Modelling the Human and Technological Costs and Benefits of USB Memory Stick Security

Adam Beautement
UCL

Robert Coles
Merrill Lynch

Jonathan Griffin
HP Labs

Christos Ioannidis
University of Bath

Brian Monahan
HP Labs

David Pym*
HP Labs &
University of Bath

Angela Sasse
UCL

Mike Wonham
HP Labs

Abstract

Organizations deploy systems technologies in order to support their operations and achieve their business objectives. In so doing, they encounter tensions between the confidentiality, integrity, and availability of information, and must make investments in information security measures to address these concerns. We discuss how a macroeconomics-inspired model, analogous to models of interest rate policy used by central banks, can be used to understand trade-offs between investments against threats to confidentiality and availability. We investigate how such a model might be formulated by constructing a process model, based on empirically obtained data, of the use of USB memory sticks by employees of a financial services company.

1 Introduction

Organizations deploy systems technologies in order to support their operations and achieve their business objectives. In so doing, they encounter tensions between the confidentiality, integrity, and availability of information.

In formulating security policies that are intended to resolve such tensions to the organizations satisfaction, people (e.g., CEOs, CIOs, CISOs, security managers) with responsibility for information and systems security face the following two problems:

1. Poor economic understanding of how to formulate, resource, measure, and value security policies; and
2. Poor organizational understanding of the attitudes of users to both information and systems security and of their responses to imposed security policies (see, for example, the UK Foresight 'Cyber Trust and Crime Prevention' report [ST04]).

Consequently, the effectiveness and value of the policies with which users are expected to comply are very difficult to assess, as are the corresponding investment decisions [And01, AM06]. We believe that, in order to assess the effectiveness and value of security investments in a system, be they in people, process, or technology, it is necessary to have a conceptualization (i.e., a model) of the system, including its users, and its economic environment.

In this work, we present an entirely novel approach to the problem of modelling the economic effectiveness of implementing security policies within an organization. The following are the key components of our approach:

- We test the hypothesis that there is a trade-off between the components of investments in information security that address confidentiality and availability (for our present purposes, we suppress integrity);

*Corresponding author; email: david.pym@hp.com

- We capture, for now primarily conceptually rather than mathematically, the trade-off between availability and confidentiality using a model inspired by a macroeconomic model of the Central Bank Problem [RM01, RM03]. Our approach, which considers aggregate values of confidentiality and availability under variation in investment, stands in contrast to the microeconomic approaches described by Gordon and Loeb [GL02, GL06];
- Rather than provide a detailed mathematical formulation, which at this stage in our investigation we are not ready to formulate, we conduct an empirical study together with a (rigorously structured) simulation based on the empirical data and the processes executed by the system. Our simulations embody the dynamics of the conceptual model;
- Our empirical data is obtained from semi-structured interviews with staff at two organizations, a financial services company and a research organization, with a focus here on the financial services organization;
- We demonstrate the use of the model to explore the utility of trade-offs between availability and confidentiality.

The results of our study, and variations upon it, will inform our efforts to design and calibrate economic models of the kind we discuss.

The remainder of the paper is structured as follows: In § 2, we explain the form of the economic model of the response of confidentiality and availability to security investments that is of interest to us; in § 3, we expain how we have obtained our initial empirical data; in § 4, we explain the key features of our process-model of the use of USB memory sticks and, in § 5, we explain how this model is realized in our systems modelling language, Demos2k [Dem]; in § 6, we explain our experimental study, including its relationship to the economic model we sketch in § 2; finally, in § 7, we explain how we intend to pursue this work, explaining the directions empirical study, process modelling, and economic modelling. We also include two appendices, one containing a summary of the empirical data and one containing the code for our (executable) model.

2 The Central Bank Problem and Information Security

A well-known problem in macroeconomics concerns the setting of interest rates by a central bank in order to manage, say, inflation and (un)employment. The basic model derives from a line of work including Taylor [Tay93], Barro and Gordon [RG83], Nobay and Peel [NP].

In very brief summary, for readers who may be unfamiliar with the background, the basic set-up of the model is as follows [RM01, RM03]:

- Inflation and unemployment are related as

  $$u_t = u_t^n - \lambda(\pi_t - \pi_t^e) + \eta_t$$

  for $\lambda > 0$, where u_t, u_t^n, and π_t are, respectively, the rates of unemployment, natural (or target) unemployment, and inflation; π_t^e is the (public) forecast of inflation at time t, constructed at time $t-1$, determined rationally as

  $$\pi_t^e = E_{t-1}\pi_t$$

  where E_{t-1} is the expectation conditional on the set of all relevant information available at time $t-1$, denoted I_{t-1}; η_t is an aggregate supply disturbance;
- The natural (or target) rate of unemployment evolves over time, with Δu_t^n depending on the Δu_{t-k}^ns;
- The central bank affects the rate of inflation via a policy instrument, such as a base interest rate. Such an instrument is imperfect, with imperfections represented by the error term ϵ_t in the following equation, in which $i_t \in I_{t-1}$:

  $$\pi_t = i_t + \epsilon_t;$$

- The central bank's preferences for inflation and unemployment are captured by a utility, or loss, function of the following form:

$$U(\pi_t, u_t) = \frac{\phi}{2}(\pi_t - \pi_t^*)^2 + (\frac{\phi}{\gamma^2})(\exp(\gamma(u_t - u_t^*)) - \gamma(u_t - u_t^*) - 1),$$

  where π_t^* and u_t^*, respectively, are the target rates of inflation and unemployment, and ϕ is a parameter. Here the target unemployment rate is the expected (natural) rate of unemployment:

$$u_t^* = E_{t-1}(u_t^n).$$

  It is assumed that the target inflation, π_t^*, can be approximated by a constant term [RM01, RM03].

Note that the utility function taken in this set-up employs the linex function [Var74, Zel86, CPP06], of the form

$$g(x) = (\exp(\alpha x) - \alpha x - 1)/\alpha^2,$$

where α is a parameter. In comparison with the use of a quadratic utility function, the linex function admits asymmetry whilst retaining the quadratic as the special (limit) case when α tends to zero.

We argue that a form of the central bank problem (model) can be deployed to explain trade-offs in investments in information security. In our present case, we are concerned with the trade-off between availability and confidentiality, in the particular setting of the overall availability of information derived from the use of USB memory sticks set against the overall increased exposure of confidential information that is a consequence of their use. The analogy goes as follows:

- Availability and confidentiality, respectively, correspond to inflation and unemployment. The policy instrument is the level of investment in information security countermeasures;
- Availability and confidentiality are related as follows:
  - As availability increases, the potential for exposures increases, and confidentiality decreases. Confidentiality is also reduced by increased levels of threat to confidentiality

$$C = -\lambda A + \epsilon_C$$

    where λ is a parameter and ϵ_C is a non-decreasing stochastic process (so expectation non-zero) for the threat to confidentiality;
  - Availability depends both on the level of investment in information security, negatively in the case of the study discussed in this paper, and on the level of threat to availability

$$A = -\psi I + \epsilon_A$$

    where the instrument I is security investment or, perhaps, system complexity, ψ is a (possibly negative) parameter and ϵ_A is a non-decreasing stochastic process for the threat to availability. More generally, we might require also a term in changes ΔI in the instrument I, with various dependencies;
- For utility, in terms of expectations, we might take, for example,

$$E(U(C, A)) = E(((\exp[\alpha A] - \alpha A - 1)/\alpha^2 + \frac{\phi}{2}C^2),$$

  where ϕ is a parameter, as before;
- Such a formulation does have analytic solutions for I, in terms of expectation, of the form

$$I = E(\frac{1}{\psi}[\epsilon_A - \frac{\epsilon_C}{\lambda} - \frac{1}{\alpha\lambda^2\phi} + \mathrm{ProductLog}[\frac{\exp(\frac{\alpha\epsilon_C}{\lambda} + \frac{1}{\lambda^2\phi})}{\lambda^2\phi}]]),$$

  where, as in Mathematica [Cen], $\mathrm{ProductLog}[z]$ is a solution for w in $z = w\exp(w)$. A discussion of this solution and its significance is beyond our present scope, as is a discussion of a multi-period model.

As we have remarked, in the context of information systems, the instrument I might be a measure of investment in information security, or a measure of the complexity of the system. For an example of the latter, we might take a 'complexity parameter', $x \in [0, 1)$, and then take $I = 1/(1 - x)$. Then if $x = 0$, we have a maximally simple system (a single unit) and, as x approaches 1, and so I approaches infinity, we can obtain an arbitrarily complex system.

In business contexts, systems users who have access to confidential and business-critical information make widespread use of USB memory sticks. They do so for good reasons: these devices efficiently enable data transfer between all manner of business colleagues and partners. The use of these devices also exposes organizations to risks of losses of confidential data, owing to their capability to transfer all kinds of data conveniently and cheaply to anyone capable of receiving it. Thus there is a trade-off between availability and confidentiality (we suppress consideration of integrity issues in this context, where it can be argued that they are minor), and there is an incentive incompatibility between the users of the systems and owners of the policies.

In this paper, we study the use of USB memory sticks by the staff of a financial services firm, in the context of a model of the form discussed above. We do not attempt to reify such a model analytically, even at this level of detail. Rather, we demonstrate the dynamics of a simple instance using an executable model of the system of USB users using a process model.

The model, built on the basis of empirically obtained data, executes processes that track availability and breaches of confidentiality under specified levels of security investment. In assessing our experimental results within the executable model, we employ, for illustrative purposes, perhaps the simplest form of utility function that might possibly be useful:

$$U(C, A) = \alpha(A - \beta C),$$

where α and β are parameters; the details of the choices here are explained in § 6.

3 An Empirical Study

To obtain an empirical basis for our model, we conducted a study to elicit factors that contribute to corporate and individual security cost. One of the academic researchers conducted 17 in-depth interviews with security staff, employees, and managers in the two companies that are partners in this research project. The interviews remained anonymous.

The interviews were semi-structured, exploring

- the tasks and responsibilities of interviewees,
- their perception of the risks facing the company,
- their attitudes to the company's security polices and security measures, and
- the perceived impact of security measures on individuals' tasks and responsibilites, and company productivity.

Whilst the interviews covered a range of security policies and measures, all interviewees were asked about one specific security problem: USB sticks. They were asked

- if they used USB sticks (all did),
- how they used them as part of their tasks and responsibilities,
- about the relationship between the risks facing their company, and their USB stick usage,
- if whether any of their USB stick usage contravened the companies security policies, and if so,
- why they thought contravening the security policy was justified.

We suggested the company was considering making the use of encrypted USB sticks mandatory (for the financial services company, this was actually the case), and asked interviewees to

- explore the cost and benefits of such a policies for the company, and
- explain the cost and benefit for them and their tasks and responsibilities.

The interviews were transcribed, and analyzed using techniques from Grounded Theory. Grounded Theory [SC90] is a qualitative data analysis method widely used in social sciences, which allows identification of salient concepts and relationships between them. Over the past 10 years, the method has been successfully applied to model user perceptions and attitudes in Human-Computer Interaction in general. Adams and Sasse [AS99] used this approach to identify factors that affect employees' perceptions of corporate security policies, and [WS01] modelled employee decision-making on compliance with password security policies.

For the study reported in this paper, only the sections on USB stick policies and tasks and situations surrounding their usage were analyzed. We coded the interviews using axial coding (the first stage of Grounded Theory) to produce an inventory of the individual employee's cost and benefit associated with USB stick usage, and the cost and benefit for the organization. Data were coded by two researchers independently.

The range of roles performed by the interview subjects was relatively diverse, from security managers to part-time researchers, as was the range and frequency of security related comments they produced. There were also noticeable differences in USB usage between the various interview subjects. From the interviews, we were able to identify two main USB stick usage scenarios. These scenarios broadly corresponded to the type of organization for which the subject worked. We have focused on the first of these scenarios in which the USB stick is used as transport medium for data. This scenario is described in detail below. The second scenario, corresponding to the research organization, in which the USB stick is also used as a primary data storage device will not be covered here.

This scenario is more representative of the financial services organization. In this scenario, the USB stick is primarily used for temporary storage for transit between locations such as an employee visiting a client company to deliver a presentation. The data required to deliver the presentation would be copied from the company's computer system onto the USB stick and taken to the client's location. Any data which must be brought back to the home company can be copied from the client's system onto the USB stick and brought back by the employee.

The data in this case is always backed up, either on the home company's system or the client company. The data is never unique and so a loss of a security stick cannot constitute a long-term availability issue. While a short term loss of availability can be detrimental — the cost is to the individual, with possible small collateral reputation loss for the parent company if the clients need to resend data, etc. — it is unlikely to have a significant impact on the company.

A far bigger concern for the security manager in this scenario is the potential confidentiality issues resulting from company data being transported through unsecure locations while in transit to and from the client. If the USB stick were to be lost or stolen at this time, while containing unencrypted data, then the cost in terms of reputation and lost business would be to the company itself rather than the individual. While the company can punish the individual internally, it cannot recoup its losses by doing so. This scenario encourages the security manager to take a 'confidentiality first' approach when designing the USB control policy. We opted to focus on this scenario when describing our individual and organizational costs as it provided a relatively simple set of actions that encompassed the key points.

At this point we created a list of the actions required to complete the task in the scenario. This then was converted into a set of tables detailing at each stage the task, the cost to the individual, the cost to the organization, a possible failure mode at that juncture, and the cost to each of that failure. Appendix A contains the results of the empirical study, in tabulated form.

The data obtained in our empirical study which has not been explored in this paper will be considered in future work.

4 The Conceptual Model

The empirical study discussed in §3 has presented ways in which USB sticks are used in two large organizations. In particular, this study shows that certain classes of events and risks arise during the course of the life-histories of USB sticks and their owners. This information provides a rich corpus that we can use to make modelling decisions. Accordingly, we have embodied these classes of events and risks within the process model we now present. More specifically, we take as the primary input to our model the data obtained from the financial services organization.

For simplicity, we consider the organization of interest to consist in the collection of its individuals. Thus we can capture the behaviour of the organization, at this rather crude level of abstraction, by capturing the behaviour of a typical individual.

