A Labelling System for Derived Data Control

Enrico Scalavino, Vaibhav Gowadia, and Emil C. Lupu

Department of Computing, Imperial College London
{escala, vgowadia, e.c.lupu}@imperial.ac.uk

Abstract. Existing ERM/DRM systems and more generally usage control systems aim to control who accesses data and the usage data is subject to even after the data has been disseminated to recipients. However, once the data has been used, no control or protection is applied to the information created as result of the usage. We propose a solution to derive protection requirements for *derived data* that makes use of Multi-Level Security (MLS) labels to associate data with its protection level and usage functions (*transformations*) with the protection requirements of the data they can derive. Users are also associated with clearance labels according to their roles. Clearance and data labels are used to determine whether a user can access data as in traditional Mandatory Access Control systems, while labels associated with transformations are used to derive labels for derived data. The solution assumes that the amount of sensitive information flowing from the input to the output of a transformation can be deduced from the input data and the transformation itself, so that adequate protection can be associated with the derived output.

1 Introduction

Controlling the usage of digital resources has been the focus of an intense research activity in recent years. Individuals and organisations share a vast amount of data often in an uncontrolled fashion. Fast and pervasive data sharing facilitates both social relationships and inter-organisational cooperation but also raises new issues that were partially neglected before. Most information flows freely without restrictions; the only controls are governance procedures with which employees in companies are expected to comply and *Data Sharing Agreements (DSAs)* between partners regarding the handling of shared data. However, data has often an intrinsic commercial or personal value and must therefore be protected from undesired accesses and usages, regardless of its physical location and thus even after it has been received by unknown remote parties. Further attention to this topic has been prompted by industrial interests in the development of Digital Rights Management (DRM) solutions and by public interests in the enforcement of an increasing amount of legislation regarding the handling of private data such as HIPAA in the US.

Approaches proposed by both academia and industry are sometimes referred to as *Enterprise Rights Management* (ERM) and rely on a client-side Virtual Machine (VM) that ensures data usage complies with the associated usage policies. Data is cryptographically protected before being disseminated so that only a central *trusted authority*

(TA) (or *Control Centre*) can issue the decryption keys and access rights to the authorised VMs on client devices. Solutions often rely on Trusted Computing (TC) architectures to guarantee trusted execution on VMs. Although the number of existing solutions covers different requirements and scenarios, they are generally based on a common perspective and only differ in their management of user authentication, policy and rights retrieval, audit of user actions and other tasks [1–5].

Despite the wide range of control policies that can be specified and deployed in current solutions, the important issue of catering for *derived data* has been generally ignored. In DRM derived data is defined as a resource that contains parts of an original work. Park and Sandhu [6] define derived data as an object created as a consequence of exercising rights on an original one (e.g. a log file). Stemming from this definition, we define derived data as *an object created as a consequence of performing a transformation on one or several original ones*. The concept of *transformation*, i.e. any function applicable to one or several data objects that either modifies them or returns a new one, is central to our solution. Here, we use the terms data, objects and (data) resources interchangeably.

Derivative works are protected in most jurisdictions amongst others by copyright law, such as the Copyright Act in the US. Protection of derived data is also a big concern for companies exchanging data under specific DSAs. Results, obtained through the usage of shared information, are an asset often more valuable than the original data and thus included in the sharing agreements. Existing DRM/ERM and Usage Control systems mostly neglect the problem. A user authorised to access protected data under certain constraints could use it to produce an unprotected derivative work, thus infringing the rights of the original owner. Legislations are purposely ambiguous in defining when a resource can be considered as ”derived”. However, when there is a significant economic interest at stake, relying on the recipients' interpretation of law may prove hazardous.

Similarly to other information flow solutions [7–10] we use a labelling system to label data under several sensitivity domains. When a new data item is derived (or simply modified), the amount of sensitive information for each sensitivity domain with respect to the original data can either increase, decrease or remain unchanged. With each derivation the data may therefore be declassified or classified, i.e. its applied protection may be decreased or increased respectively. Since the sensitivity of derived data does not solely depend on the sensitivity of the original data but also on the type of transformation that has been applied to it, we associate sensitivity domains with transformations as well as with the data itself. This ensures that the derived data is correctly protected on the basis of all the resources that contributed to its creation.

The remainder of this paper is organised as follows: related work is presented in Section 2 whilst Section 3 describes an application scenario providing a context for the examples; Section 4 introduces the concept of *data label*; Section 5 introduces the concept of *transformation* and shows how our work can be applied to XML data, whilst Section 6 shows how to integrate labels into an example ERM system; Section 7 describes the label derivation mechanism for derived data; Section 8 introduces a solution to create user-customised sensitivity domains. Finally, conclusions are drawn in Section 9, which also briefly discusses future work.

2 Related Work

Protecting data and controlling its usage when it is disseminated amongst unknown recipients has been gaining increasing attention in recent years, in particular because of the impulse given to the topic by DRM and ERM systems developed in industry. The architecture at the core of most existing systems such as Liquid Machines [11], Marlin [12], Authentica [1] or Microsoft RMS (Rights Management System) [2], is well-described by Park et al. [4] who identify different architectures for ERM design. A Virtual Machine running on recipients' devices enforces a Control Set, i.e. a list of usage control policies or rights received from a remote Control Centre. Control Centres are remote services evaluating recipients' requests and issuing policies, decryption keys and encryption keys when data originators need to *publish* new data. The solutions comprising these elements generally differ in the authorisation policy language they use, the authentication mechanisms employed, the policy deployment methods and the techniques used to store credentials.

