model, it may fail to meet the security requirement. This may happen if the model's nondeterminism represents underspecified behaviour that is fixed in the implementation. This difficulty renders nondeducibility unusable for modelling underspecified systems, as one might wish to do early in the design phase.

Secondly, and perhaps more seriously, nondeducibility makes the most optimistic assumptions possible about the statistical properties of the random (or nominally random) behaviour of the system. It calls a system secure as long as there exists at least one execution ω_3 for each pair of ω_1 and ω_2, no matter how unlikely that ω_3 may be. The following example from [Thomson88] is illustrative:

> *Two sites are connected by a communications line that has a uniform bit error rate of 10^{-9}. At one site, highly sensitive information is fed into the channel, while at the other end an uncleared observer (call him the* ***viewer*** *) watches what comes out.*
>
> *There is a small probability, but a nonzero one, that an arbitrary input may be corrupted by the channel noise to form any specified*

---

[2] Other points of contention, such as its incomposability, or its hiding only inputs, are not pertinent here.

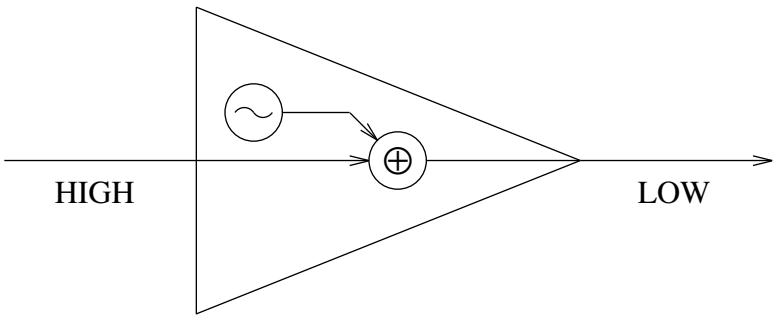

Figure 7.1: Communications Channel with Internal Noise Source.

> *output, so it is the case that every possible observation is compatible with every input. This system, then, is nondeducibility-secure, and remains so no matter how low we make the bit error probability, as long as it is greater than zero.*

This lack of robustness occurs because the model represents only the range of the noise, and because it is concerned only with absolutely certain deductions by the observer. Statistical inferences are not considered.

In the following years, researchers pursued definitions of security that were preserved by interconnection. It was believed that the composition of subsystems entails nondeterminism [McCullough88b], and so McCullough's hook-up security [McCullough87] and related restrictiveness [McCullough88b] properties admit nondeterministic behaviour. These, along with forward correctability [Johnson88] (a variation on the hook-up property), and output security [Guttman88], are composable properties that permit nondeterminism but do not address either of its problems any better than did nondeducibility.

More recently, researchers have attempted to solve the optimism problem by adding some statistical information to the model. [Wittbold90] describes what is in essence a stateless communications channel with noise. The input, output, and noise signals can each assume a range of values. By appeal to information theory, the authors make the point that the capacity of the channel is zero only if the probability distribution of received symbols is independent of the transmitted symbol. Our earlier example of this kind would be found to be insecure by this definition because, if the transmitter was sending a ''0'', the receiver would see a ''1'' with probability 10^{-9} and a ''0'' with probability $1-10^{-9}$. If the transmitter sends a ''1'', the receiver now sees

a ‘‘1’’with probability $1-10^{-9}$ and a ‘‘0’’with probability 10^{-9}.

In [Gray90], the author begins with the state machine described in [McCullough88b]. This is an asynchronous machine whose state transitions may occur on inputs, outputs, or internal events. Both outputs and next state are probabilistic; this is captured in the model by defining the relation $T \subseteq S \times E \times S$ where $t = \langle s_1, e, s_2 \rangle \in T$ means that the machine in state s_1 may engage event e and thereby move to state s_2. There may be several output events possible in any given state, and also several possible result states. McCulloughs security definition requires that high input events must not alter the set of possible low-level events that follow. Gray augments this by further requiring that the relative probabilities of the following low level events must not change.

The *flow model* of [McLean90] also seems to be loosely based on this same idea of probability invariance, but the presentation is rather sketchy and not easily compared to other examples [3]. Rather than being concerned with high and low level information, this model deals directly with high and low labelled objects. Instead of events and event sequences, it deals with states (object-value assignments) and so cannot be applied to a stateless system such as a communications channel. It also the probability of these states and of sets of states, which makes little sense unless the inputs are fixed or their statistical properties are known, neither of which appear to be the case here. Still, it does seem to contain the essential idea that the probabilities of different low-level observations must not depend on any high-level information.

