Australian Government
Department of Defence
Defence Science and
Technology Organisation

A Formal Integrity Framework with Application to a Secure Information ATM (SIATM)

Mark Anderson, Paul Montague and Benjamin Long

Parallax

Defence Science and Technology Organisation

DSTO–TR–2726

ABSTRACT

Information Security is traditionally treated in three main categories: *Confidentiality*, *Integrity*, and *Availability*. While much work has been done on modelling Confidentiality and Availability, aspects involving comprehensive modelling and quality of data integrity in complex systems appear to be, on a relative scale, much less well understood and implemented. Further, most work on Integrity and resultant implementations seems to have focussed more on a matters related to source authentication and transmission assurance. However, the quality of data aspect is becoming more critical for attention, given the increasing levels of automation of information fusion and data transformation in a globalised Cyberspace.

In this paper, we survey the existing integrity models and identify shortcomings of these with regard to a general integrity framework encompassing the quality of data aspect. We then propose and formally model a new framework, illustrating the approach with reference to use cases built around the Secure Information ATM (SIATM) - a highly accreditable security system currently under development.

Published by

DSTO Defence Science and Technology Organisation
PO Box 1500
Edinburgh, South Australia 5111, Australia

Telephone: (08) 7389 5555
Facsimile: (08) 7389 6567

AR No. AR 015-350
October, 2012

APPROVED FOR PUBLIC RELEASE

A Formal Integrity Framework with Application to a Secure Information ATM (SIATM)

Executive Summary

Information Security is traditionally treated in three main categories: *Confidentiality*, *Integrity*, and *Availability*. While much work has been done on modelling Confidentiality and Availability, aspects involving comprehensive modelling and quality of data integrity in complex systems appear to be, on a relative scale, much less well understood and implemented. Further, most work on Integrity and resultant implementations seems to have focussed more on a matters related to source authentication and transmission assurance. However, the quality of data aspect is becoming more critical for attention, given the increasing levels of automation of information fusion and data transformation in a globalised Cyberspace. Without a comprehensive ability to measure integrity systematically, consistently, and within its correct context, military systems may struggle to take full advantage of emerging trends.

Two primary and distinct models have previously been proposed as a foundation for systems to manage and reason about data quality integrity as a part of the information security equation: the *Biba Integrity Model* and the *Clark-Wilson Integrity Model*. In this paper, we first review the Biba and Clark-Wilson integrity models, highlighting the key attributes, limitations and later extensions to the models. The balance of the paper identifies research challenges in addressing integrity and, critically, proposes a new model that captures and supports a broader range of integrity dimensions. Finally, we briefly discuss a use case for the model involving an actual implementation of a security device (Secure Information ATM or SIATM) which is to undergo test deployment in several military systems.

THIS PAGE IS INTENTIONALLY BLANK

Authors

Mark Anderson
Parallax

Mark Anderson holds a Ph.D in Computer Security from Monash University, a Bachelors first class honours in Computer Science from the University of New England, and a separate Bachelor of Science in Physics and Mathematics also from the University of New England. He also holds a Graduate Certificate in Management from Deakin University. He is the principal inventor of a number of information security devices and systems which have been installed by various agencies both in Australia and overseas to protect networks and data. He is currently the DSTO Distinguished Fellow (Cyber).

Paul Montague
Command Control Communication and Intelligence Division

Paul Montague received a BA (Hons) (First Class) in Mathematics in 1987, the Certificate of Advance Study in Mathematics (Distinction) in 1988 and a Ph.D in Mathematical Physics in 1992, all from the University of Cambridge. Following a research career in Mathematical Physics, he transitioned into computer security and developed security solutions for Motorola for almost a decade. He now pursues research in security in DSTO within the Unified Security Systems Discipline of the Information Operations Branch, Command Control Communication and Intelligence Division.

Ben Long
Command Control Communication and Intelligence Division

Benjamin Long was awarded a Ph.D. in Computer Science from the University of Queensland in 2007 and, before that, a B.Sc. (Hons I) in Computer Science from the University of Queensland in 2001. At the time of writing, Ben worked as an Information Security Research Scientist within the Unified Security Systems Discipline of Information Operations Branch, Command Control Communication and Intelligence Division. He now works as a Technical Writer for JBoss Enterprise Data Services within Engineering Content Services, Red Hat Asia Pacific Pty Ltd.

THIS PAGE IS INTENTIONALLY BLANK

Contents

Appendices

THIS PAGE IS INTENTIONALLY BLANK

1 Introduction

Information Security is traditionally treated in three main categories: *Confidentiality*, *Integrity*, and *Availability*. While much work has been done on modelling Confidentiality and Availability (with a publicised focus on Denial of Service), aspects involving comprehensive modelling and quality of data integrity in complex systems which are integrated into the overall security equation appear to be, on a relative scale, much less well understood or implemented. Further, most work on Integrity and resultant implementations seems to have focussed more on matters related to source authentication and transmission assurance, for which there is a significant body of knowledge (see *e.g.* [Menezes, van Oorschot & Vanstone 2001] or [Bishop 2003]). However, given the increasing levels of automation of information fusion and data transformation in a globalised Cyberspace, the quality of data aspect is becoming more critical for attention. Automated data fusion using multiple sources under multiple jurisdictions with differing systems assurance and differing algorithmic implementations, all in a distributed environment, is expected to become the norm. Given this trend, the effect of a contaminated (or even more subtly out of context) data item or stream being fused or transformed into an output stream involving automated intermediate processing stages, and where one or more transformation processes themselves may have differing correctness, context and ownership/control/policy implementations (and where some critical decision is taken on the viewed output), is not at all well understood. Furthermore, the output may itself be viewed by some automated process which undertakes a course of action. Even if the viewing process itself is well implemented, a contaminated intermediate processing stage, or data stream which underwent fusion and transformation, could result in an incorrect course of action.

Without a comprehensive ability to measure integrity systematically, consistently, and within its correct context, military systems may struggle to take full advantage of emerging trends such as heterogeneous Cloud based systems (note that there are a still a number of challenges in the Confidentiality and Availability areas yet to be solved as well as the Integrity aspects). This limited capability causes significantly greater incurred expense in the use of many separate single purpose systems, which in themselves may introduce other errors or low integrity issues. These issues may include, for example, multiple human in the loop failure modes.

Some may argue that work in Safety Critical areas focusses primarily on integrity and this in itself is true. However, it tends to deal with the correctness of function in a single context for a system, and does not focus on multiple contexts of data and processing modules, and trends to promoting a view that a system either has a sufficient level of integrity or not rather than one which presumes multiple contexts which may change dynamically.

Two primary and distinct models have previously been proposed as a foundation for systems to manage and reason about data quality integrity as a part of the information security equation: the *Biba Integrity Model* [Biba 1977] and the *Clark-Wilson Integrity Model* [Clark & Wilson 1987]. In this paper, we first review the Biba and Clark-Wilson integrity models, highlighting the key attributes, limitations and later extensions to the models. The balance of the paper identifies research challenges in addressing integrity and proposes a new model that captures and supports a broader range of integrity dimensions.

Finally, we briefly discuss a use case for the model involving an actual implementation of a security device (Secure Information ATM or SIATM) which is to undergo test deployment in several military systems.

1.1 Biba Integrity Model

The Biba Integrity Model [Biba 1977] stated that the information protection issue has two parts: the first part addresses the proper dissemination of information, while the second part relates to the validity of information, or integrity. Biba went on to propose a model that focussed on providing a measure of integrity for subjects and objects, and the prevention of the invisible introduction of data with lesser integrity within a defined system. Biba defined a mathematical model to describe allowable read/write interactions between pairs of subjects and objects, based on a set of ordered integrity levels. Using the model, he presented three integrity policies, described below.

The *strict integrity policy* is the dual of the Bell-LaPadula confidentiality model [Bell & LaPadula 1973]. Given the function, *lvl*, which provides the integrity level of a subject or object, for any subject s (and $s1$ and $s2$), and object o, the strict integrity policy is as follows.

1. s can read o if and only if $lvl(s) \leq lvl(o)$
2. s can write to o if and only if $lvl(o) \leq lvl(s)$
3. $s1$ can execute $s2$ if and only if $lvl(s2) \leq lvl(s1)$

In summary, the policy allows data to flow from high integrity levels to low integrity levels only (see Figure 1).

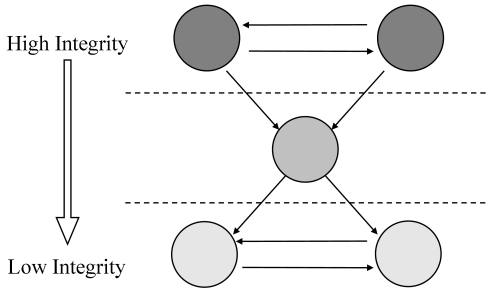

Figure 1: *Data flows towards low integrity.*

When this policy is implemented in conjunction with a confidentiality policy (such as the Bell LaPadula model) data flow is even more restrictive. If Biba and Bell LaPadula levels correspond such that integrity levels increase as confidentiality levels increase, information remains at level. However, in reality, each integrity level may contain data of different confidentiality levels. In this case, the model allows information to flow between levels; however, it flows to lower integrity levels and to higher confidentiality levels (see

Figure 2). Over time, the natural result, is a system populated by information that is highly classified with low integrity.