Several languages for policy specification have also been proposed. Standard XML-based languages such as XACML [13], XrML [14], ODRL [15] and EPAL [16] specify access and usage control policies over data disseminated amongst cooperating companies and users exchanging resources. Despite being able to express several policies such as authorisations and obligations, the existing languages and enforcement platforms do not address the problem of derived data. Policies can express *who* can use the data and *under which conditions*, but cannot express what protection should be applied to the output of the usage.

A number of studies have been conducted in the database community on *data lineage* or *provenance* [17–19]. Data Lineage concerns tracing how the data aggregated into a data warehouse has been derived from the data sources. Although the proposed solutions allow to know the lineage of the queried data, they do not deal with data protection and with the derivation of the protection requirements for the derived data. Moreover, the solutions are tied to the relational model of the data and thus unusable in a general ERM system. Atluri and Gal [20] provide a definition for *derived authorisations* as authorisations for data derived through a reversible transformation on some original data. They also propose a method to verify whether a set of derived authorisations is *safe* with respect to the authorisations applied to the original data. A set of authorisations is safe if a user not authorised to access the original data cannot derive it back by applying a reversed transformation on the derived data that he can access. However, the authors propose a simplistic derivation method for authorisations based on the union of the sets of all original authorisations.

To address the challenge of derived data our solution is based on the core concepts of Multi-level security and information flow systems [7, 8] and in particular on the approach adopted in the Asbestos operating system [10]. Information flow control systems aim to prevent the flow of sensitive data from secure processes to non-secure processes. This is achieved by associating sensitive data with security labels and processes with clearance labels and enforcing the *simple security property* and **-property*. A *label* is a set of $(tag, level)$ pairs where tags are identifiers for sensitivity domains and levels are discrete values that represent the current protection level applied to the data for each domain. In Asbestos, processes have two labels: a tracking label and a clearance label.

Tracking labels record the sensitivity of the most sensitive data received or observed so far, while the clearance label bounds the maximum tracking label a process can be associated with. Given the set of all labels LS and the set of all tags TS, a partial order is defined over the label dominance relation $\sqsubseteq$:

$$\forall L_a, L_b \in LS : L_a \sqsubseteq L_b \iff \forall t \in TS : L_a(t) \leq L_b(t) \tag{1}$$

where $L(t)$ indicates the current level for tag t in label L. Non secure information flows are prevented by forbidding processes from receiving data when the sender process' tracking label is not dominated by their clearance label. However, in data dissemination scenarios no check over the recipient clearance or rights can be performed before the dissemination as recipients are often not known in advance. Moreover, data senders other than the data originator may have little interest in checking the recipient's clearance label before sending the data. Controls cannot be introduced at programming language level either, e.g. by associating labels with variables and I/O channels, as in the Decentralized Label Model [21]. Instead data can be freely disseminated through any available channel since it is previously protected by encryption. An ERM system then guarantees that only authorised recipients will be able to decrypt the data. Our solution is based on the observation that as security labels flow from data to processes and from one process to the other, confidential content flows from data to its derivations through the transformations performed by the application accessing the data.

We propose a *floating labels system* to be integrated with traditional ERM architectures to control data derivation in dissemination environments. While in the Decentralized Label Model labels are not directly associated with data values but rather with I/O channels and variables, we consider labels as *sticky policies* [3] that are attached to the disseminated data. Our solution also allows data declassification (similar in concept to that allowed by Asbestos' discretionary labels). However, while in existing works [22] declassification is performed directly by data owners [21] or at programming language level [23] when needed by a process, we let declassification depend on the high-level transformations applied to the data when used. Moreover, information flow systems conservatively assume that any data produced by a process is the result of a *reversible transformation* of all the data received by the process in the past. Any received data contributes in fact to the increment of the recipient process's tracking label. This is not true in our system where the amount of information flowing strictly depends on the transformation used.

3 Scenario

We consider an application scenario where data is disseminated amongst several users working for cooperating organisations. Data is not only disseminated but also modified and new data is also created. As the data is transformed some information is lost and new information is created; the data protection requirements change accordingly. This implies that a different protection must be applied to the transformed data, depending on the original resources that contributed to its creation and the transformations applied. The scenario describes an accident that rapidly escalates into a threat to a larger surrounding area. Two civil protection agencies, namely the Police and the Red Cross,

immediately intervene on the scene. While lending support and carrying on the rescue operations, rescuers gather information on the accident and its effects and share it to better organise their action and manage the crisis. In this context data confidentiality must be protected for several reasons: 1) victims' privacy is governed by legislation; 2) the surrounding buildings' and area plants, as well as information on the local service providers' facilities may be used by criminals for future crimes; 3) information on the accident may cause panic and if broadcast by media may actually hinder the rescue operations.

