I. INTRODUCTION

DIFC proposed by Andrew Myers and Barbara Liskov [2], allows users to share information with a distrusted component yet control how that component propagates the shared information to others and provides an end-to-end security guarantees and several real-life implementations exist. The model extends the lattice-based information flow models by allowing users to declassify information in a decentralized way, and improves support for fine-grained data sharing. DIFC became very popular as it provides end-to-end security guarantees and there exist several implementations and applications both at the programming language level and at the level of operating systems. Language-based DIFC systems [3], [4] augment the type system to include secrecy and integrity constraints enforced by the bytecode generator. Mobile ambients [1], has been proposed to capture all aspects of mobility) within a single framework that encompasses mobile agents, interaction of ambients and their mobility.

In this paper, we propose a variant of Mobile ambients called Labelled Mobile Ambients (LMA) model and show that it forms a succinct basis for DIFC in a language independent manner and subsumes concepts of declassification and endorsement. We define the semantics of LMA and show that it realizes DIFC. This is realized by an appropriate interpretation of in, out and open actions in terms of the standard read and write operations, with additional constraints based on the labels of the ambients involved in the action to succeed. Further, we show that LMA (i) enhances the security of distributed computations by integrating well known security models, enabling a wide range of applications like safe execution of untrusted components downloaded from the internet etc., (ii) succinctly demonstrates multilevel security properties, (iii) realizes declassification and endorsement, and (iv) provides obligations to be discharged by a programming language to realize multi-level security (MLS). The model enables derivation of necessary and sufficient conditions needed for realizing MLS properties in a LMA. Further, static analysis techniques can be used to certify proper information flow in such a model and avoid run-time information flow checks as much as possible; such an analysis can be used to validate whether a given implementation of an application satisfies the desired security properties and arrive at frameworks for realizing a given security policy.

In the sequel, we shall highlight some of the crucial features of LMA only due to limited space.

II. A LABELLED MOBILE AMBIENT MODEL

In this section, we propose an advanced mobile ambient model referred to as Labelled Mobile Ambients (LMA). LMA model is the same as mobile ambients (syntax is shown in Fig. 1) except that

- ambients are labelled (labels assigned to an ambient are static, i.e. they do not change at run time) and that we have a function \textit{label} that takes the name of an ambient as input and returns its label as output. Labels of the ambients come from the lattice \( L = (L, \leq, \oplus) \)
- the semantics of the operations \textit{in}, \textit{out} and \textit{open} have to be redefined taking the labels associated with the ambients (that get affected by the execution of the operation) into account

Given a process of the form \( n [\text{opr} \ m. P | Q] \), we say that \textit{subject} \( n \) intends to perform operation \textit{opr} on object \( m \). We now present the semantics of \textit{in}, \textit{out} and \textit{open} primitives in LMA and discuss the interpretation of modelling the environment. The latter is important from the perspective of secrecy/integrity as one needs to define “trust”.

### Names

<table>
<thead>
<tr>
<th>( n )</th>
<th>( P, Q )</th>
<th>( M )</th>
<th>Processes</th>
<th>Capabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (\nu n)P )</td>
<td></td>
<td></td>
<td>restriction</td>
<td></td>
</tr>
<tr>
<td>( 0 )</td>
<td></td>
<td></td>
<td>inactivity</td>
<td></td>
</tr>
<tr>
<td>( P</td>
<td>Q )</td>
<td></td>
<td></td>
<td>composition</td>
</tr>
<tr>
<td>( !P )</td>
<td></td>
<td></td>
<td>replication</td>
<td></td>
</tr>
<tr>
<td>( n[P] )</td>
<td></td>
<td></td>
<td>ambient</td>
<td></td>
</tr>
<tr>
<td>( M.P )</td>
<td></td>
<td></td>
<td>action</td>
<td></td>
</tr>
<tr>
<td>( \text{in} n.P )</td>
<td></td>
<td></td>
<td>can enter ( n )</td>
<td></td>
</tr>
<tr>
<td>( \text{out} n.P )</td>
<td></td>
<td></td>
<td>can exit ( n )</td>
<td></td>
</tr>
<tr>
<td>( \text{open} n.P )</td>
<td></td>
<td></td>
<td>can open ( n )</td>
<td></td>
</tr>
</tbody>
</table>
Semantics of “in”

\[ n_1[n_2(P) \mid n_3[in(n_2.Q) \mid R] \mid S \rightarrow n_1[n_2(P \mid n_3(Q)) \mid R] \mid S \]

The subject of the operation in \( n_2 \) in the above process is \( n_3 \) and the object is \( n_2 \) (although, indirectly \( n_1 \) is also affected by it). If we look at the changes after the reduction, we deduce that the contents of \( n_1 \) and that of \( n_2 \) have been modified. So, we consider this as a write operation by \( n_3 \). We interpret the above reduction as follows: (i) \( n_3 \) writes \( n_2 \) and (ii) \( n_3 \) writes \( n_1 \). This would mean that information flows (i) from \( n_3 \) to \( n_2 \) and (ii) from \( n_3 \) to \( n_1 \). Under the constraints of the lattice model of information flow, for these information flows to succeed we need that \( \text{label}(n_3) \leq \text{label}(n_2) \) and \( \text{label}(n_3) \leq \text{label}(n_1) \).