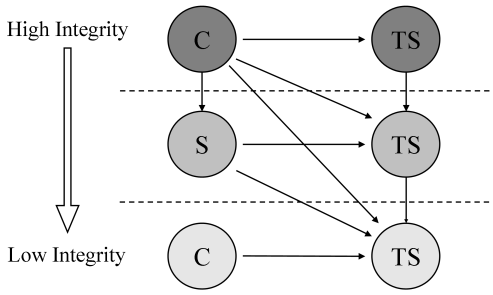

Figure 2: *Data flows towards high classification and low integrity.*

The *low-water mark policy* is similar to the above, only it allows for more interaction between entities by enabling subjects to read objects with a lower integrity level. However, once such a read takes place, the subject's integrity level becomes that of the object.

Like the low-water mark policy, the *ring policy* allows subjects to read objects with a lower integrity level but, in this case, without the need for subjects to be downgraded. After reading from a lower level, the subject can write to its level, thus allowing information to flow to a higher integrity level. However, the condition under which this is appropriate is not specified as part of the model; it is left to the subject to validate observed data. Indeed, Biba introduces a *capability policy* to cater for *trusted processes* that operate outside of the policies allowing data integrity levels to be upgraded.

Biba also introduced a discretionary *protection policy* to allow administrators to specify what particular users can do (read/write/execute) to particular objects via subjects. Again, this is a "dual" of traditional access control lists in discretionary security policies for confidentiality maintenance.

There have been several applications and variations of the Biba model ([Hu & Feng 2008]; [Zhang 2009]; [Lunt et al. 1990]; [Schell & Denning 1986]; [Zun, Tao & Weihua 2009]; [Tang & Qing 2006]), including two that incorporate the Bell-LaPadula confidentiality model ([Zhang, Yun & Zhou 2008]; [Lipner 1982]).

1.2 Clark-Wilson Integrity Model

[Clark & Wilson 1987] expanded the scope of integrity maintenance when compared to the Biba model by including protection against authorised, but improper, modifications for the purpose of preventing and detecting fraud in commercial systems. The Clark-Wilson model proposes two integrity levels based on a distinction between *constrained data items* (CDIs), data that is already part of the system, and *unconstrained data items* (UDIs), new data that is being introduced to the system.

Well formed *transformation procedures* (TPs) and *integrity verification procedures* (IVPs) are introduced as the means of ensuring a system progresses from one valid state

to another, based on one or more inputs (see Figure 3). Typically, TPs operate on a set of CDIs. However, certain certified processes may also operate on UDIs in order to introduce the data to the system, thus labelling them as CDIs.

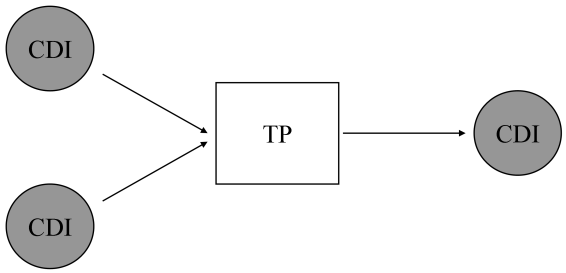

Figure 3: *A well formed TP operating on two CDIs with a combined output.*

The model is described by a set of *enforcement rules* and *certification rules*: those that state what can be maintained by the system, and those that state what must be done by entities outside of the system, respectively.

Separation of duty and general access control is catered for by requirements to restrict what TPs users can execute, and on which particular CDIs. Other security requirements regarding *user authentication* and *audit logging* are also expressed within the rules.

Although the Clark-Wilson model assumes a single high level of integrity applies to CDIs, a model with more explicit rules for managing system data at multiple integrity levels (for instance, Biba's model) may be incorporated within it [Bishop 2003].

There have been several applications and variations of the Clark-Wilson model ([Hanigk 2009]; [Ge, Polack & Laleau 2004]; [Qingguang, Sihan & Yeping 2006]), including one which, as [Bishop 2003] describes, incorporates the Biba model as the integrity policy used [Zhou-Yi, Ye-Ping & Hong-Liang 2010].

1.3 Limitations of the Models

The Clark-Wilson model provides a general framework in which data integrity can be managed; however, it can be interpreted in many ways and needs to be coupled with a model providing stricter guidance for practical data integrity management.

Both models describe the need for human intervention and certified or trusted processes to upgrade or establish data integrity. Little guidance for such processes is provided, thus leaving the models inherently subject to integrity decay.

Although the Clark-Wilson model allows for processes to operate on multiple data sources (unlike Biba's model), neither model considers how the existence of multiple sources of related data can affect their assigned integrity levels. For example, one piece of data may increase or decrease the integrity level of another based on whether the data is confirmed or contradicted, and also on the degree of independence of the data sources. Furthermore, the models do not capture contextual information by which independence is

inferred. There are numerous examples of where this can be critical depending on the mission of the processing taking place and the data incest problem for sensors. As a particular example consider the safety critical case where a plane crashed due to a faulty altitude sensor which provided the same input into two seemingly separate and independent systems (the pilots asked a tower for an altitude check on hearing an alarm, but unfortunately the tower used the same faulty sensor as the plane for its determination [Casey 2006]).

Although Biba's model aims to prevent potential corruption or decrease in assured integrity of data by lower integrity subjects, the model assumes subjects won't corrupt data at (or below) level deliberately or accidentally, whereas the Clark-Wilson model attempts to address this by requiring separation of duty. Here Clark and Wilson attempt integrity maintenance but note that the model implies a single level of integrity (either integrity is maintained or it is not) thus negating opportunities to make decisions based on the notion of *sufficient integrity* for the particular context and circumstances at hand (these may change under different conditions which cannot be encoded in the TP). Even if the Biba model is included along with the notion of separation of duty as per [Bishop 2003], the resultant model still implies a single context for a system when making a comparative assessment between entities thus placing enormous constraints on what would constitute *sufficient integrity*.

Traceability is necessary to provide auditing and analysis, including the ability to make decisions about data integrity based on the history of data. The Clark-Wilson model allows for traceability, whereas there is no discussion in the Biba model.

1.4 Requirements and Challenges for a New Model

The systems for which we want to manage integrity involve data from all manner of sources and different processing requirements across multiple jurisdictions, all with varying contexts and levels of integrity. In these systems we need a way of supporting the automation of decisions regarding the integrity of data and processing as it is sourced, used and maintained. A quantitative measure of integrity (Biba employs a simple quantitative scheme) will capture the notion of such varied levels and enable an implemented system to interpret it for decisions seeking sufficient integrity for the context and circumstances of the moment. In terms of Clark-Wilson, all data in our system will be CDIs tagged with an associated measure of integrity. Ideally, the measurement can be comparable across entities (for example, subjects and objects), while maintaining context and not introducing out of context comparisons

We want our model to be able to increase or decrease data integrity levels based on multiple sources of data. In particular, we are interested in what we call *integrity amplification*, that is, when multiple sources of data are combined to produce data of a higher integrity level. This can provide a method to address the inherent decay trend in existing integrity models and assist to avoid the need for system wide certified upgrade processes or human intervention. This may be achieved by directly providing support for measuring the independence of data sources. To reason about multiple data sources effectively, our model will need to account for the context of the data. Data may have a high level of integrity in one context but a lower level in another.

There are many dimensions to context such as data source, time of creation, method of instantiation, and classification (to name a few) which can all contribute to the context of data; the number of possible contexts is actually infinite and dependent on the particular purpose for which processing is undertaken. Our model will need to provide practical support for these multiple dimensions in order to contribute to a measure of integrity which can be used in the context of the moment.

Like Clark and Wilson, we want our model to provide support for traceability. For example, we must be able to identify when and from where certain data has been introduced, and when, how and by whom operations have been made, and whether operations have been performed by one individual or multiple.

For the reader's convenience, the following table highlights the differences between the two prominent existing models and our desired model.

	Biba	**Clark-Wilson**	**Desired Model**
Quantitative measure	Y	N	Y
Comparable across entities	Y	N	Y
Automated integrity amplification/ attenuation	N	N	Y
Data source independence	N	N	Y
Data context sensitivity	N	N	Y
Traceability	N	Y	Y
Separation of duty	N	Y	Y

Table 1: *A comparison of existing models against requirements.*

2 Overview of Approach

Our approach is rooted in the observation that the integrity of a Data Element is dependent on a **multi-dimensional** Context Vector. In an appropriate coordinatization of the space spanned by such Context Vectors, each axis will represent ideally a well-defined **independent** and **atomic** conceptual context, *e.g.* authentication or reputation. In our model, the integrity of the Data Element is derived from this in a manner which will be described below.

