Data-Centric Access Control for Cloud Computing

Thomas Pasquier, Jean Bacon
Jatinder Singh
University of Cambridge
firstname.lastname@cl.cam.ac.uk

David Eyers
University of Otago
dme@cs.otago.ac.nz

ABSTRACT

The usual approach to security for cloud-hosted applications is strong separation. However, it is often the case that the same data is used by different applications, particularly given the increase in data-driven ('big data' and IoT) applications. We argue that access control for the cloud should no longer be application-specific but should be data-centric, associated with the data that can flow between applications. Indeed, the data may originate outside cloud services from diverse sources such as medical monitoring, environmental sensing etc. Information Flow Control (IFC) potentially offers data-centric, system-wide data access control. It has been shown that IFC can be provided at operating system level as part of a PaaS offering, with an acceptable overhead.

In this paper we consider how IFC can be integrated with application-specific access control, transparently from application developers, while building from simple IFC primitives, access control policies that align with the data management obligations of cloud providers and tenants.

Keywords

Information Flow Control, Cloud Computing, Data Protection

1. INTRODUCTION

Given the shared nature of cloud infrastructure, and security concerns holding back its uptake, a key focus has been on isolating tenants (data and processing) in order to prevent interference and information leakage. The goal of isolation is to segregate tenants, protecting their data and computation, and to limit a tenant's (direct) knowledge of others. A common approach involves containing tenants by allocating them their own virtual machines (VMs), each VM maintaining its own operating system (OS). Containers [34] have enabled strong isolation of tenants over a shared OS, and more recently Unikernels [16] have made library OSs practical, allowing applications' software stacks to be compiled down to run directly over the hypervisor.

SACMAT'16, June 05-08, 2016, Shanghai, China

DOI: http://dx.doi.org/10.1145/2914642.2914662

Though strong isolation of tenants is important, many applications and services will require data sharing across and outside isolation boundaries. For example, government or a medical institution may provide several related services, the data for which may correspond to the same people, whose data has to be recorded separately for every service. Data originating outside the cloud, such as that gathered from medical or environmental sensors, may pass through several different cloud services before being stored in cloud-hosted databases, and subsequently used for various purposes. We see a requirement for both isolation and controlled sharing for cloud-hosted data, particularly when cloud services form part of wider 'big data' and Internet of Things (IoT) architectures [31].

Guidance, regulations and laws exist regarding the protection of data, particularly personal data. For example, EU directives aim to restrict the circumstances in which personal data may leave the EU's geographical boundaries [6]. Another example is the need to anonymise/pseudonymise medical data when used for research [35]. Both cloud providers and tenants are subject to data management obligations, many of which concern the flow of information.

Access control mechanisms currently in place in the cloud do not entirely meet the requirements of highly regulated sectors. These mechanisms typically address access to data by principals, and do not implement any further control once the data has been accessed. Further, these mechanisms are often principal-centric, application dependent and heterogeneous in their implementation. In practice, this means that as some data flows through a complex multi-component system, it may fall under different access control regimes, with varying granularity (e.g. a front-end application authenticating individual users versus a back-end database authenticating entire applications for whole-table access).

While the above mechanisms contribute towards data security, they are insufficient to meet entirely the complex requirements of today's software systems. None of them can control the proper usage of data once "out of the hands" of the data owner, i.e. beyond their direct control. Each mechanism has its place, and we propose to complement them with a means to express and enforce data usage requirements throughout a multi-component system. We argue that access control for the cloud should no longer solely be application-specific but should be data-centric, controlling data flows between applications. *Information Flow Control* (IFC) provides such a mechanism.

An outstanding challenge for IFC in cloud computing is to integrate IFC with data-specific access control require-

ments to enable continuous, data-centric, system-wide control. Work on IFC to date has focussed on steady-state operation, rather than on how IFC is set up as part of (dynamic) application lifecycles, particularly when applications may need to collaborate—considerations that are especially relevant to cloud computing. Further, there is potential for IFC to enable the data management obligations of cloud tenants, cloud providers and third parties to be clearly defined, and when combined with audit, to help demonstrate that obligations have been met.

The contributions of this paper are in detailing how an IFC system can best be designed and engineered to meet such concerns. The IFC mechanism provider, e.g. a cloud PaaS provider where IFC is enforced at OS level, can be expected to provide a correct enforcement mechanism, but cannot take responsibility for the correct definition of application policies. Those with data management responsibilities are usually the cloud tenants/application managers. They should be able to specify data-centric policies that operate independently of provider specifics and across application instances running on behalf of end users. Ideally, application developers should not need to be aware of policies; policy-related behaviour should be separated from the application. This improves general security and allows applications to run unmodified within different policy domains, e.g. organisational or jurisdictional. Again, policy should be data-centric, rather than principal or application-centric.

Application managers (cloud tenants) therefore need to be able to set up an IFC framework, often on behalf of individual end users, within which application instances are governed by the IFC regime. In simple cases, application instances may be isolated, in which case IFC ensures that data is not deliberately or inadvertently leaked due to bugs or misconfigurations. More generally, applications may be required to collaborate and share data, sometimes as part of an umbrella organisation or as unrelated applications running on behalf of the same user. We address how this can be achieved, especially the support the IFC system can provide to application managers, including a novel approach to dynamically interpose security transformation services to meet policy requirements and data management obligations, in a manner transparent to application instances.

