Data Distribution and Encryption Modelling for PaaS-enabled Cloud Security

Yiannis Verginadis, Ioannis Patiniotakis, Gregoris Mentzas
Institute of Communications and Computer Systems
National Technical University of Athens
Athens, Greece
{jverg.ipatini,gmentzas}@mail.ntua.gr

Simeon Veloudis and Iraklis Paraskakis
South East European Research Centre (SEERC)
International Faculty of the University of Sheffield,
CITY College
Thessaloniki, Greece
{sveloudis,iparaskakis}@seerc.org

Abstract—Some of the most valuable business benefits that accompany the cloud adoption cannot be exploited without addressing, first, new data security challenges posed by cloud computing distributed nature. A promising approach for alleviating these risks is to provide a security-by-design framework that will assist cloud application developers in defining appropriate context-driven policies that enhance cloud security at design-time and enforce access control at run-time. This paper discusses a generic and extensible formalism, called Context-aware Security Policy Model that can be tailored to the particular needs of different cloud applications for enhancing the privacy and confidentiality of sensitive data.

Keywords-Context-aware security; cloud security; Data privacy; Security by design

I. INTRODUCTION

The ever increasing adoption of Cloud computing brings about significant benefits for enterprises and users with respect to cost reduction, increased flexibility and business agility. Virtualised IT resources in the cloud enable organisations to realise significant cost savings and accelerate the deployment of new applications, thus transforming business and government at an unprecedented pace [1]. Nevertheless, several vulnerabilities constitute major concerns, as their potential exploitation may result in data confidentiality and integrity breaches [2].

Our work towards alleviating such security risks, involves an approach for assisting cloud application developers in defining effective security controls for the sensitive data of cloud applications [3]. To this end, a security-by-design framework is currently being developed, as a PaaS security solution with the objective to guide and assist developers through the process of defining the way that sensitive data should be persisted, while setting appropriate access control policies for safeguarding them. A prerequisite for this framework is a generic and reusable modelling formalism that will be used as the background for constructing appropriate code level annotations in order to materialize the security-by-design vision. This modelling background has two significant aspects. The first one involves all the ontological artefacts that allow for the generic description of classes, properties and instances that dictate the way that sensitive data should be persisted in cloud infrastructures (e.g. preferred or excluded locations, cryptographic protection required etc.). In this paper, we focus on these aspects of the security model. Its second part hinges upon an adequate access control scheme, one which takes into account the inherently dynamic and heterogeneous nature of cloud environments. This mainly refers to context-aware access control and it has been thoroughly discussed in our previous work [4].

In this respect, this paper sets out to present the part of a Context-aware Security Policy Model that can be considered valuable for a security-by-design framework. Our main objective is to shed light on the ontological reusable artefacts that allow for the creation of bootstrapping policies that will guide the way that sensitive data should be stored and encrypted in the cloud. We believe that the ability to provide generic and reusable ontological templates for expressing such security requirements, during cloud applications development, can provide the basis for a valuable toolset for maintaining data confidentiality and privacy in the cloud.

The rest of this paper is organized as follows. Section II sets the high-level view of all the relevant ontological artefacts for a security-by-design framework. Section III elaborates on the main data distribution & encryption modelling elements while Section IV presents the policy modelling approach for code level annotations that enhance cloud security. Finally, Section V presents conclusions and future work.

II. CONTEXT-AWARE SECURITY META-MODEL

In [4], we presented a meta-model that captures the main facets of a Context-aware Security Model; a model that can serve as the background framework for enhancing cloud security at design-time and constructing ontological templates for access control policies, to be enforced at runtime. This model involves elements that can be used for:

- enhancing cloud security by modelling fragmentation, distribution and cryptographic protection features that certain data artefacts must have. This refers to the Data Distribution and Encryption Element (DDE) that we thoroughly discuss in this paper.
- granting or denying access to sensitive data on the basis of dynamically-evolving contextual attributes [4]. This includes the Security Context Element, the Permission Element and the Context Pattern Element.

According to this meta-model, instances of these aforementioned facets formulate the Context-aware
Security Model. Based on this model, we are currently developing a security-by-design framework called PaaSword [3] that allows cloud application developers to annotate their code in order to denote the way sensitive data should be bootstrapped (e.g. fragments, distribution locations, excluded providers, encryption etc.) and automatically produce enforceable context-aware access control policies. Next, we discuss only the context model facets that are relevant to cloud security enhancement at design time.