In practise, the number of context dimensions in our "universe" of systems is effectively infinite. An assignment of a meaningful value for all dimensions typically would not make sense, and, naturally, for reasons of practicality of implementation, we do not want to attach to each Data Element an all-encompassing vector which assigns a value to a component in each of these dimensions. Hence the approach adopted is one of specifying a particular integrity model relevant to a system or set of systems, with that model identifying the context dimensions of relevance. Only context values in those dimensions are required to be associated with Data Elements in that model. In addition, as we will describe, the specification of a particular integrity model also describes functions specific to that model which will update the Context Vector of a Data Element in response to actions on that data, and evaluate integrity for that Data Element as a function of its

Context Vector. (Issues to do with the handling of data aggregation, *e.g.* evaluation of integrity across files, will be briefly touched on later. For now, the focus is on the simple atomic data constituents within the model.)

We introduce the concept of a Model List as an ordered collection of models. Specific models in a Model List are to be referenced via a Model Index.

To each Data Element we attach an Integrity Label.

Data	Integrity Label

where an Integrity Label comprises

- a Model Index,
- an Initial Integrity value, and
- a Context Vector.

The Integrity Label is securely bound to the Data Element, though the mechanism of this binding is not relevant at this level of abstraction.

In turn, a Context Vector comprises an ordered set of one or more scalars, where each scalar represents the value of a context dimension. Note that the actual space (*e.g.* continuous vs discrete) may be specified independently for each model and for each dimension.

The Model Index identifies an Integrity Model. This comprises

- a list of the context dimensions which are of relevance to this model,
- a functional specification, *contextUpdate*, of the update of the Context Vector in response to an Event, and
- a functional specification, *evaluateIntegrity*, of the evaluation of the Integrity of the Data Element from the Context Vector and Initial Integrity.

The approach taken is as follows. The Context Vector attached to a Data Element is initialised appropriately for the selected Model Index, *i.e.* it is assigned a value for each of the Context Dimensions identified in the specific model as of relevance. Typically, this will take place either when the Data Element enters the system (across some defined boundary) or is created (special case of entering the system). Subsequently, each Event which occurs within the system and which has an effect on that Data Element will update the Context Vector as per the corresponding function, *updateContext*, specified in the specific model. At any instant, the Integrity of the Data Element may be evaluated from its current Context Vector, again using the corresponding function, *evaluateIntegrity*, from the specific model.

2.1 Transition of Data Elements Between Models

The framework must support transition of Data Elements between models, and also facilitate entry of fresh data into the system. In this paper, we address this problem via use of an Initial Integrity value attached to each Data Element.

When a switch of Integrity Model is made (*i.e.* by updating the Data Element's Model Index), the current Context Vector may be used to evaluate the current integrity value (via the *evaluateIntegrity* method of the initial model). This value may then be stored in the Initial Integrity value of the Data Element in the new model and/or an initial Context Vector is assigned in the new model (for the new set of relevant dimensions). Hence the *evaluateIntegrity* method must take into account the Initial Integrity value as well as the current Context Vector in determining the current integrity of the data. Similarly, we also include the Initial Integrity value as an input to the *contextUpdate* method, since such a dependency cannot in general be excluded. A formal representation of this approach is given in Appendix B.

However, we note here that the authors regard this approach to data transition as not necessarily natural within the given framework. Further research is planned to explore other alternatives. Other approaches than the use of the Initial Integrity value above might include the following.

- The simplest approach is for the Context Vector attached to each Data Element to include *all* of the Context Dimensions relevant to the set of models through which the Data Element passes. This, of course, will typically be impractical in terms of the size of the vector alone. In addition, within a given model, one could not reasonably expect it to actively maintain, in response to events, the set of dimensions which it does not regard as relevant; hence these Context Dimensions would be out of date when the transition to the next model is made.

- A slight variation on the above approach is to accumulate Context Dimensions in the Context Vector attached to the Data Element as it passes through the various models. This mitigates the worst case approach above of handling all Context Dimensions at all times. However, this might lead to a proliferation of models, and, moreover, one would be forced to either, in a given system, cope with a multitude of models (for data arriving from different external models), or adopt a single superset model within the system, growing with every new external model from which data is encountered.

- In another approach, one could introduce some mapping (indexed by the pair of model indices) which takes the final vector from one model and maps it to an initial vector in the new model. This is, in some sense, equivalent to the Initial Integrity value approach adopted currently, in that we need to ensure that the evaluation of the integrity in the original model for the final vector is the same as the evaluation of the integrity in the new model for the initial vector in that model. However, it may be a more natural formulation for many purposes. One issue with this approach though would be that the choice of the initial vector in the new model will typically not be unique. A (typically infinite) set of vectors will evaluate to the correct integrity; with not all of these vectors necessarily being truly equivalent, since this will depend on the specific details of the models. The use of an Initial Integrity value instead avoids having to make what might be a semi-arbitrary choice in the mapping.

- Finally, an alternative approach is to introduce an additional integrity model (in our indexed list) which is the fusion (in some sense to be defined) of the two models involved in the transition, and which acts as a staging post for the transformation of the Data Element and its metadata.

3 Formalised Generic Integrity Model

In this section, we shall present an Object Z formulation of the generic integrity model described in the previous section.

3.1 Data Types

We introduce a generic type describing the data. At the moment, we treat data as simply opaque chunks.

$$[Data]$$

For the context and integrity values, we introduce top level classes which are used in the specification of the generic structure of an integrity model. Particular integrity models will sub-class these in order to accommodate their specific needs, *e.g.* one model may require discrete values (such as Biba - see the Appendices), whilst another may utilise continuous values, such as probability-based models.

ContextValue

$$IntegrityLevelType ::= IntegrityLevelProbType \mid IntegrityLevelBinaryType$$

IntegrityLevel

$$type : IntegrityLevelType$$

We also define a class to represent a User, including a user rating of reputation. Note that this will be dynamically maintained by a mechanism currently out of scope, based on *e.g.* past user actions and the consequences of those actions.

User

$$reputation : \downarrow ContextValue$$

The event types and context dimensions are universal to all models. We shall populate them here with specific values pertinent to our SIATM use case to be explored later. These two definitions will simply grow as more models and instantiations thereof are added to our universe.

$$EventType ::= login \mid logout \mid copyToUSB \mid copyFromUSB \mid printDoc \mid transferDoc \mid downgradeDoc \mid witnessedDowngradeDoc$$

$$ContextDimension ::= sourceAuthentication \mid sourceAuthorisation \mid sourceReputation \mid tamperResistance \mid reliability \mid time \mid completeness \mid storage \mid transport \mid generic$$

For future reference when we discuss the SIATM in a later section, *sourceAuthentication* will represent the degree of confidence in the authentication of the source (perhaps a user) of a Data Element. It will be affected by both the number and quality of authentication factors utilised. *sourceReputation* will represent the degree of confidence in the source's reputation, based upon their history of actions. *sourceAuthorisation* will represent the degree of confidence that the authorisation of the user's access to the DataElement was correctly verified.

3.2 Event

The Event object captures the time-stamped type of an event, the associated user and the associated context. For example, for ATM-related operations, the context values in the associated context vector will represent features specific to that ATM and its environment. Note that this is a generic sequence of undefined length, whereas the context vectors associated with individual Data Elements are of fixed length corresponding to a specified Integrity Model (see below). Hence the Event may capture more information than is required for a given model in which it is being used.

As part of this construct, we introduce a specific *ContextVector* type to capture a set of mappings from Context Dimensions to Context Values (specifically, the relevant sub-class of Context Value pertinent to the specific model in question).

$$ContextVector == ContextDimension \pfun \downarrow ContextValue$$

Event

$type : EventType;$
$timestamp : \mathbb{N};$
$user : User;$
$contextVector : ContextVector$

In addition, we define an event class specifically for logging of events in an audit trail. In our ATM specific example later, this will be used in the ATM's internal audit log.

LogEvent

$type : EventType;$
$timestamp : \mathbb{N};$
$user : User$

3.3 Integrity Model

The integrity model specifies the relevant Context Dimensions for that model, the relevant Context Dimensions for a given Event Type within the model, how a corresponding context vector is to be updated according to a given event, and how the integrity of a tagged Data Element is to be computed from the Context Vector and Initial Integrity of that Data Element.