2. BACKGROUND

We first outline current approaches to achieve application and principal-specific access control, as background for addressing IFC integration. We argued in §1 that access control should be data centric for a cloud-service model, and that IFC provides such an approach. We therefore define the basic model and mechanisms of IFC.

2.1 Current access control mechanisms

Access controls (AC), comprising authentication and authorisation, are the main means to control the dissemination of information. Typically, a principal is authenticated, and perhaps associated with various roles. Authorisation to access data is then carried out at policy enforcement points in the application to grant or deny access to system objects by principals (in roles). Once access to data is granted, generally no further control is applied to ensure the data is handled properly; the application is trusted not to leak the data. This has been seen as a shortcoming of AC systems as discussed in §5.2, but is typical of cloud implementations.

This application-specific approach is insufficient when it is important to remain in control of data after access. For example, in an IoT scenario, personal medical data gathered by sensors, monitoring a patient at home, may flow into cloud services and databases. In such a scenario, IFC allows the patient's policy on how the data can be used throughout its lifetime to be attached to the data. For example, a tag *medical-research* on data ensures that it can flow only to those conducting medical research, who also have this tag.

2.2 Information Flow Control (IFC)

It is vital for computer systems to control how information flows through them. IFC tracks and constrains the flow of information continuously throughout whole systems and ensures that data is handled according to the associated policy. Research on IFC dates back to the 1970's [5] in the context of centralised military systems. Here, data was classified system-wide as *public, confidential, secret* and *top-secret*. Later, decentralised IFC was proposed [21], and has formed the basis of subsequent IFC research and implementations, including our work on CamFlow [27].[1] When implemented at the OS level, IFC can be described as a data-centric, continuous, Mandatory Access Control (MAC) enforcement mechanism.

IFC relates to two data properties: its *secrecy* and its *integrity*; respectively, where the data is allowed to flow to (as defined by Bell and LaPadula) and where it can flow from (as defined by Biba). These concerns are represented by associating with an entity A, two security labels $S(A)$ for secrecy and $I(A)$ for integrity; *active* (e.g. processes) and *passive* (e.g. data) entities are labelled. Many IFC models use labels that comprise a set of tags, each tag representing a particular security concern (e.g. $S = \{medical\}$, $I = \{validated\}$). Tags are defined as required in order to represent policy, for example, relating to how personal medical data can flow. The **security context** of an entity is defined as the state of its two labels, S and I. The flow of data between entities composing the systems is only allowed towards equally or more constrained entities in order to guarantee for example, the proper usage of data.

These requirements are captured in the following constraints, which are applied on every data flow from an entity A to an entity B:

$$A \to B, \text{ iff } \{S(A) \subseteq S(B) \wedge I(B) \subseteq I(A)\}$$

Creation flows. If an entity *creates* an entity (active or passive), the created entity inherits the labels of its parents. In a context of OS-level IFC enforcement, examples of entity creation include a process creating a file, and a process forking a child process.

Privileges for label change. In addition to their S and I labels, certain entities may have privileges to add and/or remove tags from these labels. If an active entity A has a privilege to add t to its secrecy label, we denote this $t \in P_S^+(A)$, and to remove t from its secrecy label: $t \in P_S^-(A)$ (and similarly $P_I^+(A)$ and $P_I^-(A)$ are the privileges for integrity). An active entity may therefore have four privilege sets in addition to its security context. Though a created entity inherits the labels (security context) of its creator, privileges are not inherited and have to be passed explicitly. Application managers will typically set up application instances in security

[1] http://camflow.org/

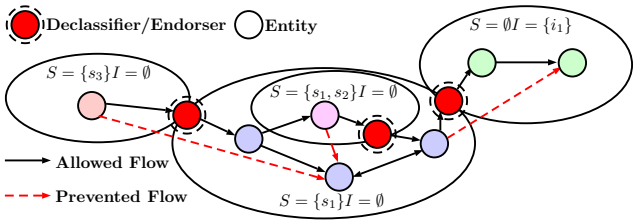

Figure 1: Declassification and endorsement.

contexts, without the privileges to change them, see §4.

Tag Ownership. In some IFC models [20], the concept of tag ownership is used to assign privileges to an entity. Privileges must be passed on with care, especially a privilege to remove a tag from a label, see §2.3 and §5.3.

Trust. If IFC is enforced at OS level, the applications running above the OS are obliged to run under the policy constraints expressed by the IFC labels' tags. They do not need to be trusted not to leak data [12]. In a cloud context, we believe that it is a reasonable assumption that the cloud service provider is more trusted and trustworthy than the many tenants' applications.[2] This means that when parties need to collaborate they do not need mutual trust, but only a shared trust in the underlying IFC enforcement mechanism.

Audit. Enforcement of IFC can provide the opportunity for recording flow decisions to build a provenance-like audit graph [26]. This can be analysed to understand where, how, why and by whom the data was manipulated within the system. This audit data, captured during IFC enforcement, can help to demonstrate compliance with regulations [24] by providing tangible traces, showing how the data was handled.