The purpose of our model is to embody the behaviour of our intended macroeconomics-inspired model of the relationship between the confidentiality and availability of information owned by an organization that uses USB memory sticks to support its operations. In this model, the instrument that is available to the organization is investment in information security. For the purposes of this study, we identify the following three types of investment:

- *Training* — individuals are trained to understand and work within the organization's information security policies;
- *IT Support* — the organization provides specialist IT personnel to help individuals resolve problems;
- *Monitoring* — the organization monitors the behaviour of the individuals with respect to its information secuirity policies.

Our focus of attention for this model concerns the use of encryption of data held on USB memory sticks.

For each type of investment, we consider the idea of a *transfer function* which associates to a given level of investment a certain parameter that is used to calculate the effect of a given level of investment. In the cases of *Training* and *IT Support*, the transfer function returns a value in the real interval $[0, 1]$; in the case of *Monitoring*, the transfer function returns a (real) time interval. There are many reasonable choices for these functions, and we take simple exemplars, chosen primarily for their shape, on the presumption that more investment will generally increase the business proficiency and efficacy of the matter of interest, and guided by the following considerations:

- Whether they are monotonic increasing/decreasing;
- What limits they tend to;
- The presence of threshold effects for investment; and
- Algebraic simplicity.

We do not claim anything else for these particular functions — we do not know a priori what these functions ought to be, and so we leave that as an open question for further investigation. We consider them in turn.

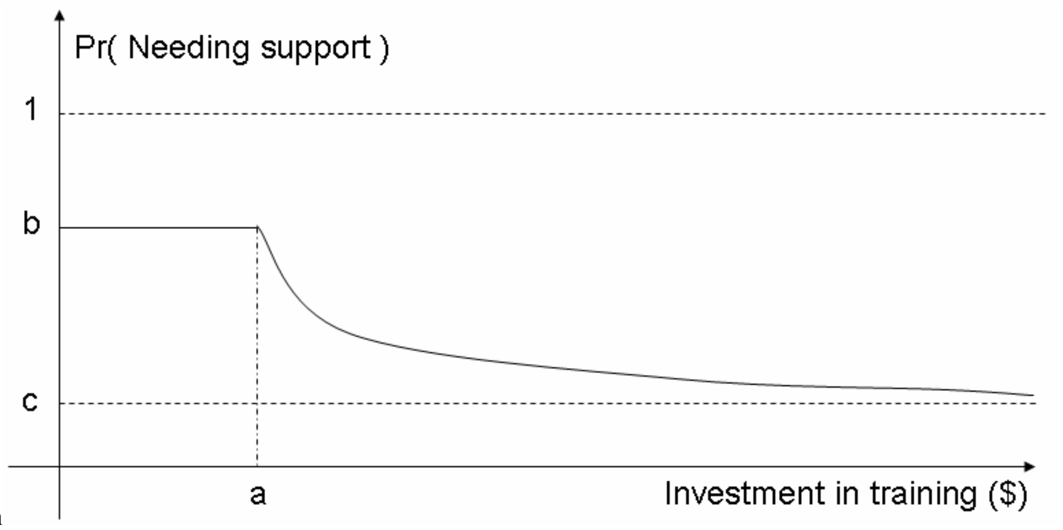
First, the Training transfer function: The idea is that this transfer function takes the portion of the overall security investment budget allocated for training and specifies the probability of the individual making support calls. As the budget for training increases, the individual becomes more proficient and needs to make fewer and fewer support calls. We assume, however, that there is always a background need to make some support calls; for example, to do with aligning the USB encryption with organizational systems configurations. Thus the transfer function has output in $[0, 1]$ and is monotonically decreasing with increasing training budget. We further assume that a minimal amount of training is needed before there is any reduction in the probability of an individual making a support call. The form we have chosen for this function, where inv is the investment variable, is:

$$trainingTF(inv) = (b - c)(\min(1, a/inv)) + c,$$

illustrated in Figure 1; the parameters a, b, and c are defined as follows:

- a = minimum training investment threshold: The amount of investment needed before there is any effect on training and reduction on the probability of needing support;
- b = maximum probability of needing support: This value is attained when no training at all is given;

- c = minimum probability of needing support: We assume that there is a baseline, underlying need for IT support, no matter how trained the employees are. Clearly, we require $b \geq c$.



inin

Figure 1: The 'Training' transfer function

Second, the IT Support transfer function: The idea here is that as security investment in IT support increases, the probability of a successful interaction with support also increases. The transfer function shows how this investment affects this probability and this time monotonically increasing. Just as for training, there is a minimum amount of investment required before any benefit is realised. The form we have chosen for this function is:

$$ITsupportTF(inv) = \max(0, b(1 - a/inv),$$

illustrated in Figure 2; the parameters a and b are defined as follows:

- a = minimum IT support threshold: The minimum amount of investment required before there is any effect on the probability of the success of IT support;
- b = maximum probability of successful support: This is naturally a limiting value, which we assume can be achieved arbitrarily closely.

Finally, the Compliance Monitoring transfer function: The idea here is that as security investment in compliance monitoring increases, this leads to an effective increase in the frequency with which compliance checks are made, so potentially improving the effectiveness of monitoring. Consequently, the time interval between checks will decrease. The transfer function specifying the time interval should therefore monotonically decrease as budgeted investment increases — the form of this function is conveniently chosen to be:

$$monitoringTF(inv) = (b - c)(\min(1, a/inv)) + c,$$

illustrated in Figure 3. The parameters a, b, and c are defined as follows:

- a = minimum monitoring investment threshold: The minimum amount of investment required before there is any reduction on the time interval between monitoring checks;
- b = maximum time interval between monitoring checks: A notional maximum amount of time between checks — in practice, this can simply be a very large number;

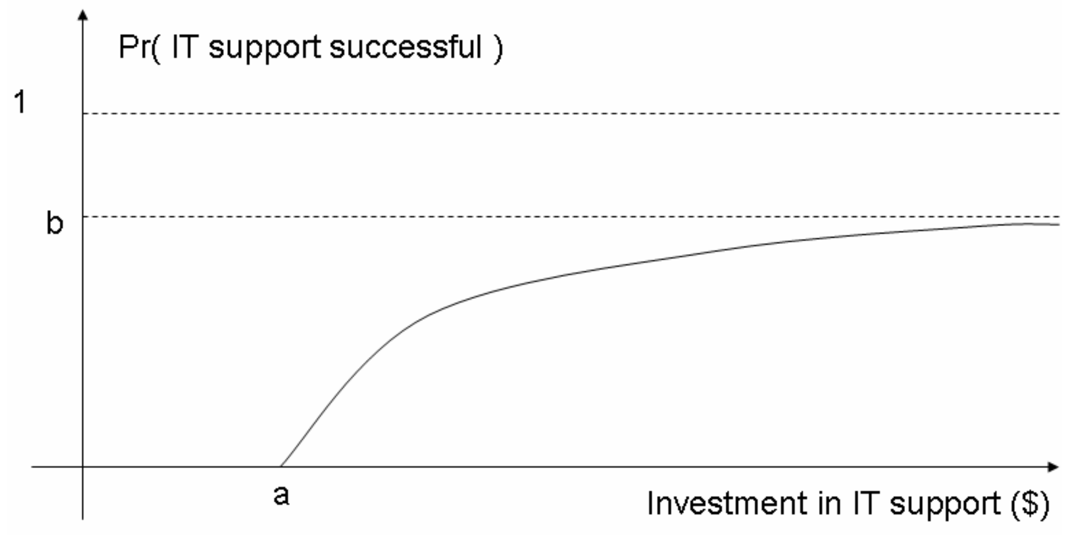


Figure 2: The 'IT Support' transfer function

- c = minimum time interval between checks: It is assumed that each check must take some amount of time to complete — thus the time interval *between* these checks cannot be less than this. Clearly, we require $b \geq c$.

The transfer functions are used to determine the probability that a typical individual will employ encryption, in the manner intended by the security policy, when using a USB memory stick. Note that we are not in a position to give an analytic definition of this probability. Rather, this is the point at which we appeal to our empirical data and the simulations provided by our model (the code is given in Appendix B). A key component of the model is the *individual's scoring function*,

$$indScore : \mathbb{R}^4 \rightarrow \mathbb{R},$$

expressing an individual's cost–benefit over the following four indicators:

- Successful data transfers (*trf*) — successful transfer of data is treated as a proxy for an individual's productivity;
- Embarrassments (*emb*) — events which damage the reputation of the individual, such as inability to recall a password in the presence of a customer;
- Reprimands (*ding*) — management may reprimand individuals for failing to comply with policy, and repeated reprimands may lead to serious sanctions;
- Negative experiences with IT Support (*nsup*) — interactions with IT Support may be unsatisfactory, and may fail to solve an individual's problem.

For the present study, we take the scoring function to be given by

$$indScore(trf, emb, ding, nsup) = dtSF(trf) + eSF(emb) + dSF(ding) + nsSF(nsup),$$

where $dtSF$, eSF, dSF, and $nsSF$ are chosen functions that capture the dependency of the overall score on the evident components. Note that the scoring functions eSF, dSF, and $nsSF$ are all negative-valued and decreasing because embarrassments, reprimands, and negative IT Support experiences all have negative impact on an individual's assessment of the cost-benefit trade-off of security activities.

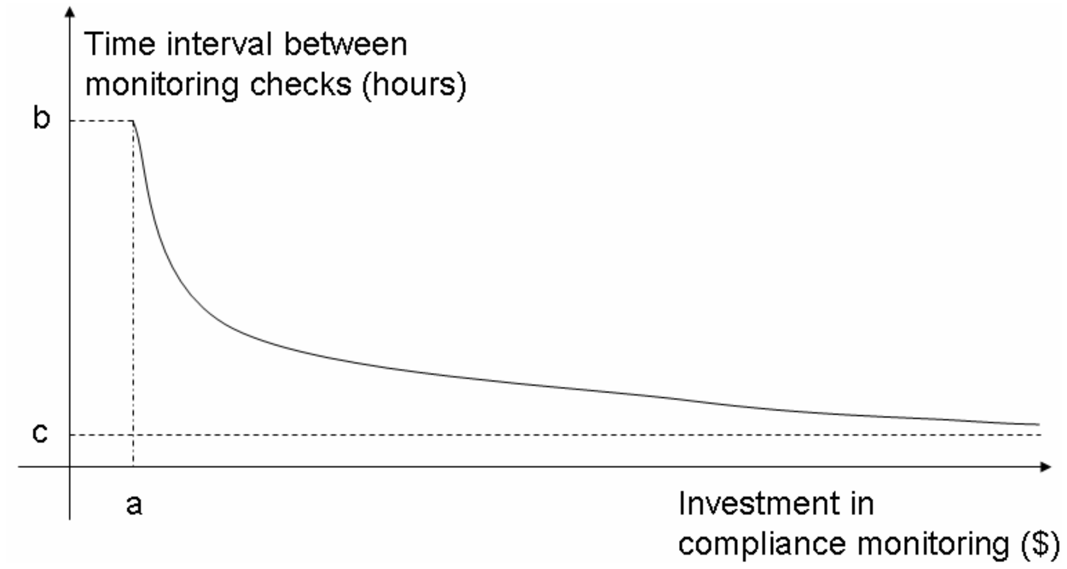


Figure 3: The 'Compliance Monitoring' transfer function

As usual, there are many reasonable choices for these functions, and we take simple exemplars. In all cases, the specific functions used depend on some specific 'calibration parameters'. Rather than consider these parameters in detail, we explain here just the general form of the functions.

First, the scoring function for successful data transfers, illustrated in Figure 4, captures the existence of a limit on the maximum possible reward to the individual, no matter how high his productivity:

$$dtSF(trf) = a(1 - \frac{b}{trf + b}),$$

where $a, b > 0$ are calibration parameters.

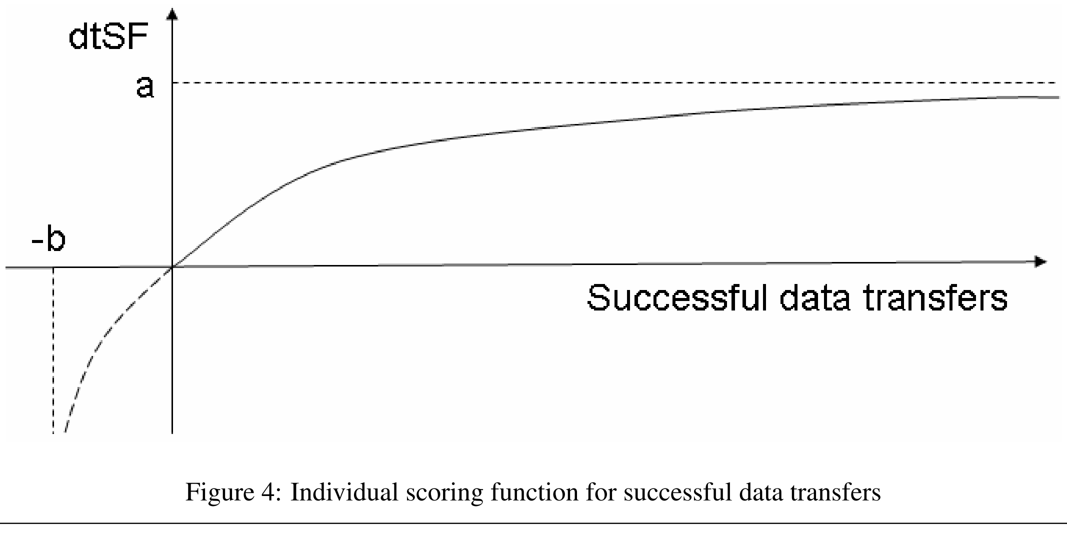


Figure 4: Individual scoring function for successful data transfers

Personal embarrassments reduce the individual's score, so the scoring function eSF, illustrated in Figure 5, is negative decreasing; we assume that costs of embarrassments accumulate unboundedly:

$$eSF(emb) = -a(emb),$$

where $a > 0$ is a calibration parameter.