IntegrityModel

$$
\begin{array}{|l}
\hline
unitVector : ContextVector \\
\hline
relevantDimensions : \mathbb{F}\ ContextDimension; \\
relevantEventDimensions : EventType \leftrightarrow ContextDimension; \\
contextUpdate : (\downarrow IntegrityLevel \times ContextVector \times Event) \nrightarrow ContextVector; \\
evaluateIntegrity : (\downarrow IntegrityLevel \times ContextVector) \nrightarrow \downarrow IntegrityLevel \\
\hline
\mathrm{dom}\ unitVector = relevantDimensions \\
\mathrm{ran}\ relevantEventDimensions \subseteq relevantDimensions \\
\mathrm{dom}\ contextUpdate = \{l : \downarrow IntegrityLevel;\ v : ContextVector;\ e : Event \mid \\
\quad \mathrm{dom}\ v = relevantDimensions \land \\
\quad e.type \in \mathrm{dom}\ relevantEventDimensions \land \\
\quad \mathrm{ran}(\{e.type\} \lhd relevantEventDimensions) = \mathrm{dom}\ e.contextVector\} \\
\mathrm{ran}(\mathrm{dom}\ evaluateIntegrity) = \{v : ContextVector \mid \mathrm{dom}\ v = relevantDimensions\} \\
\hline
\end{array}
$$

The idea is that each Data Element (see below) will carry a context vector recording its context for each of the relevant context dimensions for the model. However, each event is only required to carry context information about the set of dimensions relevant to that **event** in the model. This is for reasons of economy - we do not want to require every event to be forced to carry information unrelated to that event's operation. It also means that we may more easily build up a model containing multiple events event-by-event, since we will not need to retrofit new dimensions to the schemas representing existing events when we add a new event bringing new dimensions to the model.

On that last point, the intent is that the relevant dimensions for the model as a whole will be as small as possible, for the reasons espoused in the introduction to this section. Hence, typically we would expect the relevant context dimensions for the model as a whole to be the union of the dimensions relevant to each of the event types associated with the model.

Note that the *unitVector* is a Context Vector (covering the relevant Context Dimensions of the model) which is used to initialise the Context Vector for new Data Elements. Typically, in a probabilistic type model, it is the vector $(1, 1, \ldots, 1)$.

3.4 File

We introduce the concept of a File as a sequence of Data Elements, and it is the Data Element which is integrity-decorated. The granularity at which this breakdown occurs is

not relevant at this abstract level. We note though that when reasoning about the integrity of a file, it may be that the entire file needs to be treated itself as a subject of integrity considerations. Concepts such as Completeness and Consistency only really make sense at this level, rather than at the level of a Data Element. The introduction of the relevant hierarchical relations amongst Data Elements within a given file are left to future work.

For each such Data Element, an initial Context Vector is assigned on creation. This Context Vector may then be updated, using the function specified by the model, as each Event occurs to the data. The overall Integrity is a function of this Context Vector and an Initial Integrity value, computed using the function specified by the model. As described above, these operations of context update and integrity evaluation are model specific.

Note that the model referenced by the Data Element is a sub-class of the Integrity-Model. We shall later define sub-classes representing specific integrity models. We also stress here that though we referred earlier to each Data Element carrying an index into a list of integrity models, and this is how an implementation would be realised, for the purposes of the abstract model, the Data Element actually includes the integrity model itself.

DataElement

$$
\begin{array}{l}
data : Data; \\
context : ContextVector; \\
initialIntegrity : \downarrow IntegrityLevel; \\
integrity : \downarrow IntegrityLevel; \\
model : \downarrow IntegrityModel; \\
contextUpdate : Event \nrightarrow ContextVector \\
\hline
\mathrm{dom}\ context = model.relevantDimensions \\
\mathrm{dom}\ contextUpdate = \mathrm{ran}(\mathrm{dom}\ model.contextUpdate) \\
\forall\, e : \mathrm{dom}\ contextUpdate \bullet contextUpdate(e) = model.contextUpdate(initialIntegrity, context, e)
\end{array}
$$

getIntegrity

$$
\begin{array}{l}
integrity! : \downarrow IntegrityLevel \\
\hline
integrity! = model.evaluateIntegrity(initialIntegrity, context)
\end{array}
$$

We note that *contextUpdate* does not actually update the context vector of the Data Element. It simply outputs what would be the updated context vector should the operation occur. For example, in the Biba Strict Integrity policy, integrity values do not change. One may regard the model as stating that operations are forbidden should the integrity value change were the operation to be allowed. See Appendix A for more discussion.

We finally define the file, with a function to evaluate the integrity of a specified Data Element in a given file.

File

$$
\begin{array}{l}
fileData : \mathrm{seq}\ DataElement
\end{array}
$$

$$evaluateIntegrity \mathrel{\widehat=} [\, d? : \operatorname{ran} fileData \,] \bullet d?.getIntegrity$$

getId

$$file! : File$$

$$file! = self$$

4 Probabilistic Model

As a specific example, and for later use, we define here an instance of the generic model using explicit functions for *contextUpdate* and *evaluateIntegrity*, where context values are interpreted probabilistically, *i.e.* the Context Value represents the level of confidence appropriate to that particular Context Dimension for the Data Element in question.

To that end, we explicitly define the appropriate sub-classes of the ContextValue and IntegrityLevel spaces.

ContextValueProb

ContextValue

$$value : \mathbb{R}$$

$$value \geq 0 \land value \leq 1$$

The integrity level is a (non-negative) real, which will ultimately be defined in our example model as the length of the context vector, once we have assigned a metric to that context space.

IntegrityLevelProb

IntegrityLevel

$$value : \mathbb{R}$$

$$type = IntegrityLevelProbType$$

We need a utility function which performs component-wise multiplication of context vectors. Note that this function ignores context dimensions not in common between the two operands.

$$multiply : ContextVector \times ContextVector \to ContextVector$$

$$\begin{aligned}&\forall\, a : ContextVector;\ b : ContextVector \bullet \\ &\quad multiply(a, b) = \{ d : ContextDimension;\ v : ContextValueProb \mid \\ &\quad \exists\, p, q : ContextValueProb \bullet (d, p) \in a \land (d, q) \in b \land v.value = p.value * q.value\}\end{aligned}$$

We also define another utility function which sums a given real-valued function over a specified set.

$$[S]$$

$$setSum : (S \to \mathbb{R}) \to \mathbb{R}$$

$$\begin{aligned}
&\forall f : \operatorname{dom} setSum \bullet \\
&\quad f = \varnothing \Rightarrow setSum(f) = 0 \land \\
&\quad f \neq \varnothing \Rightarrow (\forall p : S;\ q : \mathbb{R} \mid (p, q) \in f \bullet setSum(f) = q + setSum(f \setminus \{(p, q)\}))
\end{aligned}$$

We can then define our model.

Context update, because of the simple probabilistic interpretation, is simply (context dimension) coordinate-wise composition of (independent) probabilities by multiplication. The initial context vector will typically start as the sum of the individual orthogonal unit basis vectors (*i.e.* $(1, 1, \ldots, 1)$). Then our update mechanism reflects the fact that we interpret each component as a confidence level in the validity of that particular aspect of context, and we start with full confidence, for argument's sake.

Note that since some of the dimensions relevant to the model are not relevant to a specific event, the confidence values in those dimensions are not affected by the update. For example, the context value corresponding to the context dimension measuring the confidence in the authentication of a user may be largely irrelevant to the integrity of the file in an operation such as file transfer (with no opportunity for editing of the data). However, establishment of the user's identity is clearly critical for integrity of the data in other use cases, as well as of course for other security factors, such as confidentiality of the file.

The net effect of this simple scheme is a gradual decay in confidence (and hence integrity). We shall address this issue later when we discuss fusion/amplification (see Appendix C).

In order to evaluate the integrity, we introduce a "metric" g on the space of context vectors. This will measure the contribution to integrity of the corresponding context dimensions. In terms of the probabilistic interpretation given to the coordinates, some other measure probably makes more sense, *e.g.* the simple product of the independent context dimensions' probabilities. Hence we stress that this particular choice of a "Euclidean-distance" metric is just an example - specific instantiations of the framework as a particular model will need to make a choice of function which is of value in their particular scenario.

Typically, we would expect the matrix g to be diagonal, but in general it will be a symmetric non-negative definite matrix[1], which allows for the fact that our coordinate choice may not be orthogonal, *i.e.* the context dimensions may not be independent. In such cases, armed with the "dot product" (inner product) of vectors defined by this metric, one may then go through a standard procedure to identify an orthogonal basis (and diagonalise the metric in those new coordinates).

Essentially, the integrity is the "length" of the Context Vector v, *i.e.* $\sqrt{(\sum_{i,j} v_i v_j g_{ij})}$. The use of this "metric" also allows us to weight contributions to the overall integrity from

[1]We can force it to be positive definite by simply, in the first place, selecting context dimensions which have some non-zero impact on the overall integrity, *i.e.* choose our set of Relevant Dimensions for the model appropriately.

different dimensions according to their relevance in that particular integrity model, *i.e.* in the context in which integrity is being evaluated.