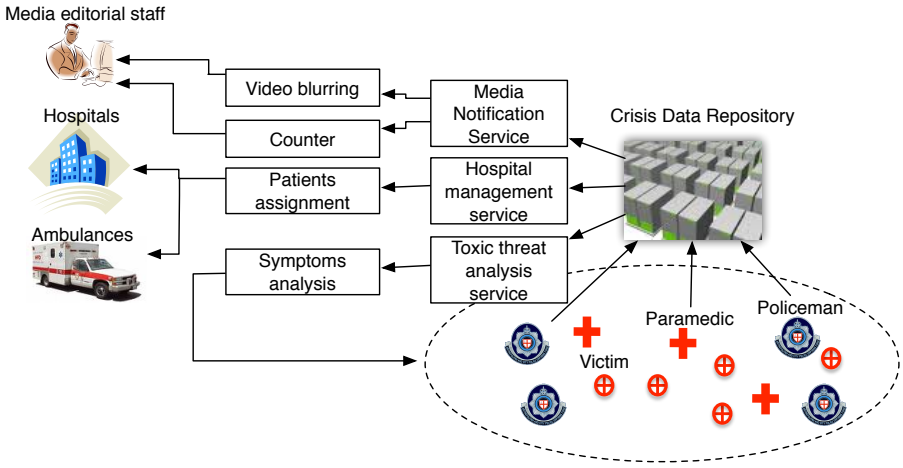

Fig. 1. Data sharing and elaboration scenario.

Figure 1 shows a particular interaction in this scenario where information on the accident is sent to a central repository. The data we consider for the initial dissemination includes the victims' medical and personal information, a video of the accident and rescue scene obtained by a local CCTV system and information on the nearby hospitals and care centres. Three services located in the repository process the disseminated information and re-disseminate the results in the accident area: 1) a *media notification service* managed by police; 2) a *hospital management service* managed by Red Cross and 3) a *toxic threat analysis service* also managed by Red Cross. We will illustrate in our examples how our labelling system derives protection requirements for the output of the three services. In particular, the media notification service takes as input the victims' information and periodically updates a statement for the media with the currently known number of casualties. It also decreases the video's resolution so that faces cannot be recognised and gruesome details cannot be distinguished. The hospital management service combines the victims' information with the list of nearby care centres and generates a document specifying for each injured person the centre where he/she will be hospitalised. The document is periodically disseminated to all ambulances and

paramedics. Finally, the toxic threat analysis service analyses the victims' conditions and calculates the risk of a chemical contamination in the area. The information is then disseminated among the rescuers so that they can react to a possible danger.

The information generated by the three services has different security requirements with respect to the original information gathered by the rescuers and stored in the central repository. On the one hand private information is removed and images of the scene can be accessed by a broader public since sensitive details are removed. On the other hand information on the hospital destination of the victims is considered sensitive while the information on the current threats the crisis is posing is highly confidential.

4 Sensitivity Domains and Data Labelling

Our solution stems from the idea that data can be protected under different domains and for different independent reasons. For example, data may be protected because it contains *private* or *commercially sensitive* information, or it may be related to *public safety* or *national security*. As in MLS systems, we identify each of such domains with a *security tag* and associate it with the data. When the data is modified, the amount of sensitive information it contains for each of the applied domains can either increase or decrease, as shown in Figure 2.

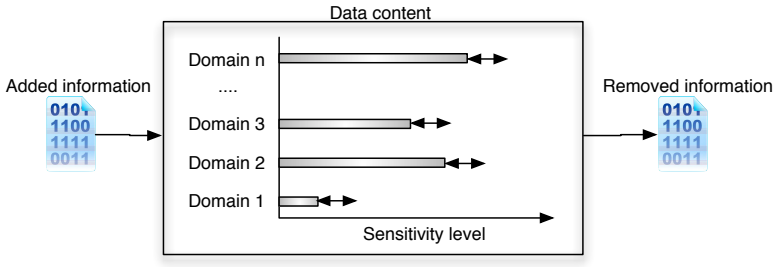

Fig. 2. The amount of sensitive information for each domain varies with changes to the data.

We represent this situation by defining a tag for each domain and associating it to a range of discrete security levels. In traditional information flow systems such as Asbestos such ranges are fixed. In other words, $\forall t \in TS, L \in LS : \ 0 \leq L(t) \leq n$. In our solution we adopt flexible ranges so that each domain can be associated with a different number of security levels. Each data item is associated with a *data label* containing a discrete security level for each existing tag and representing the current protection applied to the data for each domain.

Whenever new data is created, the initial level for each tag in its data label is decided by several *Content Verification Procedures* (CVPs). CVPs are boolean functions that verify specific conditions on the data content. Each security level of each tag is

associated with a CVP specifying whether the data label should contain that level for that specific tag. In our system a tag specification looks as follows:

$$t : * \rightarrow (0, CVP_{0,t}) \rightarrow (1, CVP_{1,t}) \rightarrow ... \rightarrow (i, CVP_{i,t}) \rightarrow ... \rightarrow (n, CVP_{n,t})$$

where the $*$ level indicates that a security domain is not applicable to the data item, i.e. that the data must not be protected under that domain. Note that the $*$ value is applied whenever no CVP returns true, otherwise the highest level whose CVP returns true is applied. CVPs are particularly useful whenever the initial security level for a given tag should not be decided manually by the data originator but on the basis of higher-level organisational policies.