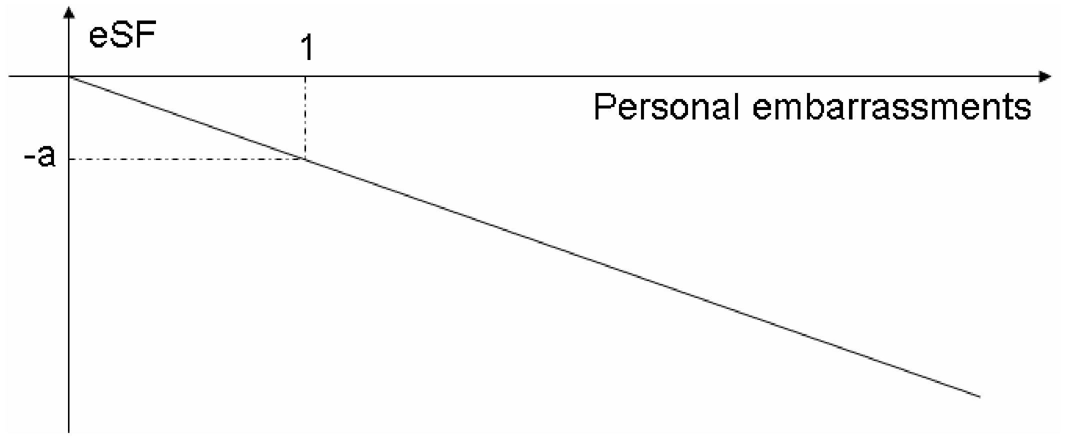


Figure 5: Individual scoring function for personal embarrassments

Reprimands from management also reduce an individual's score, and the greater the number of reprimands, the smaller the effect of subsequent reprimands. The function dSF, illustrated in § 6, has the following form:

$$dSF(ding) = a(\frac{b}{ding + b} - 1),$$

where $a, b > 0$ are calibration parameters.

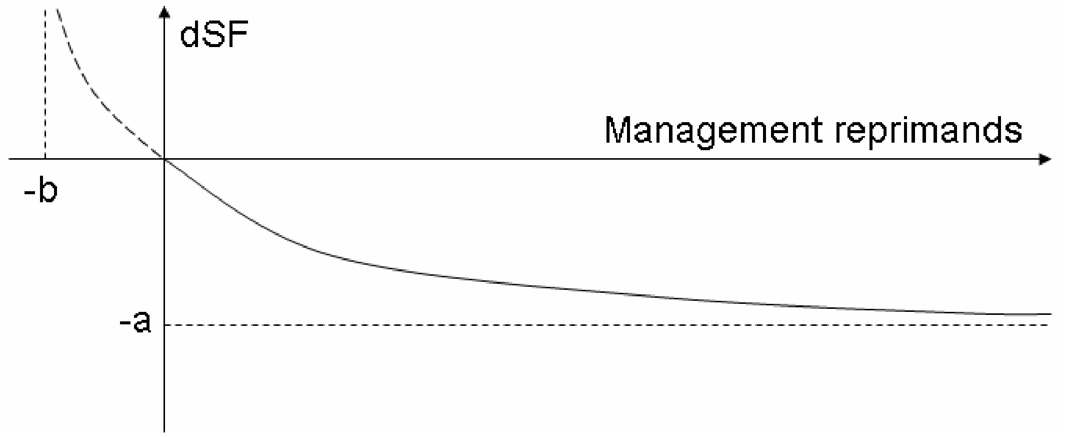


Figure 6: Individual scoring function for management reprimands

Finally, we consider the function $nsSF$, illustrated in Figure 7. Here we assume that the user's response to his failing to receive adequate support deteriorates as he experiences more such failures. We expect that it eventually overrides other factors, representing the encryption technology's becoming unusable and being given up. We take

$$nsSF(nsup) = -a(nsup^2)$$

with a calibration parameter $a > 0$.

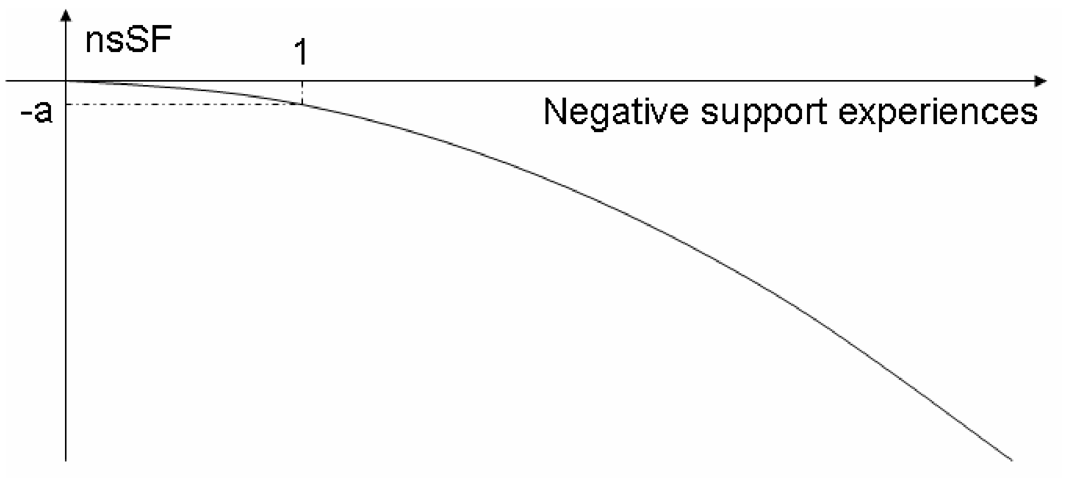


Figure 7: Individual scoring function for support failures

The typical individual's probability of using encryption is now obtained as follows:

- By using the above transfer and scoring functions, the model essentially becomes a function with a number of input parameters that maps over security investment, then security budget proportions, then probability of encryption, resulting in an overall numerical score as output. Formally, this is:

$$\begin{aligned} \text{model} \quad : \quad & \text{security-investment} \rightarrow \text{security-budget-proportions} \rightarrow \\ & \text{probability-of-encryption} \rightarrow \text{score} \end{aligned}$$

  Intuitively, this function represents the typical individual's score given all these input parameters. We also assume, however, that the typical individual responds rationally to the organizational environment (as determined by the security investment and the security budget proportions) by choosing how frequently he uses encryption, so as to maximize his perceived overall score. This rational maximization of benefit by the typical individual is therefore the basis for choosing the encryption probability;

- Mathematically speaking, our procedure for computing the probability p of encryption is to take $p \in [0, 1]$ such that p is the (unique) value that maximizes the overall score as a function of security investment and security budget proportions:

$$\sup\{model(sec)(sec{-}budget)(p) \in \mathbb{R} | p \in [0, 1]\}$$

  where $sec \in sec{-}range$ and $sec{-}range$ is a subset of $\mathbb{R}$, representing the range of security investments to be investigated and where $sec{-}budget$ ranges over the budgetry splits we could make (e.g., IT support, etc.) Technically, this function might have several optima as p ranges over [0,1]; that is unlikely since the transfer and scoring functions are clearly monotonic (and also concave/convex) and we assume that they are sufficiently smooth for there to be a unique choice maximizing the score;

- This function is expressed in terms of an executable discrete-event model involving stochastically generated events (see § 5). Therefore, the numerical answers that we obtain are generally approximate. In effect, the computation we are making involves fixing discrete values for the security investment, the security budget proportions and then performing a range of experiments ranging over discrete values for the probability of encryption. Each of these experimental variations are then performed a large number of times in order to obtain statistically valid outcomes from which we choose the probability value that maximizes the score. Intuitively, the multiple runs performed for each of the choices taken represents finding the average score over our typical population (we assume, for now, a homogeneous population).

The probability of using encrytion has direct consequences for the utility function that derives from the model. The calculation of this function is explained in § 6.

5 An Executable Model

The conceptual model described in the previous section is reified using our modelling tool, Demos2k [Dem, Bir79], which executes discrete event models of systems of resources and processes. Demos2k has a rigorous mathematical semantics [BT93, BT94, BT98, BT01a, BT01b] based on process algebra [Mil83, Mil89, PT06, PT07], which can be understood in both asynchronous and synchronous terms. Our modelling technique is to deploy the discrete mathematical tools of resource semantics [Pym02, PT06, PT07], process algebra [Mil89, PT06, PT07], and probability theory/stochastic processes [Dem, Tof94] in the style of classical applied mathematics (see [YMP06] for another example of the approach); that is, we identify levels of abstraction that are appropriate to the questions of interest, and avoid representing irrelevant detail.

We model the life-history of the composite entity 'a typical individual together with his current USB stick' to illustrate how various forms of risk are encountered within a given amount of time. By modelling these risk encounters explicitly, we can obtain a better quantitative picture of how the risks identified are naturally distributed. Modelling this composite entity (i.e., the 'user') allows us to ignore aspects of an individual's own life that do not involve any dealings with the USB stick.

For there to be any risk to confidentiality or availability, we need to introduce some particular sources of hazard. For this investigation, there are two principal components contributing to the hazards that arise: the user's physical location and the categories of people with whom the user intentionally or unintentionally shares data. For the purposes of this model, we broadly categorize the people we share data with as follows: whether they are a colleague or business partner who might legitimately share the information (i.e., a 'Friend'), or someone who will actively misuse the information gained to somehow harm the organization or the user (i.e. a 'Foe'), or, finally, someone who appears to the user as a Friend but *in actual fact* acts like a Foe (i.e., a 'Traitor'). Both of these aspects — location and categories of people we share data with — are explicitly represented in the model.

The outcome of running the model will be values of various performance indicators gathered as a part of simulating the life-histories:

- Number of successful data transfers to/from the USB device: This is used as a straightforward proxy for productivity — we assume that using a USB stick to transfer data has business benefit;

- Total number of exposures: Occasions on which information was transferred to either a Foe or a Traitor;

- Total number of 'reveals': A 'reveal' is less significant than an exposure and arises when a colleague or business partner (i.e., a Friend) is given information that they did not have a right to see. Because they are Friends, they are not expected to use that information to cause harm to the organization or the user. One way in which this can arise is via 'accidental archiving' — information that was unintentionally made available alongside other information that was intended to be shared.

Various other indicators are also gathered as output from each run; these have already been discussed in § 4.

The model comprises three main concurrent processes: *lifeUSB*, *movement*, and *measure*:

- lifeUSB: This process captures the activities of the 'individual plus his USB stick'. The user essentially interacts with different kinds of people in different locations, and indicators are accumulated as a result. Particular events involving the USB stick, such as $add/modify$, $write$, $delete$, etc., are randomly selected according to (discrete) probability distributions, conditional upon current location. As a result of these actions and interactions, we use a combination of time penalties and indicators to capture and account for the risks encountered.

- Movement: This process concurrently and independently moves the user from location to location, spending some time in each place. The different locations we use are:

  - Home: The user's personal home;
  - Desk: The main place of (solitary) work for the user;
  - Conf: This is where business meetings with Friends (and, potentially, Traitors) occur;
  - BizClient: Business meetings/workshops/conferences with business partners or other actors (i.e., principally Friends, but with some potential for talking to Traitors and Foes);
  - InTransit: This represents intermediate locations (e.g., on a plane, in a hotel, in a car) between main locations.

  Each location naturally has its own associated risks and opportunities for interaction. The transitions between locations follow the graph presented in Figure 8. Note that we assume that the user can move directly between the workplace locations Desk and Conf without going via the riskier InTransit location. Future locations for the user are chosen according to a location-dependent probability distribution, as well as the period of time they spend there;

- Measure: A book-keeping process that samples the various indicators gathered on a regular basis throughout each run.

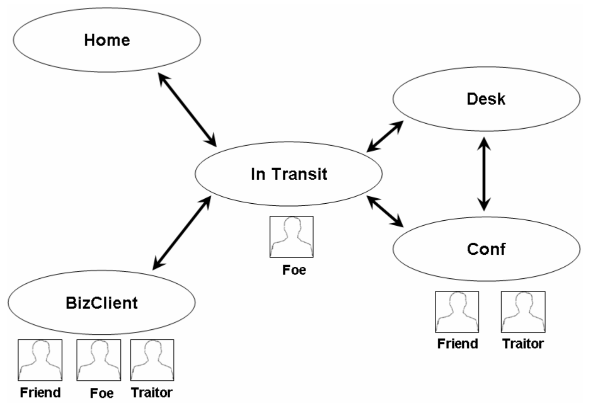


Figure 8: Locations and roles

6 The Experimental Space

Now we have our executable model, we can use it to explore how the level of security investment by an organization is connected to particular levels of availability and confidentiality, as modulated and affected by changes in typical employee behaviour, vis-à-vis his use of USB memory sticks. The organization's choices of amount and balance of security investment affect the usage of encryption on USB sticks by typical employees. This usage results in levels of information availability and confidentiality loss, which translate into business value for the organization.

Our experiments, performed using Demos2k [Dem] and its DXM experiment manager [Mon08], varied the following numerical instruments:

- Security Investment: This indicates the level of yearly investment per individual in security related cost. The range we have explored is: 20, 50, 100, 200, 500;

- Budgetary Proportions: Although we have three areas in which to invest — *training, IT support* and *monitoring* — we have specified a fixed value of training, since it is a one-off cost. So we have investigated the trade-off between investment in IT support on the one hand, and monitoring on the other. In practice, we have choosen to investigate 3 values of support proportion: 0.25, 0.5 and 0.75[1].

Each of these 15 (3×5) sample points represents a particular experimental variation. Following the approach to obtaining the individual's probability of using encryption, explained in $\S$ 4, within each of these variations we then need to range over $Pr(Enc)$, the probability of encryption, (from 0.1 to 0.9 in steps of 0.2) and finally run each of these 300 times to obtain results of some statistical value.

For simplicty of presentation in this paper, we have had to restrict the number of experimental simulations, and so we have adopted a coarse-grain 'sampling' strategy to choose parameters. We plan to conduct a more thorough and systematic experimental investigation based on empirical evidence to support the form of the transfer and scoring functions; where that is not possible, we hope to perform a systematic investigation of the space of parameters. The objective of such an investigation is to provide detailed guidance for conditioning economic models of the kind we have discussed.

6.1 Exploratory Fit of Additional Calibration Parameters

The transfer and scoring functions given are each dependent upon a number of numerical parameters — at this stage, it has not been possible to find obvious choices for these parameters — there are no easy and obvious sources of data, and there are no 'natural scales' that we could obviously exploit in order to make considered and easily justified choices. Further empirical study and experimental work will be necessary to address this issue.

Instead, we have taken the pragmatic decision to make choices of these parameters that illustrate a range of behaviour. To do this, we have conducted a series of exploratory (ad hoc) searches through the space of additional calibration parameters, helping to locate values of these parameters that yield useful observable output. We cannot claim therefore that this study has given definitive or canonical results. We instead claim that there is evidence here for examining the connections between these concerns in greater depth.