2.3 Security context domains

As described, the security context of an entity is its pair of labels, S and I. A security context domain comprises entities with the same labels. The flow of data can therefore be within a security context domain or into a more constrained domain. Once data has flowed into a more constrained domain further flows are confined to that domain or into increasingly constrained domains. For example, as shown in Fig. 1, data tagged as s_1 can flow to an entity tagged with $S = \{s_1, s_2\}$ but then can only flow within the $S = \{s_1, s_2\}$ domain. Generally, building a system with increasing constraints can lead to situations of "label creep".

In practice, perhaps after a certain time has elapsed, secret data may need to be made publicly available, or when data has gone through an encryption or anonymisation process it is allowed to flow more freely. To achieve these things, an IFC system needs to support more complex flow policies. We now discuss how these are provided within the IFC model. The rest of this paper is concerned with how such processes can be provided as part of an IFC system deployment, to ease adoption of IFC by cloud tenants.

[2] The major cloud providers tend to have more technical expertise than most tenants, and are more visible to regulators. The transfer of data from cloud providers to government agencies is a different problem, particularly where there is a legal requirement relating to "national interest". Because of this, mechanisms have been proposed to constrain data within geographical boundaries [10], or to encrypt data to prevent government surveillance [14]. These issues are beyond the scope of this paper.

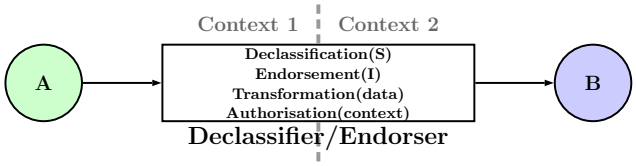

Figure 2: Declassifier/Endorser services as security context transformers.

Certain entities within an IFC system are given the capability to modify their labels in order to transfer information across security contexts; these are the four privilege sets defined in §2.2. For entity A, a label transformation is denoted:

$$A[S, I] \rightsquigarrow A[S', I']$$

An entity that performs such security context modification is called a *declassifier* when it modifies secrecy constraints, and an *endorser* when it modifies integrity constraints. Endorsers and declassifiers can therefore be seen as trusted gateways between security context domains, where the overall IFC constraints would prohibit a direct flow. Fig. 1 shows how trusted gateways allow information to flow across security domains when IFC constraints would disallow the flows. Such gateways can help ensure that regulation is enforced, e.g., medical data might only flow to a research domain if it has gone through a declassifier that applies a specified anonymisation algorithm. Therefore, a *transformation* of the data might also be needed, as well as checks, such as the time the data is authorised to be released.

Fig. 2 shows the basic behaviour of a declassifier/endorser. The main purpose, from an IFC perspective, is to apply a function to transform the S and/or I labels. For example, applying the transformation:

$$A[S, I] \rightsquigarrow A[S \setminus \{medical\}, I]$$

that indicates declassification over *medical*.

We aim to minimise the extent to which application developers need to be aware of IFC specifics. In §4.2 we show that declassification and endorsement can be offered as services by the IFC system, sometimes associated with transforming the data, i.e., tailored to the required policies of each application domain.

We now discuss how policies can be expressed in terms of IFC labels and consider how security context changes can be incorporated transparently into system design.

3. EXPRESSING POLICY IN IFC

In §2 we introduced IFC as a data-centric continuous MAC scheme. In this section, we discuss how complex policy can emerge from simple IFC constraints.

3.1 Simple applications

We envisage that some simple applications will be able to run in the same security context throughout their lifetimes. In this case, they can be set up as IFC-unaware application instances by an IFC-aware application manager, see §4.1. An example is an application instance, with no cross-application data sharing requirements, that is set up for an authenticated user. IFC isolates this user's instance and generated data from those of other users, by enforcing the non-interference principle between security contexts.

Consider the EU directive that aims to restrict when per-

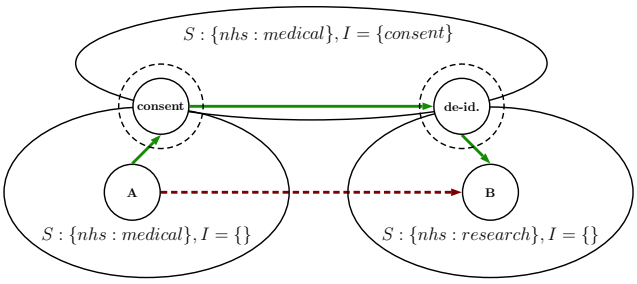

Figure 3: IFC enforcement of the medical data sharing policy, as described in Listing 1.

sonal data can leave EU's geographical boundaries [6]. We can envisage a security context domain where a tag *EU* is used within entities' security labels to authorise flows within the EU and to prevent flows outside, to entities without the *EU* tag. Such tags can be bound to hardware guarantees such as the mechanism presented in [10], that allows the geo-location of a server to be verified.

3.2 Building complex policies

Two examples show the need for security context changes:

1. Medical data must only be stored in encrypted form [7].

2. Medical data can be used for research purposes only if the consent of the owner is obtained and the data is anonymised [35], as illustrated by the policy described in Listing 1.[3]

We have seen that the IFC model, via declassification and endorsement, supports the required security context changes (§2.3). To achieve (1) we assume the database labels are set up so that only appropriately labelled data, e.g. *encrypted*, can flow into it. An encryption function must be applied to the data together with an endorser to add *encrypted* to its integrity label.