5 Transformation and Data Model

In this setting, protecting derived data means deciding the security labels associated with it. To this end, we assume users and applications manipulate data through a series of *transformation functions*. These need to be known in advance by all partner organisation for two reasons. First, data originators or their organisations must be able to specify policies controlling how such transformations are used on the data they disseminate, even after the data crosses the organisational boundaries. Second, to specify how derived data must be protected, data originators must know how transformations actually process the original data.

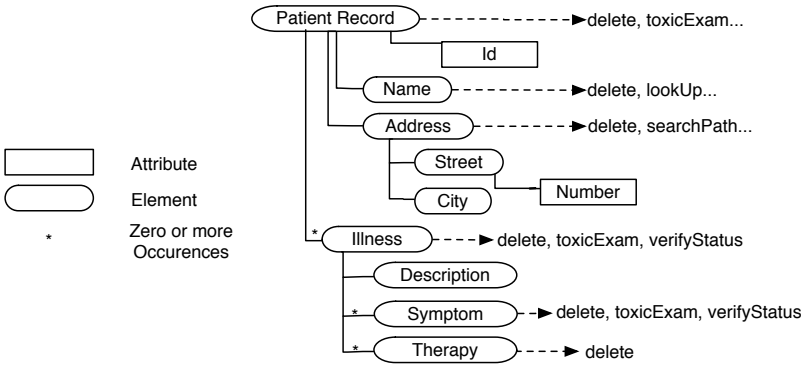

Fig. 3. Am example XML structure specifying transformations applicable to each element.

We use here XML data as an example for data labelling, without any loss of generality for our approach. For the sake of simplicity, we consider an XML document as a collection of nested elements. Elements can be either *data containers* or *data items*, i.e. they can contain other elements or unstructured data. Both data containers and data

items can be associated with attributes. We will not consider XML *entities* or *links* and consider only *valid* XML documents, i.e. documents that conform to a pre-defined document type definition (DTD) or XML schema. Consider the XML structure for a victim's medical record depicted in Figure 3. As described in works such as [24, 25], any element and attribute of the XML structure can be associated with an access control policy. Similarly, elements and attributes can be associated with data labels and can be accessed and modified by a transformation. The set of transformations that can be performed on the data is specified at schema or DTD level. In this way, whenever an organisation or one of its members creates an XML document of a specific type, the set of transformations that can be applied is known, even when data is shared with external partner organisations. Using XML documents makes also easier to define CVPs, as they can leverage the document structure to analyse the data content.

6 Enforcement and Evaluation

In the following we will use role hierarchies in conjunction with our labelling solution. Consider the example role hierarchies for Police and Red Cross shown in Figure 4. Data must be protected under four different domains: privacy, video privacy (for video data), media (for police official statements) and confidentiality (for sensitive information on the rescue operations). We consider the sensitivity of private information and of police statements to vary across two levels, i.e. $0 \rightarrow 1$ and the sensitivity of confidential information to vary across three levels (see section 7.2 for further explanation on the example). Roles are assigned permissions in terms of clearance labels containing a security level for each existing tag. Permissions are inherited along the hierarchy, therefore, given the set of all roles RS, for each pair of roles (r_i, r_j) such that r_i dominates r_j in the hierarchy, r_i's label dominates r_j's. In other words:

$$\forall r_i, r_j \in RS, t \in TS | r_i \gg r_j : \; L_{r_i}(t) \geq L_{r_j}(t) \tag{2}$$

For example, the police commander is assigned the label {(Privacy, 1),(VideoPrivacy,1),(Media,1),(Confidentiality,3)}. If not specified explicitly, roles are assigned to level 0 that also contains the special *public* group indicating every external subject who is not included in the role hierarchy.

Our labelling system is used to substitute the policy language or access control model of traditional ERM systems. For the sake of generality, we will not assume here any specific ERM platform. Clearance labels and tag definitions are kept on the system Control Centre. Whenever an XML document D is created and disseminated, each of its elements is first associated with a label L_e (included in the XML structure as an attribute) and encrypted with a symmetric key k_e. The approach proposed in [24] can be used to use the same encryption key for elements with the same data label. The set of keys $\{k_1 \dots k_n\}$ is then in turn encrypted for a specific Control Centre and attached to the data. Before accessing the data, a recipient *rec* must request permission and the decryption keys from the Control Centre, which verifies, for each data element in the document, the *Simple Security Property* (SSP):

$$\forall t \in TS \; \exists r \in RS | rec \text{ has role } r \text{ and } L_r(t) \geq L_D(t) \tag{3}$$

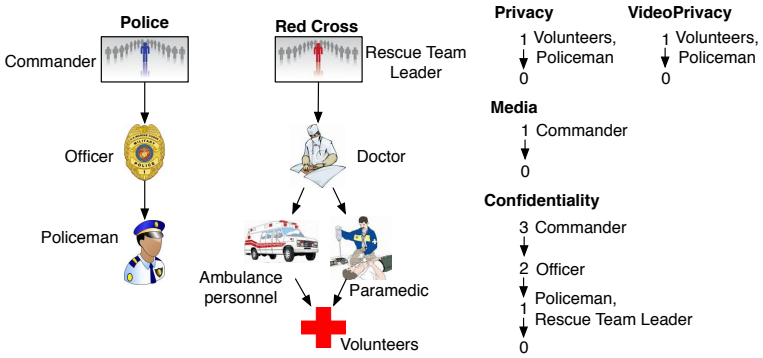

Fig. 4. Example of role hierarchy and sensitivity tags for the crisis management scenario.