We need one more utility function:

$$sqrt : \mathbb{R} \to \mathbb{R}$$

$$\forall r : \mathbb{R} \mid r \geq 0 \bullet \exists i : \mathbb{R} \mid i \geq 0 \bullet i * i = r \land sqrt(r) = i$$

ProbabilityModel

IntegrityModel

$$unitVector : ContextDimension \pfun ContextValueProb$$

$$\forall v : \operatorname{ran} unitVector \bullet v.value = 1$$

$$g : (relevantDimensions \times relevantDimensions) \to \mathbb{R}$$

$$\operatorname{ran}(\operatorname{dom}(\operatorname{dom} contextUpdate)) \subseteq ContextDimension \pfun ContextValueProb$$

$$evaluateIntegrity \in IntegrityLevelProb \times (ContextDimension \pfun ContextValueProb) \pfun IntegrityLevelProb$$

$$\forall i : IntegrityLevelProb;\ v : ContextDimension \pfun ContextValueProb \mid (i, v) \in \operatorname{dom} evaluateIntegrity \bullet (\exists integrityContribution : ((relevantDimensions \times relevantDimensions) \to \mathbb{R}) \bullet (\forall d, e : relevantDimensions \bullet \exists p, q : ContextValueProb \bullet v(d) = p \land v(e) = q \land (integrityContribution(d, e) = g(d, e) * p.value * q.value)) \land (\exists j : IntegrityLevelProb \bullet (j = evaluateIntegrity(i, v) \land j.value \geq 0 \land j.value = i.value * sqrt(setSum[relevantDimensions \times relevantDimensions](integrityContribution)))))$$

$$\forall l : IntegrityLevelProb;\ v : ContextVector;\ e : Event \mid (l, v, e) \in \operatorname{dom} contextUpdate \bullet \exists f : ContextVector \bullet contextUpdate(l, v, e) = multiply(v, f) \land f = unitVector \oplus e.contextVector$$

The extension of the context vector of the event e above to a temporary vector f is a bit of a subtlety. We need to ensure that, when multiply is invoked successively on the initial context vector attached to a piece of data, that we maintain context values for each of the relevant event dimensions in the model. Since the event's context vector is not required to have entries for all of the model's relevant dimensions, as discussed above, we risk losing some information. Hence, an extension vector with the multiplicative identity in each of the extended components, is introduced.

5 Application to SIATM Use Cases

The Secure Information ATM (SIATM) is an ATM-style machine currently under development. The underlying concept is that it handles transactions involving information,

as opposed to currency. It will be deployed as the gateway for information transactions to/from a classified network or networks, and provides a strong audit trail linking such transactions to (strongly) identified users in a highly accreditable solution. There are a number of possible use cases which may be supported. Here we shall focus on a few of the simpler ones, including USB transfer of content between SIATMs, printing of RFID-tagged documents and downgrade of documents between classification levels. With regards to RFID-tagging of documents, this allows for the automation of an electronic Classified Document Register (CDR), adding printed documents automatically to the user's CDR, and transferring documents between users' CDR's with ease via a suitable transaction at the SIATM involving both users. The focus is on security, though combined with ease of use.

We model the SIATM at an idealised abstract level as containing an audit log of critical operations (modelled as a sequence of events) and a set of files. It may have at most one USB memory stick inserted, and it may have at most one user logged in at any given time. In addition, we model a CDR documenting ownership of the set of files. For simplicity of exposition, it simply tracks current ownership status rather than a history of files owned for each user.

We shall use the probabilistic model described in the previous section. Note that these probabilities are to be interpreted as conditional probabilities given the assumption that an appropriate signature check (a trivial integrity guard) on the file passed correctly, *i.e.* we focus on the broader aspects of integrity which are the focus of this paper and assume simplistic unauthorised data modification is prevented.

As a general point in this analysis, we note that the reliability and tamper resistance of the SIATM are critical - any malicious software in the device can clearly modify the underlying file data.

Regarding document printing, we introduce the notion of hard copies of documents in the system, as well as a CDR. Note that the CDR tracks ownership not only of these hard copies, but also of electronic files stored on USB memory sticks (the only medium we have in our model).

Obviously, we need to introduce a specific Integrity Model. To that end, we utilise and extend the previously introduced probabilistic model. The SIATM-specific tailoring of the generic probabilistic model simply involves specification of the exact set of relevant dimensions and the relevant dimensions for each event type.

ModelOne

ProbabilityModel

$$
\begin{aligned}
&relevantDimensions = \{sourceAuthentication, \\
&\quad sourceAuthorisation, tamperResistance, reliability, time, \\
&\quad completeness, storage, transport, sourceReputation\} \\
&\mathrm{dom}\ relevantEventDimensions = \{copyToUSB, downgradeDoc, printDoc, \\
&\quad transferDoc, copyFromUSB\} \\
&\mathrm{ran}(\{copyToUSB\} \lhd relevantEventDimensions) = \\
&\quad \{sourceAuthentication, sourceAuthorisation, tamperResistance, \\
&\quad reliability, time, completeness, storage, transport\} \\
&\mathrm{ran}(\{downgradeDoc\} \lhd relevantEventDimensions) = \\
&\quad \{sourceAuthentication, sourceAuthorisation, tamperResistance, \\
&\quad reliability, time, completeness, storage, transport, \\
&\quad sourceReputation\} \\
&\mathrm{ran}(\{printDoc\} \lhd relevantEventDimensions) = \\
&\quad \{sourceAuthentication, sourceAuthorisation, tamperResistance, \\
&\quad reliability, time, completeness, storage, transport\} \\
&\mathrm{ran}(\{transferDoc\} \lhd relevantEventDimensions) = \\
&\quad \{sourceAuthentication, sourceAuthorisation, tamperResistance, \\
&\quad reliability, time, completeness, \\
&\quad sourceReputation\} \\
&\mathrm{ran}(\{witnessedDowngradeDoc\} \lhd relevantEventDimensions) = \\
&\quad \{sourceAuthentication, sourceAuthorisation, tamperResistance, \\
&\quad reliability, time, completeness, storage, transport, \\
&\quad sourceReputation\} \\
&\mathrm{ran}(\{copyFromUSB\} \lhd relevantEventDimensions) = \\
&\quad \{sourceAuthentication, sourceAuthorisation, tamperResistance, \\
&\quad reliability, time, completeness, storage, transport\}
\end{aligned}
$$

The USB stick is introduced. It is simply a container for files.

USB

$$files : \mathbb{F}\ File$$

readFile

$$
\begin{aligned}
&\Delta(files) \\
&file? : File \\
&\hline
&files' = files \cup \{file?\}
\end{aligned}
$$

writeFile

$$file! : files$$

The ATM is a container of files (with designated access by users) and a log, and may have a user logged in and/or a USB memory stick inserted. Various hard-coded parameters

indicate its reliability in various context dimensions. For example, *userAuthenticationValue* indicates the probability for the system of correct authentication of a user using a biometric factor in addition to an RFID card swipe. For simplicity in this example model, this is assumed to be constant across all users.

ATM

$$
\begin{array}{l}
files : \mathbb{F}\ File; \\
log : \text{seq}\ LogEvent; \\
user : \mathbb{F}\ User; \\
timeATM : \mathbb{N}; \\
usb : \mathbb{F}\ USB; \\
allowedUsers : \mathbb{F}\ User; \\
fileAccess : File \leftrightarrow User; \\
reliabilityValue : ContextValueProb; \\
tamperResistanceValue : ContextValueProb; \\
userIdentificationValue : ContextValueProb; \\
\qquad\qquad [\text{user does simple RFID swipe}] \\
userAuthenticationValue : ContextValueProb; \\
\qquad\qquad [\text{biometric authentication}] \\
userAuthorisationValue : ContextValueProb; \\
storageReliabilityValue : ContextValueProb \\
\hline
(\#user) \leq 1 \\
user \subseteq allowedUsers \\
(\#usb) \leq 1
\end{array}
$$

logEvent

$$
\begin{array}{l}
\Delta(log) \\
e? : LogEvent \\
\hline
log' = log \frown \langle e? \rangle
\end{array}
$$

tick

$$
\begin{array}{l}
\Delta(timeATM) \\
\hline
timeATM' = timeATM + 1
\end{array}
$$

login

$$
\begin{array}{l}
\Delta(user) \\
user? : User; \\
e! : LogEvent \\
\hline
user = \varnothing \\
user? \in allowedUsers \\
e!.type = login \\
e!.timestamp = timeATM \land e!.user = user? \\
user' = \{user?\}
\end{array}
$$

$loginATM \mathrel{\widehat=} login \parallel logEvent$

insertUSB

$\Delta(usb)$
$usb? : USB$

$user \neq \varnothing$
$usb = \varnothing$
$usb' = \{usb?\}$

removeUSB

$\Delta(usb)$

$usb \neq \varnothing \land usb' = \varnothing$
$user \neq \varnothing$

writeFile

[*writeFile* is a utility method utilised]
[by a few high level use cases,]
[*e.g.* for printing and also transferring to USB.]
[It outputs a copy of the file with an updated]
[context vector, as well as a log entry for the event]