To achieve (2), as shown in Fig. 3,[4] the data owner's consent must be established and indicated with the data, an approved anonymising function must be applied to the data (e.g. simple deidentification for internal research), and a declassifier must transform its labels. The data is therefore constrained to flow between related applications in a (potentially) large-scale *medical* domain, in accordance with the obligations of the organisation, and demonstrably so. The assurance that such policies are expressible and enforceable, with adherence demonstrable makes cloud deployment of application domains that handle sensitive data more feasible.

We have argued that the provider of the IFC mechanism cannot be aware of the need for these data and label transformations, which therefore must be the responsibility of, in this case, "the managers of medical applications". We believe there is scope to provide system support for these transformations (§4.2). In some cases, data transformations are not needed, but only label transformation, such as when secret data can be released publicly after a specified time has elapsed.

Another policy could relate to data usage by third parties. The French National Data Protection Agency (NDPA) presents a set of recommendations when the manipulation of customer data is concerned, in an electricity smart metering context.[5] Data can be shared between parties, if it aligns with providing the services the customer registered for. Here we envisage that data is labelled, e.g. $S = \{ownerID\}$ and the service instances to which that user's data can flow are similarly labelled.

Another French NDPA recommendation is that the collected data can be used for marketing purposes only when anonymised, for example through aggregation. Here we see the need for an anonymising aggregation process and a security context change so that the output of the aggregation process can flow to the appropriately labelled marketing study.

The specification of IFC constraints and declassifiers/endorsers can ensure that such constraints are respected.

4. IMPLEMENTING POLICY USING IFC

We now discuss how the simple and complex policies defined above can be implemented in an IFC system.

4.1 Initialising application instances for users

Many application instances run in an unchanging security context throughout their lifetimes, subject to simple policies as described in §3.1. Application instances must be set up in the appropriate security context, according to the application and the principal on behalf the instance is acting.

In decentralised IFC, sets of tags can be created by any applications without a need for a central authority. Consider an application domain such as "UK National Health Service (NHS)" where a number of related applications operate on behalf of registered users, each user having an *nhs-id*. For authentication and authorisation, we assume the credentials of an NHS user are checked, say, against a registration database. A similar scenario arises if a set of local government applications run on behalf of registered citizens, for example, local tax collection, social services, electoral roll management, etc.

In an IFC context, we assume an application manager is given a framework in which application instances, for authenticated and authorised users (e.g. through RBAC policy), can be set up with appropriate domain-specific sets of tags in their S and I labels. In addition, we assume that each user's application instance is set up with their personal *nhs-id* tag in their S label. Such tags can be stored with users' entries in the domain registration database, for example using an IFC-aware database [29]. Simple application instances do not require privileges because they need no context changes during their lifetimes.

Unrelated applications that run on behalf of the same user may also share data if the user so desires. The applications may be offered by different cloud tenants, and current isolation mechanisms may make sharing difficult. If the same tag can be agreed for a given user by different applications, data can potentially flow between them. For example, if

[3]Often regulations relating to data privacy can be represented as constraints over the flow of data associated with some authorisation and/or transformation. Note that, though beyond the scope of this paper, translating law into a machine understandable set of rules is an active area of research of the computational law community [15].

[4]The NHS is the UK National Health Service.

[5]http://www.cnil.fr/fileadmin/documents/Vos_responsabilites/Packs/Compteurs/Pack_de_Conformite_COMPTEURS_COMMUNICANTS.pdf.

```
1 [S={nhs:medical}, I={}]->[S={nhs:medical}, I={nhs:consent}]:<consent_checker>
2 [S={nhs:medical}, I={nhs:consent}]->[S={nhs:research}, I={}]:<de-identification>
3 [S={nhs:medical}, I={nhs:consent}]->[S={harvard:research}, I={}]:<anonymisation>
4 [S={nhs:medical}, I={nhs:consent}]->[S={UN:research}, I={}]:<aggregation>
```

Listing 1: Policy concerning the release of medical data.

IFC is provided in a container-based cloud service, IFC tags potentially allow cross-container flows on behalf of a given user. A negotiation is therefore needed between the cloud tenants, prior to such applications being set up. This would involve a means to bestow the capability to use a specific tag, for example by means of a message from one application to another (see [17]). The details of how this can be achieved are left for future work.

4.2 Supporting dynamic security contexts

From a system design perspective we must consider how label transformations (by declassification and endorsement), sometimes associated with authorisation checks and transformations of the data, are provided to applications. For conciseness, we refer to DETA (Declassify, Endorse, Transform, Authorise) functions carrying out DETA policies. This is in order to effect the complex policies described in §3.2.

Our aim is that an application does not need to know about security context, nor about the gateways between security contexts, but simply attempts to send data to another entity within the system. This will breach IFC policy if it represents a transfer of data across incompatible security contexts. When such a breach is detected, the underlying platform should check the DETA policies specified by the relevant tag owners, indicating that a declassifier/endorser (or a combination) would allow the data to be transferred. Such declassifiers/endorsers are then invoked by the platform, the data flows through them, authorisations, transformations and label modifications are applied, as appropriate to enable the data to reach its specified destination. The interposition of these DETA functions is likely to occur during the establishment of a connection between two parties.