In other words, the Control Centre verifies that the data recipient requesting access belongs, for each tag, to a role associated with a clearance label whose level is either the same or higher than that in the data label. If the evaluation succeeds for some data elements, the Control Centre can send the corresponding decryption keys to the requesting recipient who can then access the data.

Note that since only recipients actually satisfying the SSP can decrypt the data, there is no need for a control on the dissemination channels as in traditional information flow systems. Security is in fact enforced by the Control Centre and Virtual Machine.

7 Transformation Labelling and Labels Flow

Each transformation function that can be applied on the data is associated with a *function label* containing a sensitivity level for each existing tag. Function labels represent the security level required by data derived by the corresponding transformation. The default level for tags in a function label, if not specified explicitly, is considered to be 0. As we will describe later this means that the function does not increase the security level of the input data. The range of security levels for function labels does not include the $*$ value. Intuitively, derived data may not be sensitive and thus be associated with sensitivity level 0. However, a security requirement specified at creation time for the original data cannot be completely removed from the derived one. Therefore derived data cannot be associated with the $*$ level, unless the original data is as well. Note that function labels are different from the Asbestos' tracking labels. Function labels are defined together with the transformation and do not change with usage.

Both function and data labels are partially ordered as in MLS systems according to dominance rule 1 introduced in section 2. Two labels L_a and L_b are incomparable, i.e. $L_a \not\sqsubseteq L_b$ and $L_b \not\sqsubseteq L_a$ if at least one tag in L_a has a greater level then in L_b and at least one tag in L_b has a greater level then in L_a. The labels form a lattice whose minimal element is label $\bot = [*]$, specifying that no security is applicable or necessary. The tags' level of the maximal element $\top$ depends instead on the width of the security

ranges and contains for each tag the highest level possible. For each pair of labels L_a and L_b the greatest lower bound operator $L_a \sqcap L_b$ and least upper bound operator $L_a \sqcup L_b$ are defined as:

$$L_a \sqcap L_b(t) = \begin{cases} L_a(t) \; if \; L_a(t) \leq L_b(t) \\ L_b(t) \quad otherwise \end{cases} \tag{4}$$

$$L_a \sqcup L_b(t) = \begin{cases} L_a(t) \; if \; L_a(t) \geq L_b(t) \\ L_b(t) \quad otherwise \end{cases} \tag{5}$$

We must also consider the fact that when a specific security domain is not applicable for certain data (i.e. the level for the specific tag is equal to $*$), whatever transformation is applied to the data the $*$ value of the tag cannot be modified. We therefore introduce a further operator:

$$L_a \sqcup^* L_b(t) = \begin{cases} * & if \; L_a(t) = * \vee L_b(t) = * \\ L_a \sqcup L_b(t) & otherwise \end{cases} \tag{6}$$

7.1 Transformations and Policy Adaptation

As in existing ERM systems, once a decryption key has been obtained from a Control Centre the Virtual Machine installed on the client device locally enforces the usage control policies that are attached with the data. In our case, data is disseminated along with its applied data labels (one for each XML element or attribute). When the data is received by a recipient, the Virtual Machine ensures that any transformation applied to the data has an effect on the output data label as described below.

Whenever one or more data elements with data labels $L_{e_1} \ldots L_{e_n}$ are transformed through a transformation with function label L_f, the result d is assigned label:

$$L_d \longleftarrow (L_{e_1} \sqcup L_{e_2} \sqcup \ldots L_{e_n}) \sqcup^* L_f \tag{7}$$

For simplicity, in the following we will use the notation:

$$L_d \longleftarrow \bigsqcup L_{e_i} \sqcup^* L_f \tag{8}$$

This rule is very similar to the core rule of classic information flow systems. After the execution of a transformation function, the security level for each applicable tag is increased to be the least upper bound of the input data labels and function label. Note that the $*$ value is overridden if the labels of different input data elements have values higher than $*$ for the same tag. For simplicity and space reasons we consider here only transformations that return simple data items as output, i.e. unstructured data (and thus only one derived label is returned by rule 8). When a transformation is performed on one or more data containers, the label of the output data item is obtained considering the data labels of all the subelements as independent inputs.

The above rule only considers transformations that add value to the input data. However, transformations can also declassify data, e.g. by removing sensitive information.

To address this case, we associate transformations with two further *declassification labels*, namely the *general declassification label* L_f^g and the *relative declassification label* L_f^-. Note that declassification and function labels can be applied at the same time.

Intuitively, while the function label represents the value added by the transformation to the input data, the general declassification label represents which part of the sensitive information is lost in the transformation process. If not specified explicitly, the default value for tags in general declassification labels is the highest possible level for that tag. With L_f^g the core label derivation rule now becomes:

$$L_d \longleftarrow \bigsqcup (L_{e_i} \sqcap L_f^g) \sqcup^* L_f \tag{9}$$

The rule first considers the loss of information in the input data due to the declassification, and then increases the value of the declassified inputs according to the applied transformation. When for some tags' levels the input data does not contain enough sensitive information to be further declassified (i.e. $L_d(t) \leq L_f^g(t)$), the general declassification has no effect on such tags.