6.2 Some Confirmation of Expected Behaviour

As investment in monitoring and IT Support increased, we expected to see greater use of encryption; that was observed.

We expected to see a variation in the effectiveness of that investment as the proportion spent on IT Support vs. Monitoring was varied. As illustrated by the results below, we did not observe any such effect: the influence of a given level of investment is roughly the same for different proportions. We expected to be able to see a gradual increase in the use of encryption as investment increased, but the results show a fairly sharp transition from probability of encryption of 0.1 to 0.9 between investment values of 100 and 200. (Examining the data in more detail than shown here emphasizes this effect. The individual's optimal choice of probability (as computed from the experimental results) is always at one of the extremes, and never at a middle value.) We also expected that, above and below certain limits, there would be little extra effect from further increasing or reducing the investment level; this is not contradicted by the model (it is mildly confirmed).

6.3 Results

In $\S$ 4, we described how to extract information about our estimate for *Pr(Enc)* for a given level of security investment and budgetary proportions, based upon the individual's scoring function. Intuitively, this value is the one that produces the maximum value of this scoring function at that investment level.

The table below gives the value of *Pr(Enc)*, for the budgetary proportion dedicated to IT support vs security investment:

[1]A support proportion of 0.25 means that 1/4 of the total security investment goes towards IT support and the remainder goes towards monitoring.

	20	**50**	**100**	**200**	**500**
0.25	0.1	0.1	0.1	0.9	0.9
0.5	0.1	0.1	0.1	0.9	0.9
0.75	0.1	0.1	0.1	0.9	0.9

This table shows that, for security investment of 100 and below, the user's best choice is *Pr(Enc)* = 0.1; that is, rarely to use encryption. For security investment of 200 and above, the user's best choice is *Pr(Enc)* = 0.9; that is, nearly always to use encryption. (We did not consider *Pr(Enc)* of 0 or 1 because such utterly consistent user behaviour is rare.)

Next we tabulate the observed values of the availability measure and of the confidentiality measure over the 15 sample points, with the user's *Pr(Enc)* fixed at the corresponding value shown in the table above.

The availability measure is chosen to be the average number of successful data transfers per year carried out by the user. This is under the assumption that the purpose of the USB stick is to enable the user to transfer data on behalf of the organization.

	20	**50**	**100**	**200**	**500**
0.25	165.0933176	164.0433177	165.106651	161.2066513	161.1899847
0.5	163.4533178	163.5266511	165.5766509	162.6299845	161.453318
0.75	164.7299843	165.6333176	164.2733177	161.2266513	161.6966513

The confidentiality measure we use is a linear combination of the average number of events when confidential data is exposed and the average amount of confidential data exposed, both per year.

	20	**50**	**100**	**200**	**500**
0.25	10.02999905	8.26666588	9.326665779	5.85666611	6.626666036
0.5	8.176665889	7.876665917	9.123332465	6.106666086	6.886666012
0.75	9.519999094	7.966665909	8.569999185	6.449999386	5.486666145

We can observe that there is a substantial change in both the organization's availability and confidentiality measures as the user's probability of using encryption, *Pr(Enc)*, changes from 0.1 to 0.9.

The results are all obtained as averages over 300 independent runs. These values conservatively have a standard error of less than 10% of the values in the table. Given the number of runs required, it seems that the standard error could be halved by performing 1200 runs.

All of these results are preliminary. Further, and quite extensive, experimental work will be required to obtain adequate confidence interval estimates for the numbers quoted above.

6.4 A Utility Function

We have discussed, in § 2, a utility function approach to understanding the trade-offs between availability and confidentiality. We suggest that the simplest utility function it seems reasonable to postulate is one of the form

$$U(C, A) = \alpha(A - \beta C),$$

where α and β are parameters, which captures a simple ratio between confidentiality and availability.

Below are some tabulations of values for this function for different values of α, β, based upon the tables of availability and confidentiality numbers presented above. Exploring parameters of the utility function, illustrated in the tables below, we see that for values of $\beta = 10$ or 3, as spending on support and monitoring increases, the gain from increased confidentiality clearly outweighs the consequent loss of availability. $\beta = 0.1$ results in the loss in availability as spending increases outweighing the gain in confidentiality. Values of β in the region of 1 didn't give us useful results for utility, because statistical variation in experimental results swamps the difference between availability and confidentiality components of utility.

	20	**50**	**100**	**200**	**500**
0.25	75.44987339	94.76066264	83.65550334	119.52117	110.5353456
0.5	95.12164829	98.70045181	86.57055909	118.2674243	107.814368
0.75	80.96557825	100.1055786	91.49626593	112.6352725	124.4002981

Figure 9: Utility function for $\alpha = 1.164$, $\beta = 10.000$

	20	**50**	**100**	**200**	**500**
0.25	96.33782579	99.36347244	97.85302782	102.4985371	100.8382374
0.5	99.13512178	99.82968844	98.62371136	102.979025	100.4695461
0.75	97.17035435	101.1403263	98.87822724	101.2426083	103.6402906

Figure 10: Utility function for $\alpha = 0.714$, $\beta = 3.000$

7 Conclusions and Directions

We have reported a preliminary study. We have postulated an economic model that is suitable for capturing the utility of trade-offs between investments against confidentiality and availability in the context of the use of USB memory sticks in a financial services company. Building on empirically obtained data and on informed observations concerning policy and technology, we have used a process model to demonstrate that the hypothesized trade-off between confidentiality and availability does indeed exist, so providing evidence for the validity of the model, and to investigate the behaviour of a simple version of this model, giving good evidence to support the approach and motivate further study. We have established that individuals make cost–benefit decisions from their own (economic) perspective; we suggest organizations must understand that when making invetment decisions.

The following is a brief list of possible research directions:

- Further exploration of our experimental space, with substantial statistical analyses to inform the detailed formulation of economics models of the kind we have discussed;
- Mathematical and computational studies of the properties of these models;
- An investigation of game-theoretic approaches to the utility of the allocation security investment resources against competing priorities such as confidentiality and availability;
- More basic empirical studies of the kind we have described; for example, more studies of portable data storage media, or studies of network access control policies;
- Developments of our process modelling tool better to handle the structure of distributed systems.

The work reported here is the result of a highly interdisciplinary study. Such an approach seems to us to be necessary to make progress in this area.

8 Acknowledgements

We are grateful to the many members of staff of HP Labs and Merrill Lynch who generously gave their time to take part in our empirical studies. We also thank Jean Paul Degabriele for his advice on the early stages of this work.

	20	**50**	**100**	**200**	**500**
0.25	100.9077518	100.3704883	100.9592029	98.77427677	98.71667626
0.5	100.013201	100.0767461	101.2607345	99.63418527	98.86262497
0.75	100.7156816	101.3667113	100.4932739	98.75008864	99.09835674

Figure 11: Utility function for $\alpha = 0.615$, $\beta = 0.100$

References

[AM06] R. Anderson and T. Moore. The economics of information security. *Science*, 314:610–613, 2006. Extended version available at `http://www.cl.cam.ac.uk/~rja14/Papers/toulouse-summary.pdf`.

[And01] R. Anderson. Why information security is hard: An economic perspective. In *Proc. 17th Annual Computer Security Applications Conference*, 2001.

[AS99] Anne L. Adams and M. Angela Sasse. Users are not the enemy: Why users compromise security mechanisms and how to take remedial measures. *Communications of the ACM*, 42(12):40–46, 1999.

[Bir79] G. Birtwistle. *Demos — discrete event modelling on Simula*. Macmillan, 1979.

[BT93] G. Birtwistle and C. Tofts. An operational semantics of process-orientated simulation languages: Part I πDemos. *Transactions of the Society for Computer Simulation*, 10(4):299–333, 1993.

[BT94] G. Birtwistle and C. Tofts. An operational semantics of process-orientated simulation languages: Part II μDemos. *Transactions of the Society for Computer Simulation*, 11(4):303–336, 1994.

[BT98] G. Birtwistle and C. Tofts. A denotational semantics for a process-based simulation language. *ACM ToMaCS*, 8(3):281 – 305, 1998.

[BT01a] G. Birtwistle and C. Tofts. Getting Demos Models Right — Part I Practice. *Simulation Practice and Theory*, 8(6-7):377–393, 2001.

[BT01b] G. Birtwistle and C. Tofts. Getting Demos Models Right — Part II ... and Theory. *Simulation Practice and Theory*, 8(6-7):395–414, 2001.

[Cen] Mathematica Documentation Center. http://reference.wolfram.com/mathematica/guide/mathematica.html.

[CPP06] Mark Clatworthy, David Peel, and Peter Pope. Are analysts' loss functions asymmetric? Technical Report 005, Lancaster University Management School, 2006.

[Dem] Demos2k. http://www.demos2k.org.

[GL02] L.A. Gordon and M.P. Loeb. The Economics of Information Security Investment. *ACM Transactions on Information and Systems Security*, 5(4):438–457, 2002.

[GL06] L.A. Gordon and M.P. Loeb. *Managing Cybersecurity Resources: A Cost-Benefit Analysis*. McGraw Hill, 2006.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25(3):267–310, 1983.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[Mon08] B. Monahan. Dxm: Demos experiments manager. Forthcoming HP Labs Technical Report, 2008.

[NP] R.A. Nobay and D.A. Peel. Optimal Monetary Policy in a Model of Asymmetric Bank Preferences. London School of Economics, *Mimeo*.

[PT06] David Pym and Chris Tofts. A calculus and logic of resources and processes. *Formal Aspects of Computing*, 18(4):495–517, 2006. Erratum (with Collinson, M.) *Formal Aspects of Computing* (2007) 19: 551-554.

[PT07] David Pym and Chris Tofts. Systems Modelling via Resources and Processes: Philosophy, Calculus, Semantics, and Logic. In L. Cardelli, M. Fiore, and G. Winskel, editors, *Electronic Notes in Theoretical Computer Science (Computation, Meaning, and Logic: Articles dedicated to Gordon Plotkin)*, volume 107, pages 545–587, 2007. Erratum (with Collinson, M.) *Formal Aspects of Computing* (2007) 19: 551-554.

[Pym02] D.J. Pym. *The Semantics and Proof Theory of the Logic of Bunched Implications*, volume 26 of *Applied Logic Series*. Kluwer Academic Publishers, 2002. Errata and Remarks maintained at: `http://www.cs.bath.ac.uk/~pym/BI-monograph-errata.pdf`.

[RG83] R.Barro and D. Gordon. A Positive Theory of Monetary Policy in a Natural Rate Model. *Journal of Political Economy*, 91:589–610, 1983.

[RM01] Francisco J. Ruge-Murcia. The inflation bias when the central bank targets the natural rate of unemployment. Technical Report 2001-22, Département de Sciences Économique, Université de Montréal, 2001.

[RM03] Francisco J. Ruge-Murcia. Inflation targeting under asymmetric preferences. *Journal of Money, Credit, and Banking*, 35(5), 2003.

[SC90] Anselm L. Strauss and Juliette M. Corbine. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. Newbury Park, CA: Sage, 1990.

[ST04] Office Of Science and Technology. Foresight: Cyber trust and crime prevention project: Executive summary. 2004.

[Tay93] John B. Taylor. Discretion versus policy rules in practice. *Carnegie-Rochester Conference Series on Public Policy*, 39:195–214, 1993.

[Tof94] C. Tofts. Processes with probability, priority and time. *Formal Aspects of Computing*, 6(5):536–564, 1994.

[Var74] H. Varian. A bayesian approach to real estate management. In S.E. Feinberg and A. Zellner, editors, *Studies in Bayesian Economics in Honour of L.J. Savage*, pages 195–208. North Holland, 1974.

[WS01] Dirk Weirich and M. Angela Sasse. Pretty good persuasion: A first step towards effective password security for the real world. In *Proceedings of the New Security Paradigms Workshop, Cloudcroft, NM, September 2001)*, pages 137–143. ACM Press, 2001.

[YMP06] M. Yearworth, B. Monahan, and D. Pym. Predictive modelling for security operations economics (extended abstract). In *Proc. I3P Workshop on the Economics of Securing the Information Infrastructure*, 2006. Proceedings at `http://wesii.econinfosec.org/workshop/`.

[Zel86] A. Zellner. Bayesian prediction and estimation using asymmetric loss functions. *Journal of the American Statistical Association*, 81:446–451, 1986.

A Empirical Data

The following data tables represent our empirical results obtained from interview:

Start-up costs

Action	Individual Cost	Demos Repn.	Organizational Cost	Demos Repn.
Initial purchase of sufficient USB sticks.	None.		Financial outlay for hardware, administration of making/processing order.	Start-up cost
Purchase of controlling software and licenses.	None.		Financial outlay of software, administration of making/processing order.	Start-up cost
Installation of the software across the IT network and associated mobile devices.	Disrupted work due to workstations being taken offline for maintenance.	None.	IT staff focused on the project potentially lengthening completion times on other more routine tasks.	Start-up cost
Staff training.	Time taken up by training. Depending on effectiveness may be seen as time wasted by staff creating resentment.	None.	Instructors need to be either outsourced creating additional costs and administration (security clearance etc) or taken from within the organization meaning their usual role is unfulfilled for the duration.	training budget (start-up)

Start up costs (cont'd)

Action	Individual Cost	Demos Repn.	Organizational Cost	Demos Repn.
Maintenance of controlling software.	Periodic disruption to workstation as new patches/updates are installed.	None.	Increased workload on IT support staff, may necessitate adding staff to the payroll.	None.
Setting up user support mechanisms (documentation, web pages, help desk, etc.)	None.	None.	Creation and maintenance of web site, help desk staff costs and training.	yearly support budget
Monitoring compliance with policy.	Depending on the method used may disrupt employee's work. If it is poorly handled can also create feeling of being spied on that will be counterproductive.	None.	Additional work for support staff, depending on method. If ineffective then resources will be simply wasted attempting this task.	yearly monitoring budget
Requirement to recall password.	Increased cognitive load on user.	None.	None.	None.
Change in working habits required by new technology.	Restricted working practices due to more stringent policy. Additional stresses in the short term as people are resistant to change in their habits.	None.	None.	None.