It is clear that the inclusion of probability in these models improves their handling of nondeterminism. If we view a system as a communications channel, where a transmitter interferes with a receiver’s observations subject to perturbation by noise, the usability of the channel depends on the construction of the channel (which determines how effectively the noise is able to perturb observations) and on the amplitude and unpredictability of the noise. Nondeducibility models the channel construction and the (peak) noise amplitude, but assumes minimal predictability. Adding explicit probabilities tightens up the definition by at least partially characterising the predictability of the noise.

Unfortunately, these models are all still optimistic. They ignore the fact that there is more to a random process than a probability distribution. The only full characterisation of a signal is the joint probability distribution of its value, all previous values, and the current and previous values of all other signals in the system. Wittbold recognizes this danger and avoids it by

---

[3] McLean compares it to nondeducibility, but the comparison is based on an invalid interpretation of Sutherland’s high-level information functions as the contents of high-level files. Nondeducibility depends on the actual sensitivity of information, not on the (possibly higher) label of its container.

assuming that ‘‘... all symbol transmissions experience independent corruptions’’, but it is by no means clear that this assumption is valid in the networked and distributed systems that their model is intended to serve.

This point will be discussed further, but first it will be useful to formally define nondeterminism and to consider what exactly are the sources of this ‘‘random’’ behaviour that seems so badly in need of modelling and so difficult to handle adequately.

7.2.2. Definitions of Nondeterminism

The simplest definition of nondeterminism useful in security modelling is one that applies to state machines:

> *A state machine is **deterministic** if its next-state and output mappings are ordinary single-valued functions. It is **nondeterministic** if either mapping is probabilistic.*

The heart of the Bell and LaPadula access control model [Bell74] is a deterministic synchronous state machine. Goguen and Meseguer’s noninterference model [Goguen82], though it defines security rather differently, is also constructed around such a deterministic machine. These are both synchronous machines because they operate in well-defined cycles where they accept an input, produce an output, and change state.

Synchrony is adequate for many simple systems, but more complicated machines often undergo complex internal state changes between an input and any corresponding output, and often are able to generate outputs independently of inputs. A more faithful representation of the sequencing of operations is available in the form of event traces. Event traces are sequences of events representing system executions. The events may include input events, output events, internal events, and possibly composite events. The passage of time may even be crudely modelled by the inclusion of ‘‘clock tick’’ events. With this formalism, a system is characterised by the (usually infinite) set of event traces that represent valid executions of the system.

It is even possible to combine the state machine and event trace formalisms, as McCullough does [McCullough88b] to create a state machine that is excited by generalised events rather than only by inputs - i.e., its state may also change as the result of producing an output or undergoing an internal event. His model is nondeterministic both because, in general, more than one kind of event can occur in a given state, and because the state entered after the event is not uniquely defined.

To make some sense out of all these competing definitions, it is desirable to settle on a definition of nondeterminism that applies to a wide variety of system representations. We choose the event trace as the lowest common denominator of these representations. For synchronous state machines, the traces considered are the sequences of input-output pair events that can be

generated by the machine. For other machines, the events of the trace may be more general than this.

The essence of determinism, and the key to defining its opposite, is that a system's response to any given set of inputs is uniquely defined. We can distinguish three candidate definitions of determinism for event traces. Let T be the set of traces that characterise some system, and let $I(t)$ and $O(t)$ be the subsequences of input and output events in the trace $t \in T$, respectively.

Definition 1:

*A set of traces T is **strongly deterministic** iff, for any two traces t_1 and t_2, $I(t_1) = I(t_2)$ implies $t_1 = t_2$*[4].

This very strong statement is what results if ''response'' is interpreted to mean the entire execution trace, including not just all the events but also their relative ordering and any timing information that may be contained in the trace. The previously encountered synchronous deterministic state machines generate strongly deterministic traces.

If, instead, ''response'' is interpreted to mean the sequence of outputs, the following looser definition results:

Definition 2:

*A set of traces T is **input-output deterministic** iff $I(t_1) = I(t_2)$ implies $O(t_1) = O(t_2)$.*

Here a particular input sequence may correspond to several executions that differ in their unobservable internal events or in the precise interleaving of inputs and outputs. This would be useful to describe pipelined systems, or asynchronous systems with unbounded buffering. A deterministic state machine followed by a stage that delays the outputs generates traces of this type.