The relative declassification label represents the loss of information relative to the current sensitivity of the data. Consider for example an image data and a transformation that reduces its quality or resolution. The loss of information depends on the current resolution of the data, thus the security level required by the output data cannot be universally defined in a label. Relative declassification labels contain a real value in the range $[0 \ldots 1]$ for each tag. The default value for tags in relative declassification labels is 1. To apply relative declassification labels we introduce a further operator:

$$L_a - L_b(t) = \begin{cases} * & if\ L_a(t) = * \\ 0 & if\ L_a(t) \times L_b(t) < threshold \\ \lceil L_a(t) \times L_b(t) \rceil & otherwise \end{cases} \tag{10}$$

The relative declassification label allows any discrete level to be decreased by a specific percentage and then rounded up to the next discrete level. With L_f^- the core policy derivation rule now becomes:

$$L_d \longleftarrow \bigsqcup ((L_{e_i} - L_f^-) \sqcap L_f^g) \sqcup^* L_f \tag{11}$$

Note that a relative declassification label can cause a tag value to decrease to 0 depending on a specific threshold parameter defined with the function. The derivation rule described so far considers only transformations whose output, or at least its characteristics, are well-known. However, in many cases it is not possible to know in advance what the output of a transformation will look like. Examples are transformations performed by human users, such as text editing. A consequence of this impossibility is that a transformation may change one or more tags such that a wrong security level is applied to the output data. To address this problem we introduce the *decisional label* L^ι. Decisional labels associate each tag to a boolean value indicating whether a content verification procedure is required (as for newly created data) to determine the security level of the output data. The default value for a tag in a decisional label is *false* (i.e. CVPs are not used by default). The final policy derivation rule can thus be expressed as:

$$L_d(t) \longleftarrow \begin{cases} max(i) \mid CVP_{i,t} = true \; if \; L^t(t) = true \\ result \; of \; rule \; 11 \qquad\qquad otherwise \end{cases} \tag{12}$$

In other words, if the decisional label is set to true, the highest security level whose CVP is verified is applied to the data. If the decisional label is set to *true* for a tag but no CVP is satisfied after a transformation then the data modifications are considered not valid and the Virtual Machine provides to roll them back.

7.2 Examples

Consider the crisis management scenario described in section 3. Two distinct types of sensitive data are disseminated in the accident area: a video of the rescue operations and information on the victims. Information on nearby care centres is instead considered public. The tags agreed on by Police and Red Cross are defined as follows:

$$privacy : * \rightarrow (0, true) \rightarrow (1, NameAddressCheck());$$
$$videoPrivacy : * \rightarrow (0, true) \rightarrow (1, FaceCheck());$$
$$media : * \rightarrow (0, true) \rightarrow (1, VictimsCheck());$$
$$confidentiality : * \rightarrow (0, true) \rightarrow (1, Req(1)) \rightarrow (2, Req(2)) \rightarrow (3, Req(3));$$

where *NameAddressCheck()* is a procedure verifying whether the data contains personal names or addresses, *FaceCheck()* verifies whether a video shows identifiable faces and *VictimsCheck()* verifies whether the total number of casualties of the accident contained in a statement for the media is greater then zero. *Req(n)* CVPs simply verify whether the data originator explicitly requested an initial protection level for the data. When gathered, information on both victims and videos is labelled according to its content and the result of CVPs.

The video blurring transformation used by the media notification service decreases the video's resolution so that faces and gruesome details are made undistinguishable. It is thus associated with the general declassification label $L^g_{blur} = \{(videoPrivacy, 0)\}$ and with the relative declassification label $L^r_{blur} = \{(confidentiality, 0.5)\}$, with threshold parameter 0.5. L^r_{blur} ensures that every time a video is blurred, its confidentiality level decreases as more details are removed. The counter transformation used by the same service takes instead as input the information on victims and counts the total number of casualties, which is then inserted into an official statement for the media (updated every time the operation is run with new inputs). If the accident caused any casualties, the statement must be supervised by the Police commander before being publicly broadcast. Therefore, the counter transformation is associated with the general declassification label $L^g_{counter} = \{(privacy, 0)\}$ and the decisional label $L^t_{counter} = \{(media, true)\}$. The patient assignment transformation used by the hospital management service generates a document containing both personal information on the victims and confidential information on the hospitals they are assigned to. Therefore it is associated with the function label $L_{assign} = \{(privacy, 1), (confidentiality, 1)\}$. Finally, the symptoms analysis transformation used by the toxic threat analysis service generates, on the basis of the victims' medical conditions, an evaluation of the risk of toxic contamination

in the area. All rescuers must be aware of the risk, but the information must not be publicly disclosed to avoid panic. Therefore, the transformation is associated with the function label $L_{tox} = \{(confidentiality, 1)\}$ and with the general declassification label $L^{g}_{tox} = \{(privacy, 0)\}$.

Given the above function and declassification labels, both Police and Red Cross are sure the data produced by their services will automatically be associated with an adequate level of protection. Videos where faces are unrecognisable are no longer protected for privacy reasons and everyone can access them, unless gruesome or confidential details on the rescue operations are still visible. In the latter case, the more the resolution is decreased, the wider is the set of recipients who can access the videos. In particular, with each iteration of the blurring transformation confidentiality is decreased of one level, until level zero is reached and the video is public. Access to any statement produced by the police service is forbidden to the media if the number of victims is not zero. This is reasonable as in this case the Police commander might wait for the operations to be concluded before releasing an official statement. Information on the hospital destinations of the victims and on the current risk of a toxic contamination of the environment is also automatically protected as private or confidential (or both) so that only rescuers that need to use that information can access it.