Organizational costs

Action	Expected Cost	Failure Event	Failure Cost	Demos Repn.
USB stick is taken from storage and connected to computer.	Negligible.	Hardware or software failure prevents USB stick being connected/identified by the computer.	Time investment of support staff to reissue the USB stick, small financial cost of replacement.	Treated as a call for support.
Data is copied from computer onto the USB stick and is encrypted during transmission.	Negligible.	Data is incorrectly encrypted.	None unless the error is not spotted now, causing the failure of the presentation later.	Modern encryption techniques and technology makes this failure event very unlikely and as such it is not included in the Demos model. It appears here as it is a concern raised in the empirical data by some less informed participants.
USB removed from computer and transported to client's location.	Financial cost arranging transportation.	USB stick lost/stolen while in transit.	Potential loss of face if failure is made public. If encryption was not used the loss could be much higher and include a financial and/or legal element.	loss event (includes theft), possible exposure if found by Foe; replacement cost for USB.
USB removed from transport storage and connected to client's computer.	Negligible.	Hardware or software failure prevents the USB stick being read.	Inherited reputation loss as seen to employ inefficient employees.	Not explicitly represented as such; would appear as a support call.

Organizational costs (cont'd)

Action	Expected Cost	Failure Event	Failure Cost	Demos Repn.
Data is unencrypted and copied to client's computer.	Negligible.	Recall failure of passwords prevents data being accessed.	Inherited reputation loss as seen to employ inefficient employees.	Embarrassment (also one less successful transaction)
Data is left on the client's computer at their request (they wish to retain a copy of the presentation).	Negligible.	Confidential data accessed by an unintended party such as an untrustworthy client or competitor.	Exposure of confidential data.	Exposure event if client or colleague is a Traitor, reveal if Friend.
USB stick is removed from client's computer and transported back to the organization's location.	Financial cost arranging transportation.	USB stick lost/stolen while in transit.	Potential loss of face if failure is made public. If encryption was not used the loss could be much higher and include a financial and/or legal element.	Loss event (includes theft), possible exposure if found by Foe; replacement cost for USB.

Individual costs

Action	Expected Cost	Demos Repn.	Failure Event	Failure Cost	Demos Repn.
USB stick is taken from storage and connected to computer.	Small delay while computer finds and identifies new hardware.	None.	Hardware or software failure prevents USB stick being connected/identified by the computer.	Significant delay and associated stress while a new company-authorised USB stick is found and delivered to the user. Margin of error as regards time eroded.	Treated as a call for support with possible 'negative support experience'.
Data is copied from computer onto the USB stick and is encrypted during transmission.	Delay while data is moved and encrypted. If user is used to transferring unencrypted data this will seem to take a long time, creating impatience.	Time penalty.	Data is incorrectly encrypted.	None if the data is still readable. Significant cost if the data cannot be read and the error is not detected now.	Modern encryption techniques and technology makes this failure event very unlikely and as such it is not included in the Demos model. It appears here as it is a concern raised in the empirical data by some less informed participants.,

Individual costs (cont'd)

Action	Expected Cost	Demos Repn.	Failure Event	Failure Cost	Demos Repn.
Alternatively user opts for a workaround or ignores policy and copies unencrypted data onto the USB stick.	None; may even be seen as a benefit as time and effort have been saved.	This choice is controlled by the function 'probUseEncryption'.	Organization's monitoring discovers policy breach.	Reprimand from management.	Dings from management.
USB removed from computer and transported to client's location.	Negligible.	Movement through the 'InTransit' location.	USB stick lost/stolen while in transit.	Maximum failure cost as presentation no longer possible. Associated embarrassment and loss of face with client. Stress about job security will also occur.	Loss event, possible exposure if found by Foe, and consequent ding from management; time penalty to replace USB.
USB removed from transport storage and connected to client's computer.	Small delay while computer finds and identifies new hardware.	None.	Hardware or software failure prevents the USB stick being read.	Maximum failure cost as presentation no longer possible. Associated embarrassment and loss of face with client. Stress about job security will also occur.	Treated as a call for support with possible 'negative support experience'.

Individual costs (cont'd)

Action	Expected Cost	Demos Repn.	Failure Event	Failure Cost	Demos Repn.
Data is left on the client's computer at their request (they wish to retain a copy of the presentation).	Negligible.		Confidential data accessed by an unintended party such as an untrustworthy client or competitor. Exposure of confidential data.		
Data is unencrypted and copied to client's computer.	Some delay as data is unencrypted and moved. If user is used to moving unencrypted data then this will seem like a longer delay and cause increased stress due to the client.	Time penalty.	Recall failure of passwords prevents data being accessed.	Maximum failure cost as presentation no longer possible. Probably greater than hardware/software failure as blame is more closely linked to the individual.	Possible embarrassment depending on location (also one less successful transaction).
USB stick is removed from client's computer and transported back to the organization's location.	Negligible.	Movement through the 'InTransit' location.	USB stick lost/stolen while in transit.	Small scale stress and time investment in seeking replacement and concern over personal reputation within the organization.	Loss event, possible exposure if found by Foe, and consequent ding from management; time penalty to replace USB.