Fig. 3 illustrates such a scenario. The flow of information from A in the *nhs:medical* security context to B in the *nhs:research* security context would have been prevented through IFC policy (as $S(A) \not\subset S(B)$). However, following the policy described in Listing 1, an endorser and a declassifier are interposed on the path between A and B, by the platform. In addition to security context modifications, the endorser verifies that consent has been given for the use of data in a research context, and only allows the endorsement to occur in this case. The deidentification declassifier performs the transformation of data before it can be transferred into a different security context.

Authorisation for transforming labels and data could potentially be based on a property of the sender (e.g. does the sender have the appropriate role to send data across the two security contexts), attributes of the data or its structure, akin to an attribute-based access control decision [11], or based on the provenance of the data in a manner akin to a provenance-based access control decision [22].[6] While IFC policies themselves are relatively simple (solely based on subset relationships), DETA functions and their composition can implement policy as complex as required.

[6]Provenance data can easily be captured during enforcement of IFC, see [26]

Transformations are applied to data either to decrease its sensitivity or to increase its trustworthiness. Transformation examples include encryption, deidentification, anonymisation, etc. Further examples of transformations are as follows:
Unit conversion: In an IoT environment a database may receive data from a vast range of sensors. Some of the sensors may provide values in International Standard (SI) Units, while others may use the Imperial system or US Customary Units. In order to maintain the integrity of the database its input software is labelled with $I=\{SI\}$, to accept only SI units. An application involving persistent storage of sensor data may specify a policy that automatically converts Imperial and US units to their SI equivalent before input into the database. The process would be totally transparent for both the sensors and the cloud application, as the conversion would be handled through interposition of a declassifier/endorser service. Similarly, an application displaying the values to the end users, may specify automatic transformation to meet user preferences on output.
Sanitisation: In order to guarantee its integrity, an application may set its security context such that it accepts only sanitised data as input (i.e. specifying in its integrity label the requirement for the presence of a *sanitised* tag). Properly specified IFC policy can ensure that the data flows through a sanitisation endorser service before reaching its destination.
Declassification: After a certain period of time, the data is no longer considered sensitive and therefore can flow freely outside of the security context. Here the authorisation of the security context change is simply that sufficient time has elapsed.

As DETA policy is separated from application logic, the same application code can run under different policy regimes. The same base application can, for example, be constrained to comply with the requirements of different organisations, departments or jurisdictions, such as the EU and the US. Further, when collaborating with other parties, the policy being applied is independent from the application and defined in terms of the IFC tags by the tags' owner. This means that no trust is required in third parties, as long as proper interposition of DETA functions is guaranteed by the cloud provider (see Footnote 2).

4.3 Representing policies as graphs

We now describe how to specify the interposition of DETA services. Policy, such as that described in Listing 1, can be understood and represented as (potentially disjoint) directed graphs. The nodes in the graph represent a security context. The edges represent the specification of a DETA service, which provides a gateway from a source security context to a destination security context. The DETA specification must take account of how interactions between entities are implemented. For example, if a messaging middleware is used, the permitted message type might be part of a DETA specification. The service to interpose and other parameters as required, are also included.

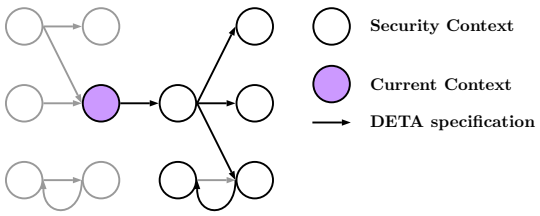

Figure 4: Policy subgraph to be considered.

In Fig. 4, the current context could be $[S = \{nhs : medical\}, I = \{\}]$, the edges originating from this node the consent checker, and the destination the context $[S = \{nhs : medical\}, I = \{nhs : consent\}]$. When IFC constraints would prevent the flow, there is a query for an appropriate path from the current security context to the destination security context. The DETA service(s) are then interposed between the relevant source and destination entities.

The concept of interposition was presented in Fig. 3, to effect a particular DETA policy. To find an applicable policy on an IFC flow failure, there is no need to query the whole set of graphs representing the whole set of policies applying to the platform. Only the subgraph which originates from the sending entity's security context, see Fig. 4, need be queried. When a new instance of an application, running in a particular security context, is created, the associated policy subgraph can be loaded. Given such caching, nodes must be informed of any changes to the global graph.

The syntax to express DETA services placement policy could be as follows (see Listing 1 for an example):
placement::= **security_context** '->' **security_context** ':' ⟨ DETA reference ⟩
security_context::= '[' 'S' '=' **label** ';' 'I' '=' **label** ']'
label::= '{' ⟨ **tag** ⟩ (',' ⟨ **tag** ⟩)* '}' | ∅
tag::= owner':'concern

The cloud provider is solely concerned with enabling the enforcement of application-provided DETA policies via the placement of DETA functions on disclosure paths. The policy being enforced is beyond its understanding or concern.