$file? : files;$
$file! : File;$
$user? : User;$
$eventType? : EventType;$
$e! : LogEvent$

$user = \{user?\}$
$file? \mapsto user? \in fileAccess$
$e!.type = eventType? \land e!.timestamp = timeATM$
$e!.user \in user$
$\exists\, m : Event \bullet$
$\quad \operatorname{dom} file!.fileData = \operatorname{dom} file?.fileData \land$
$\quad (\forall\, n : \operatorname{dom} file!.fileData \bullet$
$\quad\quad (file!.fileData(n)).data = (file?.fileData(n)).data \land$
$\quad\quad (file!.fileData(n)).context = (file?.fileData(n)).contextUpdate(m) \land$
$\quad\quad (file!.fileData(n)).initialIntegrity = (file?.fileData(n)).initialIntegrity \land$
$\quad\quad (file!.fileData(n)).model = (file?.fileData(n)).model) \land$
$\quad m.type = eventType? \land$
$\quad m.timestamp = timeATM \land m.user \in user \land$
$\quad m.contextVector =$
$\quad\quad \{sourceAuthentication \mapsto userIdentificationValue,$
[no fingerprint check, so RFID user id confidence only]
$\quad\quad sourceAuthorisation \mapsto userAuthorisationValue,$
$\quad\quad tamperResistance \mapsto tamperResistanceValue,$
$\quad\quad reliability \mapsto reliabilityValue,$
$\quad\quad time \mapsto reliabilityValue,$
$\quad\quad completeness \mapsto reliabilityValue,$
$\quad\quad storage \mapsto storageReliabilityValue,$
$\quad\quad transport \mapsto reliabilityValue\}$

[The ATM version of each method logs the event to the SIATM's log]
$writeFileATM \mathrel{\widehat=} writeFile \parallel logEvent$

readFile

[Similar utility function. Updates ATM's file set]
[with copy of given file, after updating context vector.]

$\Delta(files, fileAccess)$
$file? : File$;
$user? : User$;
$eventType? : EventType$;
$e! : LogEvent$

$user = \{user?\}$
$e!.type = eventType? \land e!.timestamp = timeATM$
$e!.user \in user$
$\exists\, m : Event;\ f : File \bullet$
$\quad \operatorname{dom} f.fileData = \operatorname{dom} file?.fileData \land$
$\quad (\forall\, n : \operatorname{dom} f.fileData \bullet$
$\quad\quad (f.fileData(n)).data = (file?.fileData(n)).data \land$
$\quad\quad (f.fileData(n)).context = (file?.fileData(n)).contextUpdate(m) \land$
$\quad\quad (f.fileData(n)).initialIntegrity = (file?.fileData(n)).initialIntegrity \land$
$\quad\quad (f.fileData(n)).model = (file?.fileData(n)).model) \land$
$\quad fileAccess' = fileAccess \cup \{f \mapsto user?\} \land$
$\quad files' = files \cup \{f\} \land$
$\quad m.type = eventType? \land$
$\quad m.timestamp = timeATM \land m.user \in user \land$
$\quad m.contextVector =$
$\quad\quad \{sourceAuthentication \mapsto userIdentificationValue,$
[no fingerprint check]
$\quad\quad sourceAuthorisation \mapsto userAuthorisationValue,$
$\quad\quad tamperResistance \mapsto tamperResistanceValue,$
$\quad\quad reliability \mapsto reliabilityValue,$
$\quad\quad time \mapsto reliabilityValue,$
$\quad\quad completeness \mapsto reliabilityValue,$
$\quad\quad storage \mapsto storageReliabilityValue,$
$\quad\quad transport \mapsto reliabilityValue\}$

$readFileATM \mathrel{\widehat=} readFile \parallel logEvent$

transferDoc

[CDR transfer functionality.]
[Acts on what will be printed documents (not files in ATM fileset).]
[Updates context vector of printed document.]

[Idea is that this metadata will be attached via an electronic record or in RFID.]

$\Delta(files)$
$file? : File;$
$file! : File;$
$source? : User;$
$destination? : User;$
$e! : LogEvent$

$user = \{source?\}$
$destination? \in allowedUsers$
$e!.type = transferDoc \land e!.timestamp = timeATM$
$e!.user \in user$
$\exists\, m : Event \bullet$
$\quad \operatorname{dom} file!.fileData = \operatorname{dom} file?.fileData \land$
$\quad (\forall\, n : \operatorname{dom} file!.fileData \bullet$
$\quad\quad (file!.fileData(n)).data = (file?.fileData(n)).data \land$
$\quad\quad (file!.fileData(n)).context = (file?.fileData(n)).contextUpdate(m) \land$
$\quad\quad (file!.fileData(n)).initialIntegrity = (file?.fileData(n)).initialIntegrity \land$
$\quad\quad (file!.fileData(n)).model = (file?.fileData(n)).model) \land$
$\quad m.type = transferDoc \land$
$\quad m.timestamp = timeATM \land m.user \in user \land$
$\quad m.contextVector =$
$\quad\quad \{sourceAuthentication \mapsto userAuthenticationValue,$
$\quad\quad sourceAuthorisation \mapsto userAuthorisationValue,$
$\quad\quad tamperResistance \mapsto tamperResistanceValue,$
[Electronic storage of file history]
$\quad\quad reliability \mapsto reliabilityValue,$
$\quad\quad time \mapsto reliabilityValue,$
$\quad\quad completeness \mapsto source?.reputation,$
[Complete set of pages depends on user trustworthiness]
$\quad\quad sourceReputation \mapsto source?.reputation\}$

$transferDocATM \mathrel{\widehat=} transferDoc \parallel logEvent$

$copyToUSBATM \mathrel{\widehat=} [\, u? : usb \,] \bullet ([\, eventType? : EventType \mid$
$\quad eventType? = copyToUSB \,] \land writeFileATM) \parallel u?.readFile$

$copyFromUSBATM \mathrel{\widehat=} [\, u? : usb \mid u? \in usb \,] \bullet u?.writeFile \parallel ([\, eventType? : EventType \mid$
$\quad eventType? = copyFromUSB \,] \land readFileATM)$

$printDocATM \mathrel{\widehat=} [\, eventType? : EventType \mid$
$\quad eventType? = printDoc \,] \land writeFileATM$

downgradeDocument

[This and next op are front ends for downgrading]
[This one with single user and next op is witnessed]

$downgradeType! : EventType;$
$sourceAuthenticationValue! : ContextValueProb$

$downgradeType! = downgradeDoc$
[User authenticates operation via fingerprint]
$sourceAuthenticationValue! = userAuthenticationValue$

witnessedDowngradeDocument

$downgradeType! : EventType;$
$sourceAuthenticationValue! : ContextValueProb$

$downgradeType! = witnessedDowngradeDoc$
[Both users authenticated operation via fingerprints]
$sourceAuthenticationValue! = userAuthenticationValue * userAuthenticationValue$

downgradeDocumentCommon

$\Delta(files)$
$downgradeType? : EventType;$
$sourceAuthenticationValue? : ContextValueProb;$
$file? : File;$
[User picks data elements to redact]
$userSelection? : \mathbb{F}\,\mathbb{N};$
$e! : LogEvent$

$userSelection? \subseteq 1 \,..\, \#file?.fileData$
$user \neq \varnothing$
$e!.type = downgradeType? \land e!.timestamp = timeATM$
$e!.user \in user$
$\exists\, u : user \bullet (file?, u) \in fileAccess \land$
$\quad (\exists f : File;\ m : Event;\ filteredData : \operatorname{seq} DataElement \mid$
$\quad filteredData = userSelection? \upharpoonright file?.fileData \bullet$
$\quad \operatorname{dom} f.fileData = \operatorname{dom} filteredData \land$
$\quad (\forall n : \operatorname{dom} f.fileData \bullet$
$\quad\quad (f.fileData(n)).data = (filteredData(n)).data \land$
$\quad\quad (f.fileData(n)).context = (filteredData(n)).contextUpdate(m) \land$
$\quad\quad (f.fileData(n)).initialIntegrity = (filteredData(n)).initialIntegrity \land$
$\quad\quad (f.fileData(n)).model = (filteredData(n)).model) \land$
$\quad m.type = downgradeType? \land$
$\quad m.timestamp = timeATM \land m.user \in user \land$
$\quad files' = files \cup \{f\} \land$
$\quad m.contextVector =$
$\quad\quad \{sourceAuthentication \mapsto sourceAuthenticationValue?,$
$\quad\quad sourceAuthorisation \mapsto userAuthorisationValue,$
$\quad\quad tamperResistance \mapsto tamperResistanceValue,$
$\quad\quad reliability \mapsto reliabilityValue,$
$\quad\quad time \mapsto reliabilityValue,$
$\quad\quad completeness \mapsto reliabilityValue,$
$\quad\quad storage \mapsto storageReliabilityValue,$
$\quad\quad transport \mapsto reliabilityValue,$
$\quad\quad sourceReputation \mapsto u.reputation\})$