Definition 3:

*A prefix-closed set of traces T is **eventually deterministic** iff for all pairs of traces $t_1 = \alpha\varepsilon$ and $t_2 = \alpha\gamma$, where α is some event sequence, and ε and γ are non-input events, implies $\gamma = \varepsilon$.*

This last definition is used in the discussion of rated event systems in [McCullough88a]. If an output or internal event occurs at some point in a trace, it must be unique, but if a new input occurs first then an entirely different trace may result. This captures the notion of machines that will act in a predictable way if given enough time to produce their outputs (hence, ''eventually''), but may act differently if a new input happens (eg. someone

---

[4] This definition and the next are really too simple to be useful, because they do not permit T to be prefix-closed. Example: let t_1 be $\langle i_1, o_1 \rangle$ and t_2 be its prefix $\langle i_1 \rangle$. But, if t_2 is a trace, then T cannot be either strongly deterministic nor input-output deterministic. This may be corrected, but the definitions become rather more complicated. See p. 129 of [Thomson88] for an instance of this.

pressing the "Interrupt" button) before the output does.

The first definition is stronger than the second, which in turn is stronger than the third. Practical asynchronous systems we might wish to characterise as deterministic are probably closer to satisfying one of the latter two definitions than the first, since they do not have rigid schedules of input deliveries. For each of these types of determinism there corresponds a definition of nondeterminism that is satisfied when the implication fails.

7.2.3. Sources of Modelled Nondeterminism

Many arguments are offered for the desirability and even necessity of nondeterminism. These may be grouped according to the nature of the effect that the nondeterminism in the model is intended to represent.

Type 1 effects are those that are the manifestations of true stochastic processes, as addressed in [Gray90]. These exist in everything we build, but in most applications a substantial part of the effort of engineering design is directed toward minimizing their effects. Rarely, devices are designed so that their operation intentionally contains a significant random component. A random number generator built around a noise diode would be an example. Such devices are the exception rather than the rule.

Processors, storage devices, and communications systems are all designed to closely approximate deterministic operation in spite of the ubiquitous thermal and other natural noise sources. As a result, although the effects of this noise cannot be entirely eliminated [5], they may be rendered so rare (eg. a disk with error correction may exhibit an uncorrected error rate of only 1 in 10^{12} bits) that it may be more appropriate to threat these residual noise effects as component failures rather than as characteristics of normal system operation.

Related to true randomness (we might call these Type 1.1) are the pseudorandom systems which, though deterministic, mimic by design the statistical properties of true random processes. Some are better at this than others, but they are all subjected to some amount of statistical analysis to see just how good their approximation is. Since they are much more convenient to use than true random sources, they find application in everything from simulators to Monte Carlo solvers, to packet retransmission delay algorithms, password generators, encryption devices, and as noise sources to limit the bandwidth of covert channels.

Type 2 nondeterminism is the kind resulting from underspecification. It is reasonable that designers would wish to construct a security model of a

---

[5] At least, not in finite time. C.E. Shannon's celebrated coding theorem requires infinite signal durations to achieve totally error-free communication in the presence of noise.

system early in its design. At that stage, there may be many details that are not yet determined, but which may affect operation in some detectable way. Still other operational characteristics may not actually be known until a prototype device is built and can be instrumented during operation. In both cases, a modeller may resort to nondeterminism to account for the variability attributable to these unknown or unresolved factors.

Type 3 is nondeterminism resulting from modelling abstraction. A security modeller may choose to omit the particulars of detailed or complex deterministic processes in the system, or may employ a modelling scheme that is incapable of representing these processes in sufficient detail to expose their deterministic nature. In this way, behaviour that is actually deterministic may appear as nondeterministic in the model.

One example of this is the CPU scheduling in a timesharing computer. Typically, the algorithms used are not terribly complex, but the inputs they require may be next to impossible to provide in the model. At each rescheduling point one must know (at least) which processes are ready to run, and, having selected one, whether it will first exhaust its quantum, suspend itself, or be pre-empted by another higher priority task. This much information can only be acquired by full simulation of the system in question, which is infeasible. A practical alternative is to model only the gross characteristics of the scheduler (i.e. non-starvation) and to use nondeterminism in the model to subsume the observable variability in execution speeds.

Another example is a system with finite input buffers that may be overrun, losing or rejecting inputs if too many arrive in a burst. The system may then exhibit different behaviour in response to a given sequence of inputs depending on how rapidly they arrive compared with how quickly the system deals with them. This could be modelled deterministically if we include timing in the model and explicitly model the input buffer and the internal events that drain it. However, it is also tempting to suppress this detail and achieve a simpler model with no timing, no input buffer, and a dose of nondeterminism to account for the otherwise unexplainable loss of inputs. Here, the more precise model may be possible but awkward.

Various scenarios presented in the literature can be seen to be instances of this model-inspired nondeterminism. [McCullough88b] contains some of these, including a version of the finite buffer overflow example. It also contains a second example of two components connected so that their outputs merge [6]. Even if the components are individually deterministic, unless their models include timing information it will not be possible to predict how their outputs interleave.

---

[6] Actually, it is sufficient that their outputs be connected to the same third component, and do not have to be merged into a single stream, for the demonstrated situation to occur.