For the sake of simplicity we showed a scenario where gathered data is processed only once and derived data is not re-processed. Defining the transformation labels for this kind of scenario is therefore relatively easy. However, there are other cases were derived data is given as input to other transformations iteratively. Examples are complex research environments were large amounts of data are processed and the results are used again as input for new experiments and tests. In those cases our approach proves to be even more useful as data protection is ensured all over the cycle and each intermediate result is correctly labelled on the basis of its current content and the transformations applied to it so far.

8 Custom Domains

Security domains represent the scopes under which data must be protected. Our construction assumes that all possible recipients for the disseminated data know all transformations that could be applied to it. The assumption is necessary as Virtual Machines must be able to enforce the label derivation rule every time data is transformed, and this would not be possible without knowing the transformation's labels. This implies that both domains and transformations must be agreed upon at organisational level, and in a Data Sharing Agreement if several distinct organisations want to share data. However, this does not allow users to define their own domains. Consider for example an employee in an organisation sending a private email to a colleague. None of the tags defined by the agreement between the organisation and its partners would apply to the data (unless it contains sensitive data for the organisation). However the user still needs to protect his email. Defining a completely new tag and attaching its definition to the data is not a viable solution. The user should in fact know all the possible transformations his data could undergo and specify the new tag's values for the function and declassification labels.

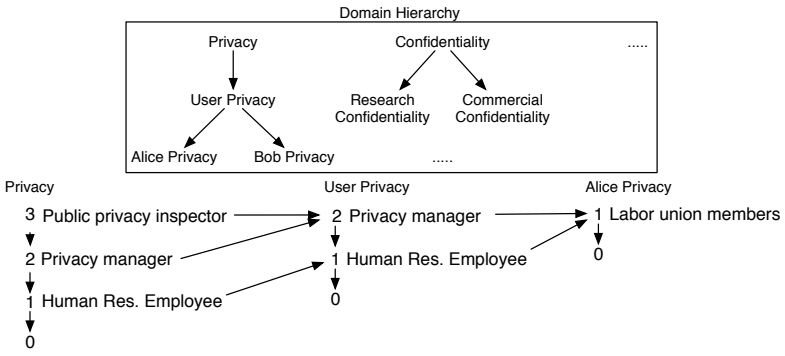

Fig. 5. Example of domain hierarchy and custom domains.

To address this problem we allow users to define custom domains extending a *parent domain* defined at organisational level. Defining such a relationship with an existing domain allows the Virtual Machine on recipients' devices to know how transformations modify the custom tag's level. The same level derivation as for the parent domain's tag can in fact be used. However, users must be able to specify security requirements for their own domains that are different from those of the parent domains. To do so a user must first specify a mapping from the parent tag's to the custom tag's security levels. The custom tag may thus have the same number or fewer levels than its parent. In the latter case several levels in the parent tag are mapped to one level in the custom tag. If the user does not specify any CVP for the new domain, default ones are applied that simply let the data originator decide (as the *Req(n)* procedures in section 7.2). The custom domain creator can then assign roles with different clearances even violating the permission inheritance rule 2 introduced in section 6. This is not a problem since custom tags are defined for users and thus are not subject to organisational constraints. In the example shown in figure 5 a custom *UserPrivacy* domain is specified by the organisation to allow employees to have personal data not subject to the control of a public privacy inspector. However Alice, the labor union representative, needs to disseminate messages amongst all the members of the union so that no one else can access them. The messages may contain names, dates, addresses for meetings and other information, therefore Alice decides to protect them as private data. Alice only needs two sensitivity levels (one for public data and one for data accessible only by members of the union) so she maps the two highest levels in the *UserPrivacy* tag to the only non-public level in her tag. Alice can then specify the clearance levels assigned to roles for her custom tag and that override those specified for the parent tag. The definition of the custom tag (level mappings and roles' clearances) is then attached to the data so that Control Centres and Virtual Machines can correctly handle authorisation requests and the data transformations. In other words the user modifies the roles' clearances by adding a new tag-value pair. In this example, Alice gives clearance 1 for the $Privacy/UserPrivacy/AlicePrivacy$ tag to members of the union. Finally, the de-

rived label for any transformation applied to such data is processed as if the data were protected by the privacy tag, with the exception that the derived levels are mapped to the custom tag's levels. Note that to avoid data leakage by rogue employees, tags at higher levels in the tag hierarchy always override tags at lower levels, if they both are applicable to the same data.

9 Conclusions and Future Work

Organisations often agree on sharing data in the expectation that systems and procedures are deployed to protect the disseminated data. However, no automatic protection is applied to the results of the data usages, for which the collaboration was initially set up. Our solution aims to offer a mechanism to control data that is derived through predefined transformations and whose content can vary according to a predefined scheme. We proposed our solution as an integration to existing ERM systems so that users do not have to bear the burden of defining further meta-policies for derived data control. However, such additional controls come at the cost of a restriction on the possible usages the data can be subject to and of a further effort for the organisations when stipulating their DSA. Note that we have only discussed data confidentiality aspects. Data integrity, i.e. the protection of data against unauthorised transformations, can be obtained with a simple extension to our model, where tags are specified for specific actions such as the *read* action (as by default in this work) and all the other applicable transformations.