4.4 DETA services

It is important that the DETA services operate in a self-contained, standalone manner. This is in order to enable the requisite DETA functions to be interposed, when and as necessary, across various applications and runtime contexts.

Given the power of DETA services, in practice such services should be rigorously validated and verified. Any erroneous declassification or endorsement service, where dynamically interposed, has the potential to impact a range of applications, in possibly unforeseen ways. Further, it will be useful to ensure that DETA processes are audited, which is useful for both detecting errors and demonstrating policy adherence, see §5.4.

Our approach is agnostic to the method by which DETA services are effected. Some services may be long-running and well-known. However, we anticipate that in many cases, DETA services should be instantiated 'just-in-time,' to execute a particular transform or perform some authorisation verification. In a traditional cloud context, this might entail invoking a particular service within a cloud VM. Unikernels offer much potential in this space, by enabling small, very lightweight and (more easily) verifiable images that could encapsulate (only) a single DETA instance, and be rapidly instantiated and executed on demand [16].

5. RELATED WORK

5.1 Comparison with sticky policies

IFC is a simple, low-level mechanism. At a higher level, *sticky policies* have been proposed to achieve end-to-end control over data [4, 28]. In sticky policy systems, data is encrypted along with the policy to be applied to that data. To obtain the decryption key from a Trusted Authority (TA), a party must agree to enforce the policy. This agreement may be considered part of forming a contractual link between the data owner and the party decrypting the data.

Sticky policies provide no means to ensure the proper usage of data once decrypted. A malicious service could be black-listed by the TA, but only if and when a breach of agreement is detected. The system builds on the trust established between the data owner, the TAs and services interacting with the data. Our IFC approach builds only upon the trust between the data owner and the cloud provider.

5.2 Usage Control

Usage Control (UCON) [23] proposes to formalise a model that extends access control beyond server-side authorisation, to encompass obligations, conditions, continuity and mutability. The aim is a unified model, incorporating traditional access control, digital rights and trust management. Access decisions are made based on pre-, post- and ongoing-properties relating to mutable subject and object attributes, traditional authorisation, obligations that the subject must meet before or during an access to the object, and conditions that represents the environmental or system status. Purpose-based policy, or usage control in relation to privacy, has been demonstrated to be feasible using IFC [13].

IFC can be integrated into a distributed system environment [38], and can run on a variety of devices including cars [2], while the trust in the enforcement mechanism can be assured through the use of 'trusted hardware' [25].

The advantages of IFC compared with UCON and sticky policies are its conciseness of expression and the speed of an OS-level enforcement mechanism. IFC constraints, coupled with effective interposition of DETA services allow complex policy to be expressed and enforced.

5.3 IFC

Since 1997, most research on IFC has followed the decentralised IFC model proposed in [21], including CamFlow [27].

IFC in systems has been investigated via clean slate OS implementations [37] before moving towards a more practical implementation for standard OSs [12]. SELinux [33], the most well known example of a MAC implementation for Linux (via a Linux Security Module), provides (centralised) IFC, albeit only over the secrecy dimension.

An individual machine enforcing IFC can be connected through an IFC-aware communication mechanism to form a distributed system respecting IFC constraints [38]. From this, it is possible to build underlying IFC enforcement for a cloud environment [1, 27]. An IFC-aware communication mechanism can take the form of fully-featured communication middleware [27, 30]. In CamFlow, IFC constraints are checked on the establishment of a connection between two entities via the communication middleware. The validity of the connection is re-evaluated when/if any entity modifies its security context.

In general, IFC research has focussed on steady state op-

eration, and the labelling of simple web-applications e.g. [12, 37, 38]. Here, for the first time, we show how decentralised IFC can operate for cloud tenant applications.

5.4 Assisting compliance

The uptake of public cloud services by regulated sectors, such as healthcare, finance, etc. is comparatively low. To leverage the public cloud, it is necessary for such sectors to demonstrate compliance with their obligations, when cloud-hosted. Awareness of law and regulation relating to cloud computing is increasing [18] and there is ongoing research on technical mechanisms towards this.[7]

As stated by a US NIST member [9], *"monitoring and addressing security and privacy issues remain in the purview of the [tenant]"*, and further that mechanisms need to be provided not only when data is at rest or in transit, but also while in use. This is especially true in regulated sectors where financial consequences of data mishandling can be severe. Techniques such as "colouring" [8] or tainting [19] of data and resources have been proposed as a means to protect data in use. These approaches are similar to IFC, but do not provide the continuous aspect of enforcement, as colour or taint are only verified and acted upon at well defined parts ("sink points") of the system. When providing the same tracking granularity as IFC, colour/taint propagation is as costly as performing the simple subset verification of IFC.

IFC as a means to assist with adhering to regulations in a cloud computing context has been explored [32], and later extended to provide tangible proof of compliance [24, 26]. Other work, such as Cloudopsy [36], has focused on the tracking of information flow to provide such tangible proof, but without addressing enforcement issues. More generally, the tracking and recording of information flow is an active area of research, often under the umbrella of provenance [3]. The continuous enforcement of IFC, that can easily be combined with such data flow tracking, makes it an appealing technology, when evidence supporting compliance (or non-compliance) needs to be obtained.