B The Demos2k Model

```
(* USB risk study

   Model of data transfer between different locations and players
   to describe various risks etc.

   LIFECYCLE elements of risk model:

   Players:
      Holder  - the USB stick's "main user".
      Friend  - legit. colleague of holder.
      Traitor - Malicious (internal) agent.
      Foe     - Malicious (external) agent.

   Locations:
      HOME - Holders HOME (zero risk of capture by Foe).
      DESK - Holders business base (low risk of capture of capture by Foe).
      CONF - Business meeting (containing friend only).
      BIZCLIENT - Business meeting - possible capture opportunity.

      TRANSIT - intermediate location - hotel and transport
      (e.g. car, plane, etc).

   2nd itn: BM (based upon JFG verbal comment)
   - eliminated "data" - no need to represent data - just use counts.
   - eliminated bins for counts everywhere
   - changed USB bin into counts for encrypted and decrypted data

   Init version - BM

*)



(*=====================================================|
|               TIMESCALING CONSTANTS                  |
|=====================================================*)

(*  The approach taken here is to:

  -  Assume a "working day" of 8hrs within which USB relevent events can
     happen including all relevent movements of location, such as travelling
     to/from business clients or travel to/from home.  We don't model the fact
     that travelling to and from home is regular everyday, nor holidays, etc
     , etc.

  -  Choose a negexp distribution of times between USB events having a
     highish average time (e.g. around three hours) to capture the fact
     that other stuff happens.

*)


cons hrs        = 1;                 // time unit = hrs

cons mins       = hrs/60;
cons secs       = mins/60;
cons msecs      = secs/1000;

cons days       = 8 * hrs;          // days = working time
cons weeks      = 7 * days;
cons months     = 4 * weeks;
cons years      = 365 * days;
cons decades    = 10 * years;
cons centuries  = 100 * years;
cons millenia   = 10 * centuries;


//cons measureInterval = 1 * days;  // for closer observation
cons measureInterval = 1 * years;   // for fast runs



(*=======================================================|
|                    RUNTIME                            |
|=======================================================*)

cons runTime = (1 * years) + (1*secs);


(*=======================================================|
|               TIME COSTS & PENALTIES                  |
|=======================================================*)

// time interval between actions
cons timePasses = negexp(3 * hrs);

// time cost for using encryption/decryption
cons encryption_time_costs = negexp(2 * mins);
cons decryption_time_costs = negexp(2 * mins);

// time taken to replace password
cons password_replacement_time = normal(1 * hrs, 5 * mins);

// time taken to find a mislaid USB (chosen to give approx. 10% chance
// of giving up after 1 day)
cons mislaidTime        = negexp(0.4343 * days);
cons mislaidTimeLimit   = 1 * days;

// time penalty for replacing a USB
cons USB_replacement_time_penalty = negexp(3 * days);
```

```
(*=====================================================|
|              SLOW-CYCLE AND OTHER COSTS              |
|=====================================================*)

cons cash                  = 1;  // DOLLAR VALUE - 1 cash unit = 1$


cons activeSecurityInvestment  = (100 * cash);  //PARAMETER TO EXPLORE

cons supportProportion     = 1/2;               //PARAMETER TO EXPLORE


cons trainingBudget     =  (100 * cash);
cons supportBudget      =  (activeSecurityInvestment * supportProportion);
cons monitoringBudget   =  activeSecurityInvestment - supportBudget;


//  Budgets are done in absolute amounts or amounts per year.
//  Simple USB stick with bundled encryption
cons newUSBStick                = 100;  // This is also used
                                       // when a USB stick is lost
cons softwareLicence            = 0;
cons installationAndMaintenance = 0;  // Assumed basic enough that the
                                      // users just pick it up and use it
                                      // (or install the software themselves).
cons totalOneOffCosts           = newUSBStick
                                  + softwareLicence
                                  + installationAndMaintenance;


// Approach: Separate budgets for training, on-going support and
// monitoring for compliance, tracked as experiences of the holder
// but not directly influencing probUseEncryption in the Demos model

// Training and support costs and effectiveness
cons trainingInvestmentThreshold  = 50 * cash;  // must spend above this on training
                                                // to gain any benefit at all
cons trainingMaxProbNeedSup       = 0.1;        // chance per attempt of needing support
                                                // with no training at all
cons trainingMinProbNeedSup       = 0.001;      // no matter how much training, there'll
                                                // always be some need for support

try [trainingBudget > trainingInvestmentThreshold] then {
  cons probNeedSupport            = (trainingMaxProbNeedSup - trainingMinProbNeedSup)
                                    * (trainingInvestmentThreshold / trainingBudget)
                                    + trainingMinProbNeedSup;
} etry [] then {
  cons probNeedSupport            = trainingMaxProbNeedSup;
}

cons testNeedSupport              = binom (1, probNeedSupport);

cons supportInvestmentThreshold   = 5 * cash;  // Below a minimum spend, no useful support
cons supportMaxEffect             = 0.9;       // No matter what you spend, some support
                                               // calls won't get dealt with successfully

try [supportBudget > supportInvestmentThreshold] then {
  cons probSupportCallSucceeds    = supportMaxEffect *
                                    (1 - supportInvestmentThreshold / supportBudget);
} etry [] then {
  cons probSupportCallSucceeds    = 0;
}
cons testSupportCallSucceeds      = binom (1, probSupportCallSucceeds);
cons negativeSupportFailsTransaction = 1;  // Does a negative support experience cause
                                           // the attempted action or transfer to fail?

// Monitoring costs and effectiveness
cons monitorInvestThreshold   = 20 * cash;  // Below a minimum spend, no monitoring
cons monitorMaxMeanInterval   = 1 * years;  // Interval between checks at minimum spend
cons monitorMinMeanInterval   = 1 * hrs;    // Can't check faster than this

try [monitoringBudget >= monitorInvestThreshold] then {
  cons monitorActive          = 1;  // there is monitoring
  cons monitorMeanInterval    = (monitorMaxMeanInterval - monitorMinMeanInterval)
                                * (monitorInvestThreshold / monitoringBudget)
                                + monitorMinMeanInterval;
  cons monitorPolicyInterval  = negexp (monitorMeanInterval);
} etry [] then {
  cons monitorActive          = 0;  // no monitoring
}

cons monitorMinTimeToNextDing = 1 * weeks;  // If the holder has got a ding from
                                            // management, he can't get dinged again
                                            // until this interval has passed.


// Individual Scoring Function calibration parameters
cons dtSF_a = 572.8;
cons dtSF_b = 156.6;
cons eSF_a  = 1.271;
cons dSF_a  = 79.44;
cons dSF_b  = 100;
cons nsSF_a = 0.01708;

// Utility function parameters - *** PARAMETERS TO EXPLORE
cons alpha  = 1;
cons epsilon  = 1;

// components of confidentiality loss
cons gamma1 = 10;  // exposures
cons gamma2 = 1;
cons gamma3 = 0;  // reveals
cons gamma4 = 0;
```

```
(*======================================================|
|           EVENTS, COMMANDS and ACTIONS                 |
|======================================================*)

cons NON_EVENT                = -1000;  // event of not selecting something to do

cons ev__DO_USB_ACT           = 1001;   // event of performing a USB stick operation
cons ev__DO_USB_INTERACTION   = 1003;   // event of interacting with other players
cons ev__LOSE_USB             = 1004;   // event of losing current USB stick
cons ev__LOSE_PASSWORD        = 1005;   // event of losing the password
cons ev__CHANGE_PASSWORD      = 1006;   // event of changing the password

//  type ev = DO_USB_ACT | DO_CHANGE_LOC | DO_USB_INTERACTION
//          | LOSE_USB | LOSE_PASSWORD | CHANGE_PASSWORD



(*======================================================|
|               ENCRYPTION OPERATION CODES               |
|======================================================*)

cons opn__DO_NOTHING      = 2001;   // operation of doing no encryption
cons opn__ENCRYPT_DATA    = 2002;   // operation of performing encryption
cons opn__DECRYPT_DATA    = 2003;   // operation of performing decryption

// type opn = DO_NOTHING | ENCRYPT_DATA | DECRYPT_DATA


(*======================================================|
|                    USB ACTIONS                         |
|======================================================*)

cons usb_act__READ_DATA   = 3001;   // action of reading data from USB stick
cons usb_act__ADD_DATA    = 3002;   // action of adding data to USB stick
cons usb_act__DEL_DATA    = 3003;   // action of deleting data from USB stick
cons usb_act__WIPE_DATA   = 3004;   // action of wiping all data from USB stick

// type usb_act = READ_DATA | ADD_DATA | DEL_DATA | WIPE_DATA


(*======================================================|
|                   LOCATIONS                            |
|======================================================*)

cons loc__HOME            = 4001;   // holders' HOME
cons loc__DESK            = 4002;   // holders' office DESK  (i.e. solitary work)
cons loc__CONF            = 4003;   // holders' office MEETINGs
cons loc__BIZCLIENT       = 4004;   // holders' BUSINESS CLIENT
cons loc__TRANSIT         = 4005;   // holder in transit

// type loc = HOME | DESK | CONF | BIZCLIENT | TRANSIT



(*======================================================|
|                   PLAYERS / ROLES                      |
|======================================================*)

cons NOONE             = -5000;   // value representing a non-choice of a player.

cons player__HOLDER    = 5001;    // principal player: the holder of the USB stick
cons player__FRIEND    = 5002;    // holders colleague: someone to receive data
cons player__TRAITOR   = 5003;    // 'insider' adversary indistinguishable from colleague
cons player__FOE       = 5004;    // 'external' adversary
```

```
// type player = HOLDER | FRIEND | TRAITOR | FOE



(*=======================================================|
|                      PASSWORDS                          |
|=======================================================*)

cons pwd__NO_PASSWORD   = 6001;
cons pwd__HAS_PASSWORD  = 6002;

// type pwd = NO_PASSWORD | HAS_PASSWORD

// the status of the holders password ...
var holderPasswordStatus = pwd__HAS_PASSWORD;
```

```
(*====================================================|
|            MISC. DECISIONS  and CHOICES             |
|====================================================*)

// decision of adding new data
cons doAddNewData  = binom(1, 40/100);

// proportion of data added to the USB stick that is confidential -
// NB: whether data is confidential is treated as independent of
// whether the data is encrypted, for now at least
cons propConfidentialWrite  = 0.5;
cons testConfidentialWrite  = binom(1, propConfidentialWrite);

// proportion of data to delete
cons propDataDeleted = uniform(10/100, 60/100);

// probability of accidental copying unintended material when giving a
// USB stick data to someone
cons probAccidentalArchive   = 3/100;
cons testAccidentalArchive   = binom(1, probAccidentalArchive);

// probability that exposure happens, given that we have the opportunity to perform exposure,
cons probExposureHappensByFoe      = 30/100;
cons testExposureHappensByFoe      = binom(1, probExposureHappensByFoe);
cons probExposureHappensByTraitor  = 3/100;
cons testExposureHappensByTraitor  = binom(1, probExposureHappensByTraitor);

// probability that an exposure is detected by the organization
cons probExposureDetected          = 20/100;
cons testExposureDetected          = binom(1, probExposureDetected);


// (conditional) probability of FOE discovering/finding/aquiring a lost USB
cons prob_foe_findsUSB [ loc__HOME      ]  = 1/100;
cons prob_foe_findsUSB [ loc__DESK      ]  = 0;
cons prob_foe_findsUSB [ loc__CONF      ]  = 2/100;
cons prob_foe_findsUSB [ loc__BIZCLIENT ]  = 5/100;
cons prob_foe_findsUSB [ loc__TRANSIT   ]  = 5/100;


(*====================================================|
|                SIMPLE POLICY FLAGS                  |
|====================================================*)

cons allow_unencrypted_usage = 1;
cons allow_encrypted_usage   = 1;
cons allow_unencrypted_write_when_encryption_fails = 0;

cons autoDeleteInForce = 0;           // Is there an auto-deletion policy in force
cons autoDeletionPeriod = 2 * weeks;  // Check after autoDeletionPeriod (approx.) and
                                      // delete everything older than autoDeletionPeriod


(*====================================================|
|             ENABLE ADVERSE ROLE PLAYERS             |
|====================================================*)

// flag to control adverse roles
cons enable_TRAITOR_player    = 1;
cons enable_FOE_player        = 1;
```

```
(*=====================================================|
|        CHOICES OF ACTIONS and EVENTS (POLICIES)      |
|=====================================================*)

// probability of using encryption to protect data confidentiality
// Generally we vary this parameter directly in dxm
cons probUseEncryption = 0.5;

// choice of encrypting content when putting new data on the USB
cons chooseToEncryptData =
     pud [((1 - probUseEncryption), opn__DO_NOTHING),
          (probUseEncryption, opn__ENCRYPT_DATA)];

// USB actions for holder

cons choiceUSBAction =
     pud [ ( (43 / 100),   usb_act__ADD_DATA  ),
           ( (55 / 100),   usb_act__READ_DATA ),
           ( ( 1 / 100),   usb_act__DEL_DATA  ),
           ( ( 1 / 100),   usb_act__WIPE_DATA )
         ];


// Selection of events to happen to holder, conditioned upon location
// Game-theoretically, these are moves by Nature ...
cons chooseEventForHolder [loc__HOME] =
     pud [ ( (91 / 100),   NON_EVENT                ),
           ( ( 6 / 100),   ev__DO_USB_ACT           ),
           ( ( 0 /   1),   ev__DO_USB_INTERACTION  ),
           ( ( 1 / 100),   ev__LOSE_USB             ),
           ( ( 2 / 1000),  ev__LOSE_PASSWORD        ),
           ( (18 / 1000),  NON_EVENT                ),
           ( ( 0 /   1),   ev__CHANGE_PASSWORD     )
         ];

cons chooseEventForHolder [loc__DESK] =
     pud [ ( (95 / 100),   ev__DO_USB_ACT           ),
           ( ( 0 /   1),   ev__DO_USB_INTERACTION  ),
           ( ( 1 / 100),   ev__LOSE_USB             ),
           ( ( 2 / 1000),  ev__LOSE_PASSWORD        ),
           ( (18 / 1000),  NON_EVENT                ),
           ( ( 1 / 100),   ev__CHANGE_PASSWORD      ),
           ( ( 1 / 100),   NON_EVENT                )
         ];

cons chooseEventForHolder [loc__CONF] =
     pud [ ( (60 / 100),   ev__DO_USB_ACT           ),
           ( (35 / 100),   ev__DO_USB_INTERACTION  ),
           ( ( 1 / 100),   ev__LOSE_USB             ),
           ( ( 2 / 1000),  ev__LOSE_PASSWORD        ),
           ( (18 / 1000),  NON_EVENT                ),
           ( ( 0 / 100),   ev__CHANGE_PASSWORD      ),
           ( ( 2 / 100),   NON_EVENT                )
         ];

cons chooseEventForHolder [loc__BIZCLIENT] =
     pud [ ( (40 / 100),   ev__DO_USB_ACT           ),
           ( (57 / 100),   ev__DO_USB_INTERACTION  ),
           ( ( 1 / 100),   ev__LOSE_USB             ),
           ( ( 2 / 1000),  ev__LOSE_PASSWORD        ),
           ( (18 / 1000),  NON_EVENT                ),
           ( ( 0 /   1),   ev__CHANGE_PASSWORD     )
         ];
```

```
cons chooseEventForHolder [loc__TRANSIT] =
     pud [ ( (97 / 100),  NON_EVENT               ),   // nothing happens to the USB ...
           ( ( 0 /   1),  ev__DO_USB_ACT          ),
           ( ( 0 /   1),  ev__DO_USB_INTERACTION  ),
           ( ( 1 / 100),  ev__LOSE_USB            ),
           ( ( 2 / 1000), ev__LOSE_PASSWORD       ),
           ( (18 / 1000), NON_EVENT               ),
           ( ( 0 /   1),  ev__CHANGE_PASSWORD     )
         ];


// choosing who to interact with, conditioned by location
cons chooseInteraction [ loc__HOME ] =
     pud [ ( 1, NOONE            ),    // no-one to interact with
           ( 0, player__HOLDER   ),
           ( 0, player__FRIEND   ),
           ( 0, player__TRAITOR  ),
           ( 0, player__FOE      )
         ];

cons chooseInteraction [ loc__DESK ] =
     pud [ ( 1, NOONE            ),    // solitary working
           ( 0, player__HOLDER   ),
           ( 0, player__FRIEND   ),
           ( 0, player__TRAITOR  ),
           ( 0, player__FOE      )
         ];

cons chooseInteraction [ loc__CONF ] =
     pud [ (    0, player__HOLDER   ),
           ( 0.95, player__FRIEND   ),
           ( 0.05, player__TRAITOR  ),
           (    0, player__FOE      )
         ];

cons chooseInteraction [ loc__BIZCLIENT ] =
     pud [ (   0, player__HOLDER   ),
           ( 0.9, player__FRIEND   ),
           ( 0.1, player__TRAITOR  ),
           (   0, player__FOE      )
         ];

cons chooseInteraction [ loc__TRANSIT ] =
     pud [ ( 1, NOONE            ),
           ( 0, player__HOLDER   ),
           ( 0, player__FRIEND   ),
           ( 0, player__TRAITOR  ),
           ( 0, player__FOE      )
         ];


// Determining where to go next, conditioned by (current) location
// note: loc__TRANSIT is naturally a transient location ... and so NOT a destination.
cons chooseDestination [ loc__HOME ] =
     pud [ (   0, loc__HOME       ),
           ( 1/3, loc__DESK       ),