III. DATA DISTRIBUTION & ENCRYPTION ELEMENTS

The DDE Elements are a significant part of the Context-aware Security Model [4] and refers to the details of encrypting data artefacts, fragmenting and distributing them in a way that even if the encryption key is somehow intercepted by an adversary, the sensitive information is still protected. The DDE elements comprise the following three top-level ontological concepts:

- **Cryptographic Type** - This class refers to the cryptographic process that should be adopted for encoding sensitive data.
- **Data Distribution** - This class captures aspects of the data distribution required for enhancing the security of sensitive data elements.
- **Data Fragmentation** - This class refers to the details about the way data should be fragmented in order to be distributed for security purposes.

For each of these top-level classes we provide details with respect to their sub-classes and the identified properties required for creating meaningful instances of the model. These instances will be used (see Section IV) for guiding the bootstrapping of sensitive data artefacts with respect to how, where and with what level of protection these data should be persisted. We note that dedicated software mechanisms are currently being developed for allowing the Administrator, the Product Manager or, depending on the organisation’s needs, even the Cloud Application developer to update and instantiate the Context-aware Security Model.

<table>
<thead>
<tr>
<th>Class Taxonomy Levels</th>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symmetric</td>
<td></td>
<td>Refers to the details of the chosen cryptographic algorithm that uses the same cryptographic key for both encryption of plaintext and decryption of cipher text (e.g. AES [6], RC4 [7]).</td>
</tr>
<tr>
<td>hasSymmetricKeySize</td>
<td></td>
<td>This property associates Symmetric with a positive integer that expresses the length of the key used for encryption/decryption (e.g. 128, 192 or 256 bits).</td>
</tr>
<tr>
<td>hasSymmetricBlockSize</td>
<td></td>
<td>This property associates Symmetric with a positive integer that expresses how long is the fixed length string of bits on which the cipher operates (e.g. 128 bits).</td>
</tr>
</tbody>
</table>

A. Cryptographic Type

The cryptographic protection of sensitive data to be managed by dedicated cloud application is discussed in terms of this ontological class. Table 1 presents the details of the Cryptographic Type element, where core classes and sub-classes are described along with their main associated properties. The Cryptographic Type is associated with the hasModeOfOperation property. This property associates the Cryptographic Type with a string that denotes how to repeatedly apply a cipher’s single-block operation to securely transform amounts of data larger than a block (e.g. ECB, CBC, CFB, OFB, CTR [5]).

B. Data Distribution

The Data Distribution class involves all the aspects required for controlling how the sensitive data artefacts should be distributed across different datastores in the cloud. This involves the following subclasses:

- **Distribution Metric** - This class quantifies the data distribution required for enhancing the security of sensitive information. It involves the properties numberOfVMS, numberOfServers, numberOfPhysicalLocations for associating a Distribution Metric instance with a positive integer that denotes the number of different VMs, servers and physical locations, respectively, that the sensitive data should be distributed to.
- **Preferred Location** - This class refers to the physical locations that should be considered as appropriate when distributing sensitive data.
- **Excluded Location** - This class refers to the physical locations that should be avoided when distributing sensitive data.
- **Preferred Provider** - This class refers to a certain trusted IaaS provider that should be considered as appropriate when distributing sensitive data.
- **Excluded Provider** - This class refers to a certain untrusted IaaS provider that should be avoided.

<table>
<thead>
<tr>
<th>Asymmetric</th>
<th>Refers to the details of the chosen cryptographic algorithm that uses different cryptographic keys for encryption (public key) of plaintext and decryption of cipher text (private key) (e.g. RSA [8], ECC [9]).</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasAsymmetricKeySize</td>
<td>This property associates an Asymmetric instance with a positive integer that expresses the length of the key used for encryption/decryption (e.g. 1,024, 4,096 bit).</td>
</tr>
<tr>
<td>hasCurve</td>
<td>This property associates an Asymmetric instance with a string that denotes the algebraic structure of elliptic curves used (e.g. P-192, P-224, P-256, P-384, P-521). Specifically, these refer to the sizes of primes used for recommending elliptic curves.</td>
</tr>
<tr>
<td>Hybrid</td>
<td>This class refers to the details of the chosen cryptographic algorithm that might combine symmetric-key and public-key cryptography (e.g. OpenPGP [10]). This is a subclass of both Symmetric and Asymmetric classes.</td>
</tr>
</tbody>
</table>
C. Data Fragmentation

The Data Fragmentation class refers to the way that sensitive data should be fragmented in order to be distributed for security purposes. The relevant classes, subclasses and properties are detailed in Table 2.