Semantics of “out”

\[ n_1[n_3[\text{out}(n_2.P) \mid Q] \mid R] \mid S \rightarrow n_1[n_2[P \mid n_3(Q) \mid R] \mid S \]

The subject of the operation out \( n_2 \) in the above process is \( n_3 \) and the object is \( n_2 \) (although, indirectly \( n_1 \) is also affected by it). If we look at the changes after the reduction, we deduce that the contents of \( n_1 \) and that of \( n_2 \) have been modified. So, we consider this as a write operation by \( n_3 \). We interpret the above reduction as follows: (i) \( n_3 \) writes \( n_2 \) and (ii) \( n_3 \) writes \( n_1 \). This would mean that information flows (i) from \( n_3 \) to \( n_2 \) and (ii) from \( n_3 \) to \( n_1 \). Under the constraints of the lattice model of information flow, for these information flows to succeed we need that \( \text{label}(n_3) \leq \text{label}(n_2) \) and \( \text{label}(n_3) \leq \text{label}(n_1) \).

Semantics of “open”

\[ n_1[\text{open}(n_2.P) \mid n_3[Q] \mid R] \mid S \rightarrow n_1[n_2[P \mid Q \mid R] \mid S \]

The subject of the operation open \( n_2 \) in the above process is \( n_3 \) and the object is \( n_2 \). If we look at the changes after the reduction we deduce that \( n_3 \) now has the access to the contents of \( n_2 \). So, we consider this as a read operation by \( n_3 \). We interpret the above reduction as follows as \( n_3 \) reads \( n_2 \). This would mean that information flows (i) from \( n_3 \) to \( n_2 \) and (ii) from \( n_3 \) to \( n_1 \). Under the constraints of the lattice model of information flow, for this information flow to succeed we need that \( \text{label}(n_3) \leq \text{label}(n_2) \) and \( \text{label}(n_3) \leq \text{label}(n_1) \).

Modelling the Environment

We treat the environment as the “root” (within which all the activities happen) and we assign it the label * (the highest label) which stands for all. The properties of the environment would then follow are: (i) the environment cannot write anything (it has no siblings or ancestors), (ii) no one can read the environment (it has no parent), (iii) we trust the environment not to read anything (i.e. not have any open capabilities) and (iv) anyone can write to the environment (in fact every ambient which wants to cross administrative boundaries would have to necessarily pass through the environment).

The LMA model realizes DIFC follows from:

- each ambient specifies an independent flow policy and retains control over the dissemination of its data
- code in an ambient with the authority of its owner ambient can modify its flow policy. It can declassify its data part by adding additional readers (through open operation). This is done on a per-ambient basis and there is no central declassification ambient
- each ambient ensures its data flow policy through in, out and open capabilities
- the environment satisfies the initial flow policy and does not have any active role in controlling the data flow

III. Security Properties of LMA

One of the advantages of the LMA is that using static analysis techniques, we can derive a set of constraints on the labels of the ambient that when satisfied guarantee that the process does not leak information. The advantages of such a static analysis are (i) it eliminates the need for runtime checks and (ii) enables certification of components (for example generated by a trusted compiler).

Observation:

The label of an ambient must be higher than the label of any of its child ambients

Rules for constructing the set of constraints: Let \( \text{enqueue}(S) \), where process \( P \) is in an ambient labelled \( I \) be the set of constraints to be satisfied by the labels of ambients of the process so that no possible execution leaks information. It can be derived using the following rules using structural induction:

1. \( \text{enqueue}(S) = \text{enqueue}(I) \)
2. \( \text{enqueue}(I) = \emptyset \)
3. \( \text{enqueue}(P \mid Q) = \text{enqueue}(P) \cup \text{enqueue}(Q) \)
4. \( \text{enqueue}([P] = \text{enqueue}(P) \cup \text{label}(P) \leq \text{label}(n)) \)
5. \( \text{enqueue}(\text{open}(n.P) = \text{enqueue}(P) \cup \{l \leq \text{label}(n)\}) \)
6. \( \text{enqueue}(\text{out}(n.P) = \text{enqueue}(P) \cup \{l \leq \text{label}(n)\}) \)
7. \( \text{enqueue}(\text{open}(n.P) = \text{enqueue}(P) \cup \{l \leq \text{label}(n)\}) \)
8. \( \text{enqueue}(\text{open}(n.P) = \text{enqueue}(P) \cup \{l \leq \text{label}(n)\}) \)

The information leak is captured below.

Definition 1: We say that a process \( P \) leaks information iff \( \exists Q \rightarrow Q \) and ambient \( n_1 \) is a child of the ambient \( n_2 \) in \( Q \) and \( \text{label}(n_2) < \text{label}(n_1) \).

Proposition 1: If \( P \) and \( Q \) are such that \( P \rightarrow Q \) then the following hold

1. \( \text{enqueue}(P) \subseteq \text{enqueue}(Q) \)
2. \( \text{enqueue}(P) = \text{enqueue}(Q) \)

Corollary 1: If \( P \) is such that \( \text{enqueue}(P) \subseteq \text{enqueue}(Q) \) then \( P \) will not leak information.

For several practical applications, strict enforcement of non-interference is too inflexible. However, it is very sensitive. In LMA. The process of declassification and endorsement envisaged in several works by Myers and others, can be realized in LMA through trusted principals.

IV. DISCUSSIONS

To our knowledge, this is the first attempt that shows an adaptation of ambient calculus to succinctly model DIFC and multi-level security properties. The LMA model provides a general framework for multi-level security in a programming language independent way. While using explicit labels for large systems may not be easy, we believe that using the notion of trusted principals, it should be possible to integrate with legacy systems. Further, generalizations of the model to dynamic labelling are being explored.

REFERENCES