           ( 1/3, loc__CONF       ),
           ( 1/3, loc__BIZCLIENT  )
         ];

cons chooseDestination [ loc__DESK ] =
     pud [ ( 1/5, loc__HOME       ),
           (   0, loc__DESK       ),
           ( 2/5, loc__CONF       ),
           ( 2/5, loc__BIZCLIENT  )
         ];

cons chooseDestination [ loc__CONF ] =
     pud [ ( 1/5, loc__HOME       ),
           ( 2/5, loc__DESK       ),
           (   0, loc__CONF       ),
           ( 2/5, loc__BIZCLIENT  )
         ];

cons chooseDestination [ loc__BIZCLIENT ] =
     pud [ ( 1/5, loc__HOME       ),
           ( 2/5, loc__DESK       ),
           ( 2/5, loc__CONF       ),
           (   0, loc__BIZCLIENT  )
         ];


// Conditional distribution capturing the amount of time spent at each location.
cons chooseTimeSpentAt [ loc__HOME ]       = (2.5 * hrs) + negexp(5 * hrs);
cons chooseTimeSpentAt [ loc__DESK ]       = (1 * hrs) + negexp(2 * hrs);
cons chooseTimeSpentAt [ loc__CONF ]       = (1 * hrs) + negexp(2 * hrs);
cons chooseTimeSpentAt [ loc__BIZCLIENT ]  = (1 * hrs) + negexp(2 * hrs);
cons chooseTimeSpentAt [ loc__TRANSIT ]    = (15 * mins) + negexp(15 * mins);
```

```
(*=====================================================|
|                 USB STATE VARIABLES                  |
|====================================================*)

var USB_encrypted_items    = 0;             // number of encrypted data items
bin (USB_encrypted_list,     0);            // encrypted data with times of creation
var USB_unencrypted_items  = 0;             // number of unencrypted data items
bin (USB_unencrypted_list,   0);            // unencrypted data with times of creation
var USB_location           = loc__DESK;     // where is the holder now?

var timeForAutoDelete = 1;  // Is it time to make an auto-deletion check yet?
                            // Starts as Yes to get the policy going



(*=====================================================|
|     REWARD / PENALTY and OUTPUT SUMMARY VARIABLES    |
|====================================================*)

var items_created        = 0;  // number of items created
var encrypted_created    = 0;  // number of encrypted items created
var unencrypted_created  = 0;  // number of unencrypted items created

var successful_transfers = 0;  // number of successful interactions
var successful_reads     = 0;  // number of successful reads
var successful_writes    = 0;  // number of successful writes

var failed_transfers     = 0;  // number of failed interactions
var failed_reads         = 0;  // number of failed reads
var failed_writes        = 0;  // number of failed writes

var USB_losses           = 0;  // number of times USB is lost or mislaid
var USB_replacements     = 0;  // number of USB replacements.
var lost_passwords       = 0;  // number of lost passwords
var password_changes     = 0;  // number of changed passwords


(*

    Definition: An EXPOSURE is an opportunity for public release of confidential material.

    A REVEAL is an opportunity for unintended release of confidential material to a friend.

    Opportunities for exposure arises when:-

      1. USB stick is lost and then found by a "foe".
      2. a "traitor"  copies unencrypted confidential info off a USB stick.

    Opportunity for a reveal arises when:-
      1. confidential info is accidentally archived to a friend’s PC.
*)

var exposures                    = 0;  // number of times that traitor/foe gets to read USB stick
var data_amount_exposed          = 0;  // amount of data captured by traitor/foe

var reveals                      = 0;  // number of reveals i.e. accidental archives.
var data_amount_revealed         = 0;  // amount of data revealed  i.e. accidental archives.

var confidentiality_loss         = 0;  // (exposures * data_amount_exposed)
var mean_time_between_exposures  = -1; // average time between exposures

var time_of_first_exposure       = 0;  // DEMOS_TIME of first exposure
var time_of_last_exposure        = 0;  // DEMOS_TIME of last exposure
var embarrassments               = 0;  // number of times a read or add action or an interaction
                                       // fails when visiting a customer
                                       // (USB_location == loc__BIZCLIENT)
var negativeSupportExperiences   = 0;  // number of times the holder needed some form of support
                                       // and didn't get what he needed
var dingsFromManagement          = 0;  // number of times the holder is caught by management
                                       // violating encryption policy
var individualScore              = 0;  // result of computing individual's scoring
                                       // function on the above values and successful actions

var orgUtility                   = 0;  // Value of organization's utility
```

```
(*=====================================================|
|       CLASSES: SCORING FUNCTIONS                      |
|=====================================================*)
class doSupportExperience = {
  local var neg = 0;
  repeat {
    getSV(negSupExp, [], true);
    neg := 0;
    try [testNeedSupport == 1] then {
      try [testSupportCallSucceeds == 0] then {
        negativeSupportExperiences := negativeSupportExperiences + 1;
        neg := 1;
      }
      etry [] then {}
    }
    etry [] then {}
    putSV(negSupExp, [neg]);
  }
}

class monitorPolicyCompliance = {
  local var timeNextDingAllowed = DEMOS_TIME;  // Initial value to ensure the holder
                                               // can get dinged from the start
  try [monitorActive == 1] then {
    repeat {
      hold (monitorPolicyInterval);
      try [USB_unencrypted_items > 0 && DEMOS_TIME > timeNextDingAllowed] then {
        dingsFromManagement := dingsFromManagement + 1;
        timeNextDingAllowed := DEMOS_TIME + monitorMinTimeToNextDing;
      }
      etry [] then {}
    }
  }
  etry [] then {}
}

// results of individual scoring functions
var score_dt      = 0;
var score_e       = 0;
var score_d       = 0;
var score_ns      = 0;

// Individual Scoring Function
class doIndScore = {
  local var trf         = 0;
  local var score       = 0;
  local var duration    = 0;
  repeat {
    getSV(indScore, [], true);
    duration := DEMOS_TIME / years;  // normalize to amount per year
    trf := successful_transfers;  // simple and consistent
    score_dt := dtSF_a * (1 - dtSF_b / (trf/duration + dtSF_b));
    score_e := - eSF_a * embarrassments/duration;
    score_d := dSF_a * (dSF_b / (dingsFromManagement/duration + dSF_b) - 1);
    score_ns := - nsSF_a * (negativeSupportExperiences/duration)
                         * (negativeSupportExperiences/duration);
    score := score_dt + score_e + score_d + score_ns;
    putSV(indScore, [score]);
  }
}

// individual results
var utility_a   = 0;
var utility_c   = 0;
```

```
// Utility function for organization
class doUtility = {
  local var u           = 0;
  local var duration    = 0;
  repeat {
    getSV(utility, [], true);
    duration := DEMOS_TIME / years;  // normalize to amount per year
    utility_a := successful_transfers / duration;  // simple and consistent
    utility_c := (gamma1 * exposures + gamma2 * data_amount_exposed +
                  gamma3 * reveals   + gamma4 * data_amount_revealed) / duration;
    u := alpha * (utility_a - epsilon * utility_c);
    putSV(utility, [u]);
  }
}
```

```
(*==================================================|
|                   CLASSES : USB ACTIONS           |
|==================================================*)
class doUSBadd = {
  local var total = 0;
  local var enc = 0;
  local var neg = 0;

  repeat{
    getSV(USBadd, [], true);

    trace("|> doing USB add data ...");

    total := USB_encrypted_items + USB_unencrypted_items;

    try [total == 0 || doAddNewData == 1] then {

      // choose to add new data onto the USB stick
      items_created := items_created + 1;
      trace(">> ITEM CREATED");

      // is encryption permitted/available for use by me?
      try [allow_encrypted_usage == 1] then {

        enc := chooseToEncryptData;

        try [enc == opn__ENCRYPT_DATA] then {
          syncV(negSupExp, [], [neg]);  // encryption involved - possible need for support
          try [negativeSupportFailsTransaction == 1 && neg == 1] then {
            failed_writes := failed_writes + 1;
            try [USB_location == loc__BIZCLIENT] then {
              // not good to fail in front of the client
              embarrassments := embarrassments + 1;
            }
            etry [] then {}
          }
          etry [holderPasswordStatus == pwd__HAS_PASSWORD] then {
            successful_writes := successful_writes + 1;
            encrypted_created := encrypted_created + 1;
            syncV(USBencryptedWrite, [], []);
          }
          etry [] then {
            // can't encrypt material - so failure

            // Is unencrypted usage permitted when encryption fails?
            try [allow_unencrypted_write_when_encryption_fails == 1] then {
              successful_writes    := successful_writes + 1;  // but it's still successful, I guess
              unencrypted_created  := unencrypted_created + 1;
              syncV(USBunencryptedWrite, [], []);
            }
            etry [] then {
              trace(">> **** FAILED **** WRITE");
              failed_writes := failed_writes + 1;
              try [USB_location == loc__BIZCLIENT] then {
                // not good to fail in front of the client
                embarrassments := embarrassments + 1;
              }
              etry [] then {}
            }

            // lack of password detected - so get new password ...
            syncV(PWDchange, [], []);
          }
        }
        etry [] then {
          // chose to write unencrypted ...
          successful_writes    := successful_writes + 1;
          unencrypted_created  := unencrypted_created + 1;
          syncV(USBunencryptedWrite, [], []);
        }
      }

      etry [] then {
        // encryption unavailable ...
        successful_writes    := successful_writes + 1;
        unencrypted_created  := unencrypted_created + 1;
        syncV(USBunencryptedWrite, [], []);
      }
    }

    etry [] then {
      // Modifying - not creating data ...

      try [allow_encrypted_usage == 1] then {
        enc := chooseToEncryptData;

        try [USB_unencrypted_items == 0 ||
             (USB_encrypted_items > 0 && enc == opn__ENCRYPT_DATA)] then {
          syncV(negSupExp, [], [neg]);  // encryption involved - possible need for support
          try [negativeSupportFailsTransaction == 1 && neg == 1] then {
            failed_writes := failed_writes + 1;
            try [USB_location == loc__BIZCLIENT] then {
              // not good to fail in front of the client
              embarrassments := embarrassments + 1;
            }
            etry [] then {}
          }
          // modifying (existing) encrypted material
          etry [holderPasswordStatus == pwd__HAS_PASSWORD] then {
            hold(decryption_time_costs); // decrypt content ...
            hold(encryption_time_costs); // reencrypt content ...

            successful_writes    := successful_writes + 1;
          }
          etry [] then {
            // can't encrypt/decrypt material - so failure
            //failed_encryptions := failed_encryptions + 1;  // not counted here more

            trace(">> **** FAILED **** ENCRYPTED READ/MODIFY");
            failed_writes := failed_writes + 1;
            try [USB_location == loc__BIZCLIENT] then {
              // not good to fail in front of the client
              embarrassments := embarrassments + 1;
            }
            etry [] then {}

            // lack of password detected - so get new password ...
            syncV(PWDchange, [], []);
          }
        }
        etry [] then {
          // modifying unencrypted material
          // USB_unencrypted_items > 0
          successful_writes  :=  successful_writes + 1;
        }
      }
      etry [] then {
        // modifying unencrypted material
```

```
        // USB_unencrypted_items > 0    -- since total > 0
        successful_writes  :=  successful_writes + 1;
      }
    }
    trace("|> completed USB add data ... ");

    putSV(USBadd, []);
  }
}

class doUSBunencryptedWrite = {
  repeat{
    getSV(USBunencryptedWrite, [], true);

    try [allow_unencrypted_usage == 1] then {
      try [testConfidentialWrite == 1] then {
        USB_unencrypted_items  :=  USB_unencrypted_items + 1;
        putVB (USB_unencrypted_list, [DEMOS_TIME]);
      }
      etry [] then {}
      trace(">> UNENCRYPTED WRITE");

    }
    etry [] then {
      trace(">> **** FAILED **** UNENCRYPTED WRITE");
    }

    putSV(USBunencryptedWrite, []);
  }
}

class doUSBencryptedWrite = {
  repeat{
    getSV(USBencryptedWrite, [], true);

    try [allow_encrypted_usage == 1] then {
      try [testConfidentialWrite == 1] then {
        USB_encrypted_items  := USB_encrypted_items + 1;
        putVB (USB_encrypted_list, [DEMOS_TIME]);
      }
      etry [] then {}
      trace(">> ENCRYPTED WRITE");

      hold(encryption_time_costs);
    }
    etry [] then {
      trace(">> **** FAILED **** ENCRYPTED WRITE");
    }

    putSV(USBencryptedWrite, []);
  }
}

class doUSBread = {

  local var total = 0;                           // total number of items
  local var prob_reading_unencrypted_item = 0;   // probability of reading unencrypted item
  local var neg = 0;

  repeat{
    getSV(USBread, [], true);

    total := USB_encrypted_items + USB_unencrypted_items;
    prob_reading_unencrypted_item := 0;
    try [ total > 0 ] then {

      trace("|> doing USB read data ...");

      prob_reading_unencrypted_item := USB_unencrypted_items/total;

      // toss coin to determine what we do here ...
      try [ binom(1, prob_reading_unencrypted_item) == 1 ] then {
        successful_reads := successful_reads + 1;
      }

      etry [] then {
        try [ USB_encrypted_items > 0 ] then {
          syncV(negSupExp, [], [neg]);  // encryption involved - possible need for support
          try [negativeSupportFailsTransaction == 1 && neg == 1] then {
            failed_reads := failed_reads + 1;
            try [USB_location == loc__BIZCLIENT] then {
              // not good to fail in front of the client
              embarrassments := embarrassments + 1;
            }
            etry [] then {}
          }
          etry [holderPasswordStatus == pwd__HAS_PASSWORD ] then {
            successful_reads := successful_reads + 1;
            hold(decryption_time_costs);
          }
          etry [] then {
            failed_reads := failed_reads + 1;
            try [USB_location == loc__BIZCLIENT] then {
              // not good to fail in front of the client
              embarrassments := embarrassments + 1;
            }
            etry [] then {}

            // read of encrypted material failed due to lack of password ...
            // now get a new password.
            syncV(PWDchange, [], []);
          }
        }
        etry [] then { hold(0); }
      }

      trace("|> completed USB read ...");
    }
    etry [] then { hold(0); }

    putSV(USBread, []);
  }
}

bin (temp_bin, 0);  // temporary repository for use in doUSBdelete
                    // gets filled and emptied twice each time through

class doUSBdelete = {
  local var who    = 0;
  local var total  = 0;
  local var remaining = 0;
  local var i = 0;
  local var selected = 0;
  local var t = 0;

  repeat{
    getSV(USBdelete, [], true);

    total := USB_encrypted_items + USB_unencrypted_items;
    try [ total > 0 ] then {

      trace("|> doing USB delete ... ");

      remaining := rnd(USB_encrypted_items * (1 - propDataDeleted));
      // Randomly select remaining items to keep from USB_encrypted_list
      // Is there a nicer way to do this?
      i := 0;
      selected := 0;
      while [getVB (USB_encrypted_list, [t], true)] {
        try [binom (1, ((remaining-selected)/(USB_encrypted_items-i))) == 1] then {
          putVB (temp_bin, [t]);
          selected := selected + 1;
        }
        etry [] then {}
        i := i + 1;
      }
      do selected {
        getVB (temp_bin, [t], true);
        putVB (USB_encrypted_list, [t]);
      }
      USB_encrypted_items := selected;

      remaining := rnd(USB_unencrypted_items * (1 - propDataDeleted));
      // Randomly select remaining items to keep from USB_unencrypted_list
      // Is there a nicer way to do this?
      i := 0;
      selected := 0;
      while [getVB (USB_unencrypted_list, [t], true)] {
        try [binom (1, ((remaining-selected)/(USB_unencrypted_items-i))) == 1] then {
          putVB (temp_bin, [t]);
          selected := selected + 1;
        }
        etry [] then {}
        i := i + 1;
      }
      do selected {
        getVB (temp_bin, [t], true);
        putVB (USB_unencrypted_list, [t]);
      }
      USB_unencrypted_items := selected;

      trace("|> doing USB delete ... ");

    }
    etry [] then { hold(0); }

    putSV(USBdelete, []);
  }
}

class doUSBwipe = {
  local var who = 0;
  local var total  = 0;
  local var t = 0;

  repeat{
    getSV(USBwipe, [], true);

    total := USB_encrypted_items + USB_unencrypted_items;

    try [ total > 0 ] then {

      trace("|> doing USB wipe ... ");
      USB_encrypted_items   := 0;
      while [getVB (USB_encrypted_list, [t], true)] {  // empty out the list
      }
      USB_unencrypted_items := 0;
      while [getVB (USB_unencrypted_list, [t], true)] {  // empty out the list
      }

      trace("|> exiting USB wipe ... ");
    }
    etry [] then { hold(0); }

    putSV(USBwipe, []);
  }
}
```

```
(*====================================================|
|    CLASSES : USB LOSS, REPLACEMENT and EXPOSURE      |
|====================================================*)
// Accidental archive
class doUSBreveal = {
  local var now    = 0;
  local var amount = 0;

  repeat {
    getSV(USBreveal, [amount], true);

    trace("|> doing accidental archive of USB ... ");

    reveals := reveals + 1;
    data_amount_revealed := data_amount_revealed + amount;