McCullough presents these examples as evidence that the composition of deterministic systems entails nondeterministic effects, and concludes that a deterministic model cannot serve as the basis of a composable security definition. However, it is the case that the actual system is deterministic, and the apparent nondeterminism is an artifact of the model and of its limitations. The modeller trades off precision for simplicity, and, in so doing, creates the impression of randomness.

7.3. PROBABILISTIC MODELS REVISITED

The practice of modelling deterministic detail as randomness is widespread in the field of computer and communications engineering. Queueing theorists commonly model processors, disks, and even users as random processes, and often as Poisson processes (even if the approximation is known to be poor) because doing so may greatly enhance the tractability of the problem.

This is an accepted approach because approximations to average-case performance, even order-of-magnitude approximations, are appropriate where these models are used. Designers of real-time systems do not use these approximations, because they require bounds for worst-case performance. The performance estimates they produce, when they err, must do so on the side of pessimism. We believe that security modelling more closely resembles design of real-time systems, in that if the models do not accurately reflect the security of the modelled system they must err on the side of pessimism. Moreover, security models must bound the worst-case operation of a system assuming intelligent and malicious attacks, which is an even stricter requirement than that faced in real-time system design.

Nondeterministic models proposed in the literature do not have this property. Even models that include probabilities may be optimistic, because they assume that the ‘‘random’’ variations in the machine’s operation have the most favourable other statistical properties, namely that every variation is statistically independent of all others.

On those rare occasions where real randomness is being modelled, it may be possible to determine accurate probabilities for a model and to justify the assumption of statistical independence. This can also be done for the pseudo-random case, if the properties of the generator can be assured.

Extending this approach to the cases where nondeterminism is only an approximation may lead to significant errors. The ‘‘noise’’ may correlate with its own previous values, as in a low-quality pseudorandom number generator. Also, noise sources modelled as independent may correlate with one another, particularly if they are manifestations of the same unmodelled mechanism. Finally, if the modelling process sweeps mechanisms and interactions under the rug of nondeterminism we cannot develop any assurance that those effects are immune to manipulation by a malicious user. In each of these

cases, a probabilistic model will give an optimistic assessment of the system's security.

7.3.1. Quasi-Determinism

In [Thomson88] and [Thomson89] it was proposed that this problem be dealt with in a very different way. Rather than adding more and more statistical detail to nondeterministic models in an attempt to reduce their optimism, Sutherland's nondeducibility model was modified to represent the effects of nondeterminism in a pessimistic way, and to do so without all of the complexity inherent in the probabilistic approach. This approach, which we will call *quasi-deterministic modelling* , consists simply of building a fully deterministic model with extra inputs, inputs that must be capable of perturbing the operation of the system to the same degree that the so-called nondeterminism would.

Extraction of randomness to form an extra input is an established modelling technique for control systems [Kumar86], where it permits the use of conventional techniques to analyse the system. In security modelling, it allows us to model the effects of nondeterminism no matter what its origin, while retaining the simplicity of a deterministic model.

If we do this to our communications channel example, we will model the system as shown in figure 7.2. Since the system is now deterministic, it may be modelled using nondeducibility without incurring either of the drawbacks described earlier (of course, other modelling methods could also be used).

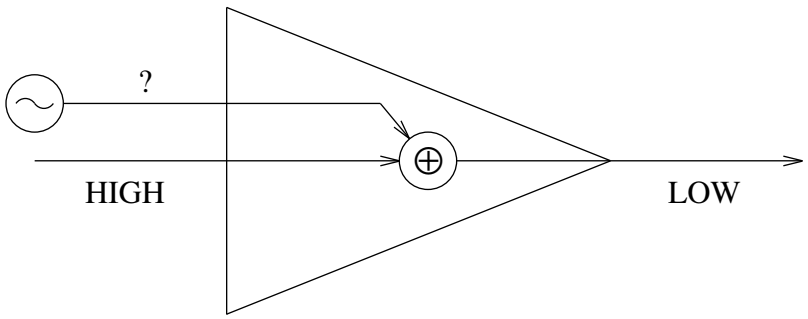

Figure 7.2: Communications Channel with Noise Source Externalized.

The precise effect of formally creating the new input depends on how it is treated during the modelling.

If we consider the noise input to be a high-level signal, it will be included in the $secret_s(\omega)$ function for the low-level observer s. Since $secret_s$ now determines both inputs to the deterministic exclusive-OR function, it is no longer independent of $visible_s$, and so the system is found to be nondeducibility insecure, which agrees with our intuition.

The reader may object in the case of the present example, because there is no compelling reason to believe that the noise signal should be considered high. If, instead, the modeller considers it a low-level input then, as