Table 2: Details of Data Fragmentation elements

<table>
<thead>
<tr>
<th>Class Taxonomy Levels</th>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational Data Fragmentation</td>
<td>hasPrivacyConstraint</td>
<td>This property associates a Relational Data Fragmentation instance with a string which restricts the way in which the fragmentation takes place (e.g. c1 = {Surname, SSN}, meaning that the two columns should not appear in the same fragment).</td>
</tr>
<tr>
<td>Horizontal Fragmentation</td>
<td>hasFragmentation</td>
<td>This class refers to the fragmentation of a relational database across its rows.</td>
</tr>
<tr>
<td>Vertical Fragmentation</td>
<td>hasFragmentation</td>
<td>This class refers to the fragmentation of a relational database across its columns.</td>
</tr>
<tr>
<td>Mixed Fragmentation</td>
<td>hasFragmentation</td>
<td>This class refers to a two-step fragmentation process of a relational database that may use both horizontal and vertical fragmentation based on certain conditions.</td>
</tr>
<tr>
<td>non-Relational Data Fragmentation</td>
<td>hasFragmentation</td>
<td>This Class refers to different types of possible fragmentation of a non-relational database.</td>
</tr>
<tr>
<td>Sharding</td>
<td></td>
<td>Sharding class refers to the details of different data distribution schemes across multiple servers, so each server acts as the single source for a subset of data/aggregates.</td>
</tr>
<tr>
<td>Replication</td>
<td></td>
<td>Replication class refers to the details of copying data/aggregates across multiple servers, so each bit of data can be found in multiple places.</td>
</tr>
<tr>
<td>Master-slave</td>
<td></td>
<td>Master-slave replication subclass (of Replication) refers to the details of making one node the authoritative copy that handles writes while slaves synchronizing with the master and handling reads.</td>
</tr>
</tbody>
</table>

We note that further details on all aspects of the Context-aware Security Model [4], including UML class diagrams and model’s serialisation in RDF, can be found here: http://imu.ntua.gr/software/context-aware-security-model

IV. DDE Policy Modeling

Our PaaS solution aims at supporting the following three kinds of security policy: (i) Data Encryption (DE) policies that specify the kind of cryptographic protection applied to a sensitive data object. (ii) Data Fragmentation and Distribution (DFD) policies that specify how a sensitive data object is fragmented and distributed over different physical servers for privacy reasons. (iii) Access Control policies that determine when to permit, or deny, access to sensitive data by taking into account a set of contextual attributes that are associated with the entity requesting the access. Both the DE and DFD policies entail security controls that are enforçable during the bootstrapping phase of a cloud application. On the other hand, Access Control policies entail security controls that are enforçable during application execution time.

In order to aid application developers in articulating effective security policies for sensitive data objects, our PaaS solution must be underlain – for each supported policy kind – by an ontological model that exhibits the following characteristics: (i) It is based on a framework of relevant concepts, and their interrelations, that capture all those knowledge artefacts that are required for describing a policy. For example, in the case of DE policies, such knowledge artefacts would include the particular encryption algorithm used, and hence the strength and kind (e.g. symmetric, asymmetric, hybrid) of the encryption. This framework was outlined in Section III. (ii) It uses an extensible formalism for accommodating the aforementioned framework that unravels the definition of a policy from the code employed for enforcing it, bringing about the following advantages:

- It allows the aforementioned framework of relevant concepts and properties to be extended and instantiated independently of the code employed by the application. Such an extension/instantiation aims at customising the framework to the particular needs of a given application.
- It forms an adequate basis for reasoning generically, i.e. independently of the code employed by the application, about the correctness and consistency of the policies, hence about the effectiveness of the security controls that they give rise to.

In addition to Section’s III Context-aware Security Model, this paper proposes an ontological model for DE policies and provide a brief account of a similar model for
DFD policies. The third kind of policy that our PaaS solution aims at supporting, namely Access Control policies, has been presented in our previous work [4].

A. DE Policy Model

1) DE Policy Rules: Following an approach inspired by the XACML standard [12], a DE policy comprises one or more rules. A rule is the most elementary structural element and the basic building block of a policy. A generic template for DE rules is depicted in Table 3.

Table 3: DE rule template.

<table>
<thead>
<tr>
<th>controlled object</th>
<th>encrypted with</th>
<th>cryptographic type</th>
</tr>
</thead>
</table>

This template defines a generic structure to which all DE rules in our PaaS framework adhere. It comprises two attributes. The first, namely **controlled object**, identifies the sensitive data object which is to be protected. It draws its values from the *pcm:Object* class of the Security Context Element [4]. This class encompasses a structure of subclasses for generically classifying the different kinds of objects that may be manipulated by a cloud application. The actual objects appear as instances of these subclasses. The second attribute, namely **cryptographic type**, identifies the encryption algorithm which is to be applied to the controlled object. It draws its values from the *pdm:CryptographicType* class of the Data Distribution and Encryption model defined in Section III. This class encompasses a structure of subclasses for generically classifying encryption algorithms according to their kind (e.g. symmetric, asymmetric, hybrid) and strength. The actual algorithms appear as instances of these subclasses.