6. CONCLUSION

Access control alone is not sufficient for current and emerging cloud-based systems. Personal data may originate from a wide range of sensors and flow to various cloud services. There is a clear need to control its flow, as it moves between systems and/or administrative domains. We see IFC as a key technology to augment traditional access control with end-to-end, lifelong data flow control.

We have focussed here on cloud-based IFC implementation, but we have begun to work on extending IFC for IoT architectures that include cloud services. A basic assumption of IoT is that data will flow widely, to a variety of services, and may come to be used for originally unforeseen purposes [31]. It is important that application logic is separate from policy to achieve such flexible application composition, while adhering to data owners' policies.

We have shown how an authenticated and authorised user can be set up in a security context with labels appropriate to the application domain and their personal identification, see §3.1. We envisage application domains such as health services, emergency services, education, smart cities, etc. A user initiating an application in such a context inherits the tags of the umbrella domain, and the user's tag as data owner is deployed as appropriate.

Work on IFC to date has focussed on steady state systems, rather than on how IFC is set up as part of (dynamic) application lifecycles, particularly when related applications may need to collaborate—considerations that are particularly relevant to cloud computing. Regarding the allocation of responsibilities: the IFC provider can have no responsibility for applications' policies but only to provide a trustworthy IFC mechanism and associated services. An application need not be written with IFC-awareness and we have shown how this can be achieved for simple policies in §3.1 and §4.1. Those responsible for data can specify data-centric management policy that will be respected system-wide, without understanding application or service specifics.

Complying with complex policies that require data transformation and security context change needs to be supported. We propose interposable label and data transformation services to achieve this, separating application logic from IFC policy enforcement, see §4. Thus, the same application code can run under different policy regimes, including different organisations, departments or jurisdictions, such as the EU and the US.

There is potential for IFC to enable the data management obligations of cloud tenants, cloud providers and third parties to be clearly defined, and when combined with audit, to demonstrate that obligations have been met. It is often stated that security concerns hinder cloud adoption. In particular, the uptake of cloud services by highly regulated sectors has lagged behind other sectors. We believe that IFC could help to facilitate adoption in such sectors.

Through automating the interposition of declassifiers and endorsers, as required for data to flow according to policy, we showed how the burden on application developers of being IFC-aware could be alleviated. This decoupling of policy specification and enforcement from application code has wide-ranging possibilities for current and future distributed systems.

[7] See IEEE Cloud, SI on Legal Clouds, 2015 and IEEE Workshop on Legal and Technical Issues in Cloud Computing.

7. ACKNOWLEDGEMENTS

This work was supported by the UK EPSRC grant EP/K011510 CloudSafetyNet. We acknowledge the support of Microsoft through the Microsoft Cloud Computing Research Centre.

8. REFERENCES

[1] J. Bacon, D. Eyers, T. Pasquier, J. Singh, I. Papagiannis, and P. Pietzuch. Information Flow Control for Secure Cloud Computing. *Transactions on Network and System Management SI Cloud Service Management*, 11(1):76–89, 2014.

[2] A. Bouard, B. Weyl, and C. Eckert. Practical information-flow aware middleware for in-car communication. In *Workshop on Security, privacy & dependability for cyber vehicles*, pages 3–8. ACM, 2013.

[3] L. Carata, S. Akoush, N. Balakrishnan, T. Bytheway, R. Sohan, M. Selter, and A. Hopper. A primer on provenance. *Communications of the ACM*, 57(5):52–60, 2014.

[4] D. W. Chadwick and S. F. Lievens. Enforcing sticky security policies throughout a distributed application. In *Workshop on Middleware Security*, pages 1–6. ACM, 2008.

[5] D. E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243, 1976.

[6] European Commission. Proposal for a General Data Protection Regulation, 2012/0011(COD), C7-0025/12, Brussels COM(2012) 11 final, 2012.

[7] J. C. Garner. Final HIPAA security regulations: a review. *Aspen Publishers, Managed care quarterly*, 11(3):15–27, 2003.

[8] K. Hwang and D. Li. Trusted cloud computing with secure resources and data coloring. *Internet Computing, IEEE*, 14(5):14–22, 2010.

[9] W. A. Jansen. Cloud hooks: Security and privacy issues in cloud computing. In *44th Hawaii International Conference on System Sciences (HICSS)*, pages 1–10. IEEE, 2011.

[10] K. R. Jayaram, D. Safford, U. Sharma, V. Naik, D. Pendarakis, and S. Tao. Trustworthy Geographically Fenced Hybrid Clouds. In *ACM/IFIP/USENIX Middleware*. ACM, 2014.

[11] X. Jin, R. Krishnan, and R. S. Sandhu. A Unified Attribute-Based Access Control Model Covering DAC, MAC and RBAC. *DBSec*, 12:41–55, 2012.

[12] M. Krohn, A. Yip, M. Brodsky, N. Cliffer, M. F. Kaashoek, E. Kohler, and R. Morris. Information Flow Control for Standard OS Abstractions. In *Symposium on Operating Systems Principles*, pages 321–334. ACM, 2007.