$downgradeDocATM \mathrel{\widehat=} downgradeDocument \parallel downgradeDocumentCommon \parallel logEvent$
$witnessedDowngradeDocATM \mathrel{\widehat=} witnessedDowngradeDocument \parallel$
$\quad downgradeDocumentCommon \parallel logEvent$

logout

$\Delta(user)$
$e! : LogEvent$

$usb = \varnothing$
$user \neq \varnothing \land user' = \varnothing$
$e!.type = logout \land e!.timestamp = timeATM$
$e!.user \in user$

$logoutATM \mathrel{\widehat=} logout \parallel logEvent$

CDR

$possession : File \pfun User$

add

$file? : File;$
$user? : User$

$file? \notin \mathrm{dom}\, possession$
$possession' = possession \cup \{file? \mapsto user?\}$

transfer

$file? : File;$
$source?, destination? : User$

$file? \mapsto source? \in possession$
$possession' = possession \oplus \{file? \mapsto destination?\}$

remove

$file? : File$

$file? \in \mathrm{dom}\, possession$
$possession' = \{file?\} \rhd possession$

System

$$
\begin{array}{l}
atms : \mathbb{F}\, ATM; \\
usbs : \mathbb{F}\, USB; \\
cdr : CDR; \\
model : ModelOne; \\
printedDocs : \mathbb{F}\, File \\
\Delta \\
files : \mathbb{F}\, File \\
\hline
files = printedDocs \cup \bigcup\{a : atms \bullet a.files\} \cup \bigcup\{u : usbs \bullet u.files\} \\
\mathrm{dom}\, cdr.possession = printedDocs \cup \bigcup\{u : usbs \bullet u.files\} \\
\forall f : files \bullet \forall d : \mathrm{ran}\, f.fileData \bullet d.model = model
\end{array}
$$

$$tick \mathrel{\widehat=} \bigwedge a : atms \bullet a.tick$$

$$loginATM \mathrel{\widehat=} [\, atm? : atms \,] \bullet atm?.loginATM$$

$$insertUSBATM \mathrel{\widehat=} [\, atm? : atms \,] \bullet [\, usb? : usbs \,] \wedge atm?.insertUSB$$

$$copyToUSBATM \mathrel{\widehat=} [\, atm? : atms \,] \bullet atm?.copyToUSBATM \wedge cdr.add$$

$$copyFromUSBATM \mathrel{\widehat=} [\, atm? : atms \,] \bullet atm?.copyFromUSBATM$$

$$printDocATM \mathrel{\widehat=} [\, atm? : atms \,] \bullet atm?.printDocATM \wedge cdr.add \parallel [\, \Delta(printedDocs) file? : File \mid printedDocs' = printedDocs \cup \{file?\} \,]$$

$$removeUSBATM \mathrel{\widehat=} [\, atm? : atms \,] \bullet atm?.removeUSB$$

$$logoutATM \mathrel{\widehat=} [\, atm? : atms \,] \bullet atm?.logoutATM$$

$$transferDocATM \mathrel{\widehat=} [\, atm? : atms;\ f? : printedDocs \,] \bullet f?.getId \parallel atm?.transferDocATM \wedge cdr.transfer$$

$$downgradeDocATM \mathrel{\widehat=} [\, atm? : atms \,] \bullet atm?.downgradeDocATM$$

$$witnessedDowngradeDocATM \mathrel{\widehat=} [\, atm? : atms \,] \bullet atm?.witnessedDowngradeDocATM$$

$$evaluateIntegrity \mathrel{\widehat=} [\, file? : files \,] \bullet file?.evaluateIntegrity$$

6 Future Work

There are a number of open questions which need to be explored in ongoing work. For example, the specific operation of data merging needs to be considered, beyond the initial discussion currently in Appendix C. This is the amplification mechanism (a critical requirement of our model), and it can serve to increase integrity if independent sources attest to the same data value(s). In general, most operations serve to reduce context dimensions' confidence values, but the data merging example is (one of) the exception(s) in which confidence in integrity may be increased. Also, one must handle the management of amplification of integrity via merging data from **partially** independent sources.

Similarly, integrity concepts such as Completeness and Consistency, which may make sense only across a file or set of files, are not yet considered.

One other factor which must be further expanded upon is the change in integrity (or *required* integrity) of the data due to change of the context/environment, *e.g.* at the

simplest level the data is no longer correct because the situation has changed. The issue of required/sufficient integrity depending on context has many critical implications for Military systems. As a simple and obvious example, a targeting process may report a geographic location within a defined error range. The resulting course of action or method of operation by a Commander could be very different if the location was in a very sparsely populated desert, as opposed to a densely populated urban environment. Context of the integrity measurement for the purposes of deciding whether sufficient integrity exists given the circumstances of the moment is essential.

One approach to these issues is to take this environmental context into account as an input into *evaluateIntegrity*. Another approach is to address the change in environment by use of a different integrity model index (which leads then to use, in particular, of a different *evaluateIntegrity* function) in that situation, with the appropriate constraints on the consistency of the set of relevant context dimensions for each model in order to make the interpretation consistent. This forms the basis of ongoing work.

The exploration in Appendix A of how to represent the Biba model within our framework raises the question of how we extend the model to accommodate security policy specification.

As discussed in Section 2.1, further research is required in order to explore various alternative mechanisms for handling data transition between models in a natural way.

Finally, we need to take the formal model further insofar as we need to explicitly claim and prove appropriate security properties.

As well as to anonymous referees, the authors are indebted to the following individuals for discussions, suggestions and advice: Angela Billard, Michael Chase, Aaron Frishling, Damian Marriott, Alex Murray, Tristan Newby and Chris North.

References

Bell, D. E. & LaPadula, L. J. (1973) *Secure Computer Systems: Mathematical Foundations*, Technical report, MITRE Corporation.

Biba, K. J. (1977) *Integrity Considerations for Secure Computer Systems*, Technical report, MITRE Corp.

Bishop, M. (2003) *Computer security: art and science*, Addison-Wesley, Boston, Massachusetts.

Casey, S. (2006) *The Atomic Chef: And Other True Tales of Design, Technology, and Human Error*, Aegean Pub Co.

Clark, D. D. & Wilson, D. R. (1987) A comparison of commercial and military computer security policies, *in IEEE Symposium of Security and Privacy*, pp. 184–194.

Ge, X., Polack, F. & Laleau, R. (2004) Secure databases: an analysis of Clark-Wilson model in a database environment, *in Advanced Information Systems Engineering - 16th International Conference, CAiSE 2004*, pp. 7–11.

Hanigk, S. (2009) Confidentiality is not enough: A multi-level Clark-Wilson model for network management, *in MilCIS 2009*.

Hu, H. & Feng, D. (2008) BIFI: Architectural support for information flow integrity measurement, Vol. 3, pp. 605 –609.

Lipner, S. B. (1982) Non-discretionary controls for commercial applications, *in IEEE Symposium on Security and Privacy*, pp. 2–10.

Liu, H.-C. (2010) A completed partial order of ordered semi-vector space of intuitionistic fuzzy values, *in Frontier Computing. Theory, Technologies and Applications, 2010 IET International Conference on*, pp. 1 –6.

Lunt, T., Denning, D., Schell, R., Heckman, M. & Shockley, W. (1990) The seaview security model, *IEEE Transactions on Software Engineering* **16**(6), 593–607.

Menezes, A., van Oorschot, P. & Vanstone, S. (2001) *Handbook of Applied Cryptography*, CRC Press.

Qingguang, J., Sihan, Q. & Yeping, H. (2006) A formal model for integrity protection based on DTE technique, *Science in China Series F: Information Sciences* **49**, 545–565.

Schell, R. & Denning, D. (1986) Integrity in trusted database systems, *in 9th National Computer Security Conference*.

Tang, L. & Qing, S. (2006) A practical alternative to domain and type enforcement integrity formal models, *in Inscrypt 2006*, Vol. LNCS 4318, Springer-Verlag Berlin Heidelberg, pp. 225–237.

Zhang, J., Yun, L.-J. & Zhou, Z. (2008) Research of BLP and Biba dynamic union model based on check domain, Vol. 7, pp. 3679 –3683.

Zhang, M. (2009) Strict integrity policy of Biba model with dynamic characteristics and its correctness, Vol. 1, pp. 521 –525.

Zhou-Yi, Z., Ye-Ping, H. & Hong-Liang, L. (2010) Hybrid mandatory integrity model composed of Biba and Clark-Wilson policy, *Ruan Jian Xue Bao (Journal of Software)* **21**(1), 98–106.