We showed as an example the possible integration of our solution with role hierarchies. Future work will focus on investigating the integration with different policy languages to offer more flexibility in the specification of the security requirements. In particular we will study a *multi-level policy* system so that complex policies specified in languages such as XACML, ODRL and XrML can be merged into the label lattice presented in this work. The main challenge in doing this is the definition of a policy's *provided security level*, i.e. a measure to define whether a policy has stricter or looser requirements then another one. Such measure would allow us to create a policy lattice and thus to apply the label derivation mechanism shown above. We also aim to formally prove the information flow properties that can be achieved in our framework (e.g. the conformance to a defined DSA or other requirements) and to investigate possible criteria to be used when assigning labels to transformations so that desired properties can be obtained. Finally, we aim to further develop and integrate our solution with the XML data model, considering transformations returning complex data structures.

Acknowledgments We acknowledge financial support from the EC Consequence project (Grant Agreement 214859).

References

1. Authentica: Enterprise rights management for document protection (2005) White Paper.
2. Microsoft: Technical overview of windows rights management services for windows server 2003 (2005) White Paper. Accessed December 2009. URL: http://www.safecomprogram.gov.

3. Mont, M.C., Pearson, S., Bramhall, P.: Towards accountable management of identity and privacy: Sticky policies and enforceable tracing services. In: 14th Int. Workshop on Database and Expert Systems Applications (DEXA), Prague, Czech Republic (September 2003) 377–382
4. Park, J., Sandhu, R.S., Schifalacqua, J.: Security architectures for controlled digital information dissemination. In: 16th An. Computer Security Applications Conf. (ACSAC), New Orleans, USA, IEEE Computer Society (2000) 224–
5. Pretschner, A., Hilty, M., Basin, D.A.: Distributed usage control. Commun. ACM **49**(9) (2006) 39–44
6. Park, J., Sandhu, R.S.: The $UCON_{ABC}$ usage control model. ACM Trans. Inf. Syst. Secur. **7**(1) (2004) 128–174
7. Bell, D.E., LaPadula, L.J.: Secure computer systems: Mathematical foundations. Technical Report M74-244, The Mitre Corp. (May 1973)
8. D., D.E.: A lattice model of secure information flow. Commun. ACM **19**(5) (May 1976) 236–243
9. Papagiannis, I., Migliavacca, M., Pietzuch, P.R., Shand, B., Eyers, D.M., Bacon, J.: Privateflow: decentralised information flow control in event based middleware. In: DEBS. (2009)
10. Vandebogart, S., Efstathopoulos, P., Kohler, E., Krohn, M.N., Frey, C., Ziegler, D., Kaashoek, M.F., Morris, R., Mazières, D.: Labels and event processes in the Asbestos operating system. ACM Trans. Comput. Syst. **25**(4) (2007)
11. : Liquid Machines and Microsoft Windows Rights Management Services (RMS): End-to-end Rights Management for the Enterprise (2006) Accessed September 2009. URL: http://www.cmdsolutions.com/pdfs/LiquidMachines% 20Windows%20RMS%20Business %20White%20Paper%20FINAL%20060213.pdf.
12. : The Marlin Open Digital Content Sharing Platform (2009) Accessed February 2010. URL: http://www.marlin-community.com/.
13. : eXtensible Access Control markup language (xacml) version 2.0 (2005)
14. Guard, C.: eXtensible rights Markup Language (XrML) 2.0 (2001) Specification.
15. Iannella, R.: Open Digital Rights Language (ODRL), Version 1.1. W3c note, World Wide Web Consortium (2002)
16. P.Ashley, S.Hada, Karjoth, G., Powers, C., Schunter, M.: The enterprise privacy authorization language (epal 1.1) (2003) Reader's Guide to the Documentation.
17. Fan, H.: Tracing data lineage using automed schema transformation pathways. In: BNCOD. (2002) 50–53
18. Fan, H.: Data lineage tracing in data warehousing environments. In: BNCOD. (2007) 25–36
19. Fan, H., Poulovassilis, A.: Using schema transformation pathways for data lineage tracing. In: BNCOD. (2005) 133–144
20. Atluri, V., Gal, A.: An authorization model for temporal and derived data: securing information portals. ACM Trans. Inf. Syst. Secur. **5**(1) (2002) 62–94
21. Myers, A.C., Liskov, B.: Protecting privacy using the decentralized label model. ACM Trans. Softw. Eng. Methodol. **9**(4) (2000) 410–442
22. Sabelfeld, A., Sands, D.: Declassification: Dimensions and principles. In: In Proceedings of the 18th IEEE Workshop on Computer Security Foundations (CSFW05. (2005) 255–269
23. Lux, A., Mantel, H.: Declassification with explicit reference points. In: ESORICS. (2009) 69–85
24. Bertino, E., Castano, S., Ferrari, E.: Securing XML documents with Author-X. Internet Computing, IEEE **5**(3) (May/Jun 2001) 21–31
25. Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P.: A fine-grained access control system for xml documents. ACM Trans. Inf. Syst. Secur. **5**(2) (2002) 169–202