[13] N. Kumar and R. Shyamasundar. Realizing Purpose-Based Privacy Policies Succinctly via Information-Flow Labels. In *Big Data and Cloud Computing (BDCloud'14)*, pages 753–760. IEEE, 2014.

[14] T. Loruenser, A. Happe, and D. Slamanig. ARCHISTAR: Towards Secure and Robust Cloud Based Data Sharing. In *International Conference on Cloud Computing Technology and Science (CloudCom'15)*. IEEE, 2015.

[15] N. Love and M. Genesereth. Computational law. In *10th International Conference on Artificial Intelligence and Law*, pages 205–209. ACM, 2005.

[16] A. Madhavapeddy, T. Leonard, M. Skjegstad, T. Gazagnaire, D. Sheets, D. Scott, R. Mortier, A. Chaudhry, B. Singh, J. Ludlam, et al. Jitsu: Just-in-time summoning of unikernels. In *Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 559–573. USENIX, 2015.

[17] M. Migliavacca, I. Papagiannis, D. M. Eyers, B. Shand, J. Bacon, and P. Pietzuch. DEFCon: High-performance event processing with information security. In *USENIX Annual Technical Conference*, Boston, MA, USA, 2010.

[18] C. J. Millard, editor. *Cloud Computing Law*. Oxford University Press, 2013.

[19] D. Muthukumaran, D. O'Keeffe, C. Priebe, D. Eyers, B. Shand, and P. Pietzuch. FlowWatcher: Defending against data disclosure vulnerabilities in web applications. In *22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 603–615. ACM, 2015.

[20] A. C. Myers. JFlow: Practical Mostly-static Information Flow Control. In *26th SIGPLAN SIGACT POPL'99*, pages 228–241. ACM, 1999.

[21] A. C. Myers and B. Liskov. A Decentralized Model for Information Flow Control. In *Symposium on Operating Systems Principles (SOSP)*, pages 129–142. ACM, 1997.

[22] J. Park, D. Nguyen, and R. Sandhu. A provenance-based access control model. In *Annual International Conference on Privacy, Security and Trust*, pages 137–144. IEEE, 2012.

[23] J. Park and R. Sandhu. The UCON ABC usage control model. *Transactions on Information and System Security (TISSEC)*, 7(1):128–174, 2004.

[24] T. Pasquier and D. Eyers. Information Flow Audit for Transparency and Compliance in the Handling of Personal Data. In *IC2E International Workshop on Legal and Technical Issues in Cloud Computing (CLaw'16)*. IEEE, 2016.

[25] T. Pasquier, J. Singh, and J. Bacon. Clouds of Things need Information Flow Control with Hardware Roots of Trust. In *International Conference on Cloud Computing Technology and Science (CloudCom'15)*. IEEE, 2015.

[26] T. Pasquier, J. Singh, J. Bacon, and D. Eyers. Information Flow Audit for PaaS clouds. In *International Conference on Cloud Engineering (IC2E)*. IEEE, 2016.

[27] T. Pasquier, J. Singh, D. Eyers, and J. Bacon. CamFlow: Managed Data-Sharing for Cloud Services. *IEEE Transactions on Cloud Computing*, 2015.

[28] S. Pearson and M. Casassa-Mont. Sticky Policies: An Approach for Managing Privacy across Multiple Parties. *Computer*, 44, July 2011.

[29] D. Schultz and B. Liskov. IFDB: Decentralized Information Flow Control for Databases. In *European Conference on Computer Systems (Eurosys'13)*, pages 43–56. ACM, 2013.

[30] J. Singh, T. Pasquier, J. Bacon, and D. Eyers. Integrating Middleware and Information Flow Control. In *International Conference on Cloud Engineering (IC2E)*, pages 54–59. IEEE, 2015.

[31] J. Singh, T. Pasquier, J. Bacon, H. Ko, and D. Eyers. Twenty security considerations for cloud-supported Internet of Things. *IEEE IoT Journal*, 2015.

[32] J. Singh, J. Powles, T. Pasquier, and J. Bacon. Data flow management and compliance in cloud computing. *Cloud Computing, IEEE*, 2(4):24–32, July 2015.

[33] S. Smalley, C. Vance, and W. Salamon. Implementing SELinux as a Linux Security Module. *NAI Labs Report*, 1:43, 2001.

[34] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. In *SIGOPS Operating Systems Review*, volume 41, pages 275–287. ACM, 2007.

[35] UK Medical Research Council. Obtaining data from the Health and Social Care Information Centre for health research - a guide for researchers, 2015.

[36] A. Zavou, V. Pappas, V. P. Kemerlis, M. Polychronakis, G. Portokalidis, and A. D. Keromytis. Cloudopsy: An autopsy of data flows in the cloud. In *Human Aspects of Information Security, Privacy, and Trust*, pages 366–375. Springer, 2013.

[37] N. Zeldovich, S. Boyd-Wickizer, E. Kohler, and D. Mazières. Making information flow explicit in HiStar. In *Proc. 7th USENIX OSDI '06*, pages 19–19, 2006.

[38] N. Zeldovich, S. Boyd-Wickizer, and D. Mazières. Securing Distributed Systems with Information Flow Control. In *5th Symposium on Networked System Design and Implementation (NSDI 08)*, pages 293–308. USENIX, 2008.