Zun, L., Tao, W. & Wei-hua, L. (2009) An integrity control model for operating system, *in Management and Service Science, 2009. MASS '09. International Conference on*, pp. 1–4.

Appendix A Biba

As another example of a specific model, we define here an instance of the generic model which implements the Biba model. We shall consider Integrity Values of only Low and High. For simplicity, we shall take Context Values also in this same space. Using a single Context Dimension, the *evaluateIntegrity* mechanism is then no more than a simple identity map from the (unique) Context Value in the Context Vector across to the Integrity Level space.

$$Binary ::= Low \mid High$$

ContextValueBinary

ContextValue

$$value : Binary$$

IntegrityLevelBinary

IntegrityLevel

$$value : Binary$$

$$type = IntegrityLevelBinaryType$$

In the specification of the model, *evaluateIntegrity* is defined as an identity map when the initial integrity is High. Otherwise the integrity, starting Low, remains Low.

contextUpdate must take the context vector of the object under consideration and produce a context vector whose (again single remember, as this is one dimensional) context value is the minimum of the context value of the object under consideration and that of the event (we have not defined an ordering on the Context Value space, and so we need to define it as a simple truth table).

BibaModel

IntegrityModel

$$unitVector : ContextDimension \pfun ContextValueBinary$$

$$\forall v : \operatorname{ran} unitVector \bullet v.value = High$$

$$\operatorname{ran}(\operatorname{dom}(\operatorname{dom} contextUpdate)) \subseteq ContextDimension \pfun ContextValueBinary$$

$$evaluateIntegrity \in IntegrityLevelBinary \times (ContextDimension \pfun ContextValueBinary) \pfun IntegrityLevelBinary$$

$$relevantDimensions = \{generic\}$$

$$\forall i : IntegrityLevelBinary;\ v : ContextDimension \pfun ContextValueBinary \mid (i, v) \in \operatorname{dom} evaluateIntegrity \bullet$$
$$(\exists p : ContextValueBinary;\ j : IntegrityLevelBinary \bullet$$
$$v(generic) = p \land evaluateIntegrity(i, v) = j \land$$
$$(i.value = Low \Rightarrow j.value = Low) \land$$
$$(i.value = High \Rightarrow j.value = p.value))$$

$$\forall l : IntegrityLevelBinary;\ v : ContextVector;\ e : Event \mid (l, v, e) \in \operatorname{dom} contextUpdate \bullet$$
$$\exists p, q : ContextValueBinary \bullet e.contextVector(generic) = p \land q.value = Low \land$$
$$p.value = High \Rightarrow contextUpdate(l, v, e) = \{(generic, v(generic))\} \land$$
$$p.value = Low \Rightarrow contextUpdate(l, v, e) = \{(generic, q)\}$$

Although we will not go into full details here, how this would be applied would be that there would be context vectors attached to subjects (users/processes) as well as data objects in the system. An operation which results in information flow to a subject/object (e.g. user when user reads data or object when user writes data) may result in the context vector of that flow target being updated.

An overarching security policy might say *e.g.* any operation that results in the integrity of the target being reduced is disallowed (Biba Strict Integrity Policy). This amounts to no write up or read down. We have not discussed security policy in this paper. It exists at the next level up - now we have the capability to evaluate integrity, our policy will specify, as a consequence, which actions are allowed. This is left to future work.

Appendix B Data Export

Transition of a DataElement from one model to another is accomplished by the following operation, which simply sets the initial integrity level of the new DataElement to the current integrity of the input DataElement, and initialises its context vector appropriately (unit value in the relevant dimensions of the new model). The raw data value is, of course, copied across unchanged.

$$
\begin{array}{|l}
exportDataElement : DataElement \times \downarrow IntegrityModel \rightarrow DataElement \\
\hline
\mathrm{dom}\, exportDataElement = \{ d : DataElement;\ m : \downarrow IntegrityModel \mid \\
\quad \forall k : \mathrm{ran}((d.model).evaluateIntegrity); \\
\quad l : \mathrm{ran}(m.evaluateIntegrity) \bullet k.type = l.type\} \\
\forall d : DataElement;\ m : \downarrow IntegrityModel \bullet \\
\quad \exists e : DataElement \bullet \qquad e.data = d.data \land \\
\qquad e.context = m.unitVector \land \\
\qquad e.initialIntegrity = d.integrity \land \\
\qquad e.model = m \land \\
\qquad exportDataElement(d, m) = e
\end{array}
$$

Note: the domain of the exportDataElement method is constrained so that it may only map between models with compatible Integrity Level spaces (or else we would require an additional layer of transition).

Appendix C Vector Space Structure and Fusion/Amplification

We shall consider the definition of addition and scalar multiplication operations on the space of context vectors **in the probabilistic model** in order to make it into a vector space.

Clearly, since we are trying to interpret the coordinates in the n-tuples as probabilities, we cannot naively add coordinate-wise or we quickly go out of the range $[0, 1]$.

As a simple example of how amplification might be achieved within our framework, we define the addition of vectors to correspond to the simple combination of the probabilities from two *independent* sources of the same data. Consider a trivial example of two sensors reporting on the outcome of a (fair and unbiased) coin toss. One reports accurately with probability c and the other with probability d. The "sum" of coordinates c and d, *i.e.* the probability that say the result is heads given that both sensors report heads, would be $cd/(1 - c - d + 2cd)$.

Extending this coordinate definition for the whole vector, for Context Vectors $\underline{c}$ and $\underline{d}$ and a scalar λ, we may define

$$(c + d)_i = c_i\, d_i/(1 - c_i - d_i + 2c_i\, d_i)$$
$$(\lambda \cdot c)_i = {c_i}^{\lambda}/((1 - c_i)^{\lambda} + {c_i}^{\lambda})$$

This effectively makes the amplification operation in the data space, for independent sources, homomorphic with vector addition in the context space. That is, the amplification a of independently sourced Data Elements d_1 and d_2 is such that $c(a(d_1, d_2)) = c(d_1) + c(d_2)$, where the function c returns the context vector of the data item. (We assume for simplicity now that the underlying data is actually the same between the two sources. Cases where there is overlap or even contradictions are to be considered once we have made sense of this simplest possible case). Note that this resultant piece of data $a(d_1, d_2)$ then decays by context update as operations are subsequently performed on it, etc.

In practise, for not fully independent sources, the amplification function (arrived at by Dempster-Shafer or other mechanism) will be such that $c(a(d_1, d_2))_i$ is less (coordinate-wise) than this (upper) bound. In fact, the discrepancy between the **vectors** $c(a(d_1, d_2))$ and $c(d_1) + c(d_2)$ will give some **measure** of the degree of dependence of the sources, *e.g.* their independence is correlated with $\frac{c(a(d_1,d_2))\cdot(c(d_1)+c(d_2))}{(c(d_1)+c(d_2))\cdot(c(d_1)+c(d_2))}$, *i.e.* making use of the scalar product in the vector space.

We may check that these definitions of vector space addition and scalar multiplication satisfy the appropriate relations to be a vector space.

However, a (trivial) observation is that we at best have a semi-vector space [Liu 2010] (there is no additive inverse of any element).

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA	1. CAVEAT/PRIVACY MARKING

2. TITLE	3. SECURITY CLASSIFICATION
A Formal Integrity Framework with Application to a Secure Information ATM (SIATM)	Document (U) Title (U) Abstract (U)
4. AUTHORS	5. CORPORATE AUTHOR
Mark Anderson, Paul Montague and Benjamin Long	Defence Science and Technology Organisation PO Box 1500 Edinburgh, South Australia 5111, Australia

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO–TR–2726	AR 015-350	Technical Report	October, 2012

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. No. OF PAGES	12. No. OF REFS
2012/1074321/1			32	19

13. URL OF ELECTRONIC VERSION	14. RELEASE AUTHORITY
`http://www.dsto.defence.gov.au/publications/scientific.php`	DSTO Distinguished Fellow (Cyber)

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for Public Release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SOUTH AUSTRALIA 5111

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS

No Limitations

18. DSTO RESEARCH LIBRARY THESAURUS

Computer security
Data integrity

19. ABSTRACT

Information Security is traditionally treated in three main categories: *Confidentiality*, *Integrity*, and *Availability*. While much work has been done on modelling Confidentiality and Availability, aspects involving comprehensive modelling and quality of data integrity in complex systems appear to be, on a relative scale, much less well understood and implemented. Further, most work on Integrity and resultant implementations seems to have focussed more on a matters related to source authentication and transmission assurance. However, the quality of data aspect is becoming more critical for attention, given the increasing levels of automation of information fusion and data transformation in a globalised Cyberspace.

In this paper, we survey the existing integrity models and identify shortcomings of these with regard to a general integrity framework encompassing the quality of data aspect. We then propose and formally model a new framework, illustrating the approach with reference to use cases built around the Secure Information ATM (SIATM) - a highly accreditable security system currently under development.