    trace(">> *** REVEAL **** (loc = %v, amount = %v)", USB_location, amount);

    trace("|> completed accidental archive of USB ... ");

    putSV(USBreveal, []);
  }
}

class doUSBexposure = {
  local var now    = 0;
  local var amount = 0;

  repeat {
    getSV(USBexposure, [amount], true);

    try [amount > 0] then {  // only counts as an exposure if amount > 0
      trace("|> doing USB exposure ... ");
      now := DEMOS_TIME;
      try [exposures == 0] then {
        time_of_first_exposure := now;
      }
      etry [] then { hold(0); }
      exposures := exposures + 1;
      data_amount_exposed := data_amount_exposed + amount;
      time_of_last_exposure := now;
      // If an exposure is detected, the holder gets a reprimand from management
      try [testExposureDetected == 1] then {
        dingsFromManagement := dingsFromManagement + 1;
      }
      etry [] then {}
      trace("|> completed USB exposure ... ");
    }
    etry [] then {}

    putSV(USBexposure, []);
  }
}

// Location-indexed probability of foe/customer acquiring USB stick
class doUSBloss = {
  local var probUSBfoundByFoe       = 0;
  local var timeMislaidFor          = 0;

  repeat{
    getSV(USBloss, [], true);

    trace("|> doing USB loss ... ");
    USB_losses := USB_losses + 1;
    probUSBfoundByFoe := prob_foe_findsUSB [USB_location];

    try [binom(1, probUSBfoundByFoe) == 1] then {
      trace(">> *EXPOSURE BY FOE* - USB was lost and then somehow recovered by FOE");
      syncV(USBexposure, [USB_unencrypted_items], []);
    }
    etry [] then { hold(0); }

    timeMislaidFor := mislaidTime;
    try [timeMislaidFor < mislaidTimeLimit] then {
        // can't be productive (at least in this way) until it's found
        hold (timeMislaidFor);
    }
    etry [] then {
      // Give up and report it lost
      hold (mislaidTimeLimit);
      syncV(USBreplace, [], []);
    }

    trace("|> completed USB loss ... ");

    putSV(USBloss, []);
  }
}

class doUSBreplace = {
  local var t = 0;
  repeat{
    // This holds for replacement time penalty
    getSV(USBreplace, [], true);
    trace("|> doing USB replacement ... ");

    USB_encrypted_items   := 0;
    while [getVB (USB_encrypted_list, [t], true)] {  // empty out the list
    }
    USB_unencrypted_items := 0;
    while [getVB (USB_unencrypted_list, [t], true)] {  // empty out the list
    }

    USB_replacements := USB_replacements + 1;

    hold(USB_replacement_time_penalty);

    trace("|> completed USB replacement ... ");
    putSV(USBreplace, []);
  }
}


(*=====================================================|
|               CLASSES : PASSWORD ACTIONS             |
|=====================================================*)
class doLosePassword = {
  local var t = 0;
  repeat{
    getSV(PWDlose, [], true);
    trace("|> doing password lose ... ");

    lost_passwords      := lost_passwords + 1;
    holderPasswordStatus := pwd__NO_PASSWORD;

    hold(password_replacement_time);

    trace("|> completed password lose ... ");
    putSV(PWDlose, []);
  }
}

// Changing the password does not lose the current encrypted content on USB stick.
class doChangePassword = {
  repeat{
    getSV(PWDchange, [], true);
    trace("|> doing password change ... ");

    password_changes    := password_changes + 1;
    holderPasswordStatus := pwd__HAS_PASSWORD;

    trace("|> completed password change ... ");
    putSV(PWDchange, []);
  }
}
```

```
(*======================================================|
|               CLASSES : USB LIFECYCLE                 |
|======================================================*)
class doUSBaction = {
  local var act = 0;

  repeat {
    getSV(USBaction, [act], true);

    // dispatch switch/case:
    try [ act == usb_act__ADD_DATA ] then {
      syncV(USBadd, [], []);
    }
    etry [ act == usb_act__READ_DATA ] then {
      syncV(USBread, [], []);
    }
    etry [ act == usb_act__DEL_DATA ] then {
      syncV(USBdelete, [], []);
    }
    etry [ act == usb_act__WIPE_DATA ] then {
      syncV(USBwipe, [], []);
    }
    etry [] then { trace("USBaction : BAD ACTION CODE %v", act); close; } // impossible ?

    putSV(USBaction, []);
  }
}

class doInteractWithUSB = {
  local var action = 0;
  local var who = 0;
  local var total = 0;                          // total number of items
  local var prob_reading_unencrypted_item = 0;   // probability of reading an unencrypted item
  local var neg = 0;

  repeat{
    getSV(USBinteract, [], true);
    trace("|> doing USB interaction ... ");

    who := chooseInteraction [USB_location];

    // if no TRAITOR's or FOE's enabled, then map to FRIEND
    try [ who == player__TRAITOR ] then {
      try [ enable_TRAITOR_player == 0 ] then { who := player__FRIEND; }
      etry [] then { hold(0); }
    }
    etry [ who == player__FOE ] then {
      try [ enable_FOE_player == 0 ] then { who := player__FRIEND; }
      etry [] then { hold(0); }
    }
    etry [] then { hold(0); }

    // now process the interaction ...
    try [ who <> NOONE ] then {

      total := USB_encrypted_items + USB_unencrypted_items;
      prob_reading_unencrypted_item := 0;

      try [ total > 0 ] then {

        prob_reading_unencrypted_item := USB_unencrypted_items/total;

        try [ who == player__HOLDER ] then {
          trace("USBinteract : Holder can't interact with self!"); close;
```

```
}
// Here both the FRIEND and TRAITOR cases are taken as being very similar - after
// all the TRAITOR player is supposed to be indistinguishable to the FRIEND - and
// so the accounting should treat them broadly the same way.
etry [who == player__FRIEND || who == player__TRAITOR] then {

  try [ binom(1, prob_reading_unencrypted_item) == 1 ] then {
    // unencrypted case
    //successful_reads      := successful_reads + 1;  // not counted here any more
    successful_transfers  := successful_transfers + 1;

    // if player is actually a TRAITOR
    try [who == player__TRAITOR] then {

      try [testExposureHappensByTraitor == 1] then {
        // TRAITOR exposes/extracts unencrypted material only from USB stick ...
        trace(">> *EXPOSURE BY TRAITOR* - read USB");
        syncV(USBexposure, [USB_unencrypted_items], []);
      }
      etry [] then { hold(0); }

    }
    etry [] then {

      // accidental archive of accessable material - i.e. USB_unencrypted_items
      try [testAccidentalArchive == 1] then {
        syncV(USBreveal, [USB_unencrypted_items], []);
      }
      etry [] then { hold(0); }

    }
  }

  etry [] then {
    syncV(negSupExp, [], [neg]);  // encryption involved - possible need for support
    try [negativeSupportFailsTransaction == 1 && neg == 1] then {
      failed_transfers := failed_transfers + 1;
      try [USB_location == loc__BIZCLIENT] then {
        // not good to fail in front of the client
        embarrassments := embarrassments + 1;
      }
      etry [] then {}
    }
    etry [holderPasswordStatus == pwd__HAS_PASSWORD] then {
      // encrypted case
      successful_transfers  := successful_transfers + 1;
      hold(decryption_time_costs);

      // if player is actually a TRAITOR
      try [who == player__TRAITOR] then {

        try [testExposureHappensByTraitor == 1] then {
          // TRAITOR exposes/extracts EVERYTHING on USB stick ...
          trace(">> *EXPOSURE BY TRAITOR* - read USB");
          syncV(USBexposure, [total], []);
        }
        etry [] then { hold(0); }

      }
      etry [] then {

        // accidental archive of accessable material - i.e. EVERYTHING!!!
        try [testAccidentalArchive == 1] then {
          syncV(USBreveal, [total], []);
        }
```

```
                etry [] then { hold(0); }

              }
            }
            etry [] then {
              failed_transfers := failed_transfers + 1;
              try [USB_location == loc__BIZCLIENT] then {
                // not good to fail in front of the client
                embarrassments := embarrassments + 1;
              }
              etry [] then {}
            }
          }
        }
        etry [who == player__FOE] then {
          try [testExposureHappensByFoe == 1] then {
            // FOE exposes/extracts unencrypted material from USB stick ...
            trace(">> *EXPOSURE BY FOE* - read USB");
            syncV(USBexposure, [USB_unencrypted_items], []);
          }
          etry [] then { hold(0); }
        }
        etry [] then {
          trace("USBinteract : BAD PLAYER CODE %v", who);
          close;
        } // impossible ?

      }
      etry [] then { hold(0); }

    }
    etry [] then { hold(0); }

    trace("|> completed USB interaction ... ");
    putSV(USBinteract, []);
  }
}


// Implement auto-deletion of old data from the USB stick
class doUSBautoDelete = {
  local var t = 0;
  local var threshold = 0;
  repeat {
    getSV (USBautoDelete, [], true);
    timeForAutoDelete := 0;
    entity (scheduleAutoDelete, scheduleAutoDelete, autoDeletionPeriod);
    // Delete anything created more than autoDeletionPeriod ago;
    threshold := DEMOS_TIME - autoDeletionPeriod;
    while [getVB (USB_encrypted_list, [t], (t<=threshold))] {
        USB_encrypted_items := USB_encrypted_items - 1;
    }
    while [getVB (USB_unencrypted_list, [t], (t<=threshold))] {
        USB_unencrypted_items := USB_unencrypted_items - 1;
    }

    putSV (USBautoDelete, []);
  }
}


class scheduleAutoDelete = {
    timeForAutoDelete := 1;
}
```

```
(*
   Movement process - this is a genuinely concurrent activity which shows
   where USB actions take place.  We are not modelling the holder per se -
   what we are modelling are the USB relevant actions made on the holder's
   USB stick.

   Thus movement simply happens at some stochastic rate, conditioned upon
   by the current location.  This represents the time spent in a give place
   determines when to move on.  The current location also influences where
   to go next.

   Valid patterns of location change:

       DESK <-> CONF

       DESK <-> TRANSIT <-> BIZCLIENT

       CONF <-> TRANSIT <-> BIZCLIENT

       HOME <-> TRANSIT <-> DESK

       HOME <-> TRANSIT <-> CONF

       HOME <-> TRANSIT <-> BIZCLIENT

*)
class movement = {
  local var waitingDuration = 0;
  local var newDest = USB_location;

  repeat {
    waitingDuration := chooseTimeSpentAt [USB_location];

    hold(waitingDuration);  // this represents the time spent in any given location.

    trace("|> Current location is %v", USB_location);

    // Are we currently in transit?
    try [USB_location <> loc__TRANSIT] then {

      // starting transition - so choose (non-transit) destination stocastically
      newDest := chooseDestination [USB_location];

      try [newDest <> USB_location] then {

        try [ USB_location == loc__CONF ] then {
          try [ newDest == loc__DESK ] then {
            USB_location := newDest;
          }
          etry [] then {
            USB_location := loc__TRANSIT;
          }
        }
        etry [ USB_location == loc__DESK ] then {
          try [ newDest == loc__CONF ] then {
            USB_location := newDest;
          }
          etry [] then {
            USB_location := loc__TRANSIT;
          }
        }
        etry [] then {
          USB_location := loc__TRANSIT;
        }
      }
      etry [] then { hold(0); }

    }
    etry [] then {

      // we are currently in transit
      USB_location := newDest;
    }
  }
}

class lifeUSB = {
  //local var who    = player__HOLDER;
  local var event  = 0;
  local var action = 0;
  local var neg = 0;

  trace("lifeUSB starts ...");

  repeat {

    hold (timePasses);

    //who := player__HOLDER;

    trace("|> lifeUSB (a) : USB_location = %v", USB_location);

    event := chooseEventForHolder [USB_location];

    trace("|> lifeUSB (b) : event = %v", event);

    // dispatch switch/case:
    try [ event == NON_EVENT ] then {
      // nothing happens ...
      hold(0);
    }
    etry [ event == ev__DO_USB_ACT ] then {
      // Apply auto-delete policy
      try [autoDeleteInForce == 1 &&
           USB_location == loc__DESK && timeForAutoDelete == 1] then {
          syncV (USBautoDelete, [], []);
      }
      etry [] then {}
      action := choiceUSBAction;
      syncV(USBaction, [action], []);
    }
    etry [ event == ev__DO_USB_INTERACTION ] then {
      syncV(USBinteract, [], []);
    }
    etry [ event == ev__LOSE_USB ] then {
      syncV(USBloss, [], []);
    }
    etry [ event == ev__LOSE_PASSWORD ] then {
      syncV(PWDlose, [], []);
    }
    etry [ event == ev__CHANGE_PASSWORD ] then {
      syncV(negSupExp, [], [neg]);  // encryption involved - possible need for support
      try [negativeSupportFailsTransaction == 1 && neg == 1] then {
        hold(0);  // We don't record failed password changes
      }
      etry [] then {
        syncV(PWDchange, [], []);
      }
    }
    etry [] then {
      trace("lifeUSB : BAD EVENT CODE %v", event);
      close;
    } // impossible ?
  }
}
```

```
(*====================================================|
|                 MEASUREMENT and ADMIN               |
|====================================================*)

var demos_sample_tick = 0;
var DAY = 0;

class measure = {
  priority(-1);
  repeat {
    trace("-------------------------------------------------------------------");
    trace("day=%v",                         DAY);

    trace("USB_unencrypted_items=%v",      USB_unencrypted_items);
    trace("USB_encrypted_items=%v",        USB_encrypted_items);

    trace("successful_transfers=%v",       successful_transfers);
    trace("successful_writes=%v",          successful_writes);
    trace("successful_reads=%v",           successful_reads);

    trace("exposures=%v",                  exposures);
    trace("data_amount_exposed=%v",        data_amount_exposed);

    trace("reveals=%v",                    reveals);
    trace("data_amount_revealed=%v",       data_amount_revealed);

    trace("items_created=%v",              items_created);
    trace("encrypted_created=%v",          encrypted_created);
    trace("unencrypted_created=%v",        unencrypted_created);

    trace("failed_transfers=%v",           failed_transfers);
    trace("failed_writes=%v",              failed_writes);
    trace("failed_reads=%v",               failed_reads);

    trace("USB_losses=%v",                 USB_losses);
    trace("USB_replacements=%v",           USB_replacements);
    trace("lost_passwords=%v",             lost_passwords);
    trace("password_changes=%v",           password_changes);

    trace("dingsFromManagement=%v",        dingsFromManagement);
    trace("negativeSupportExperiences=%v", negativeSupportExperiences);
    trace("embarrassments=%v",             embarrassments);
    trace("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++");

    hold(measureInterval);

    DAY := DAY + measureInterval/days;

    demos_sample_tick := demos_sample_tick + 1;
  }
}
```

```
(*==================================================|
|                       ENTITIES                    |
|==================================================*)

entity(doSupportExperience,         doSupportExperience,        0);
entity(monitorPolicyCompliance,     monitorPolicyCompliance,    0);
entity(doIndScore,                  doIndScore,                 0);
entity(doUtility,                   doUtility,                  0);

entity(measure,   measure,    0);
entity(USB,       lifeUSB,    0);
entity(movement,  movement,   0);

// sync entities - transactions that happen within the life of the USB
entity(sync_USBAction,             doUSBaction,             0);   // despatch class helper

entity(sync_USBadd,                doUSBadd,                0);
entity(sync_USBread,               doUSBread,               0);
entity(sync_USBdelete,             doUSBdelete,             0);
entity(sync_USBwipe,               doUSBwipe,               0);

entity(sync_InteractWithUSB,       doInteractWithUSB,       0);  // jfg - moved here to match
                                                                 // seeding with original version

entity(sync_USBunencryptedWrite,   doUSBunencryptedWrite,   0);  // unencrypted write helper
entity(sync_USBencryptedWrite,     doUSBencryptedWrite,     0);  // encrypted write helper

entity(sync_USBreveal,             doUSBreveal,             0);  // reveals i.e. accidental archive
                                                                 // of USB to other PC

entity(sync_USBexposure,           doUSBexposure,           0);  // exposure
entity(sync_USBloss,               doUSBloss,               0);
entity(sync_USBreplace,            doUSBreplace,            0);

entity(sync_LosePassword,          doLosePassword,          0);
entity(sync_ChangePassword,        doChangePassword,        0);

entity(sync_USBautoDelete,         doUSBautoDelete,         0);

hold(runTime);  // simulation run time


// Final Reckoning!!
confidentiality_loss := exposures * data_amount_exposed;

try [exposures > 1] then {
  mean_time_between_exposures :=
  (time_of_last_exposure - time_of_first_exposure)  / (exposures - 1);
} etry [] then {
  // There are very few exposures - too few to calculate an average for.
  // so in this case the average time becomes total maximum amount of time available ..
  mean_time_between_exposures := runTime;
}

trace("confidentiality_loss=%v",        confidentiality_loss);
trace("mean_time_between_exposures=%v", mean_time_between_exposures);


//  Compute individual score before the last output
local var indSc   = 0;  // a demos idiosyncracy (sp?)

syncV(indScore, [], [indSc]);
individualScore := indSc;
trace("score_dt=%v",        score_dt);
trace("score_e=%v",         score_e);
trace("score_d=%v",         score_d);
trace("score_ns=%v",        score_ns);
trace("individualScore=%v", individualScore);


//  Compute organization’s utility function
local var orgU    = 0;  // local var because of syncV

syncV(utility, [], [orgU]);
orgUtility := orgU;
trace("utility_a=%v",   utility_a);
trace("utility_c=%v",   utility_c);
trace("orgUtility=%v",  orgUtility);

demos_sample_tick := demos_sample_tick + 1;  // get final output sample ...

close;
```