2) Ontologically Representing DE Rules: A DE rule takes the form of an instance of the class `pbe:Rule` (see Figure 1). Individual rules appear as instances of this class. Two object properties are attached to `pbe:DERule`, namely `pbe:hasCtrldObject` and `pbe:hasDEElement`. These are intended to capture the two attributes of the DE rule template. They associate a DE rule with instances of the classes `pcm:Object` and `pdm:CryptographicType` respectively. These instances represent the values that the two attributes assume, i.e. the actual controlled object which the rule protects (drawn from the class `pcm:Object`) and the actual encryption algorithm used (drawn from the class `pdm:CryptographicType`).

3) Ontologically Representing DE Policies and Policy Sets: A DE policy takes the form of an instance of the class `pbe:Policy`. It is associated with the rules that it comprises through the property `pbe:hasDERule`. Inspired by the XACML standard, DE policies are grouped into policy sets. In our ontological model, a policy set takes the form of an instance of the class `pbe:PolicySet` (see Figure 2). A policy is associated with its encompassing policy set through the property `pbe:belongsToPolicySet`. A policy set may exhibit a hierarchical structure and comprise one or more other DE policy sets. This recursive inclusion is captured by rendering the `pbe:belongsToPolicySet` property applicable to DE policy sets too (see Figure 2).

B. DE Policies in Linked USDL

Section IV.A outlined a model for the generic representation of DE policies. This section demonstrates how this model can be incorporated into the ontological framework provided by Linked USDL [13], and in particular, into USDL-SEC – Linked USDL’s security profile. By drawing upon Linked USDL, our approach avoids the use of bespoke, non-standards-based, ontologies for the representation of data encryption policies (see Section V for a relevant outline of such ontologies). Instead, it is based on a diffused and lightweight ontological framework (the one provided by

![Figure 1: USDL-SEC customisation (only classes and properties used in this paper are depicted)](image1)

![Figure 2: DE ontological model](image2)
Linked USDL (which has recently attracted considerable research interest. In addition, the adoption of Linked USDL brings about the following advantages [14]: (i) Linked USDL relies on existing widely-used RDF(S) vocabularies (such as GoodRelations, FOAF and SKOS), whilst it can be easily extended through linking to further existing, or new, RDF(S) ontologies. In this respect, it promotes knowledge-sharing whilst it increases the interoperability, reusability and generality of our framework. (ii) By offering a number of different profiles, Linked USDL provides a holistic and generic solution able to adequately capture a wide range of business details. (iii) Linked USDL is designed to be easily extensible through linking to further existing, or new, RDF(S) ontologies. This is particularly important for our work as it facilitates seamless integration with the vocabularies outlined in Section III.

USDL-SEC provides a framework of concepts and properties for describing the security properties of an application. It includes the classes SecurityProfile, SecurityGoal, SecurityMechanism, and SecurityTechnology, along with a number of relevant object properties, as depicted in Figure 1 (to reduce notational clutter, we avoid prefixing the usdl-sec namespace to USDL-SEC classes and properties). A more elaborate account of these classes and properties can be found in [13].

At the highest level of abstraction, the DE policy model forms, essentially, a particular security profile to which a cloud application may adhere. In this respect, it takes the form of an instance of the SecurityProfile class, namely pbe:DEProfile (see Figure 1). A security profile is associated, through the object property hasSecurityGoal, with one or more security goals from the USDL-SEC class SecurityGoal. In the case of DE policies, the security goal is confidentiality. This is modelled in Figure 1 by associating the instance pbe:DEProfile with an instance, say pbe:DEGoal, of the Confidentiality class through the property hasSecurityGoal. The Confidentiality class forms a sub-concept of SecurityGoal. The confidentiality goal is achieved by means of a suitable data encryption mechanism. USDL-SEC provides the concept SecurityMechanism for the specification of such a mechanism. In particular, it provides the class Cryptography, a sub-concept of SecurityMechanism, an instance of which, say pbe:DEMechanism, represents the DE mechanism offered by our PaaS framework. This instance is associated with the pbe:DEGoal instance through the property isImplementedBy.

The DE mechanism represented by the instance pbe:DEMechanism is realised by means of an underlying concrete security technology. USDL-SEC provides the concept SecurityTechnology for the specification of such a technology. In our model, the DE mechanism is realised by the DE technology provided by our PaaS framework. This is modelled by introducing the pbe:PaaSDE subclass (see Figure 1), along with the instance pbe:DETechnology which represents this DE technology. This instance is associated with the DE mechanism through the property isRealizedByTechnology. The pbe:PaaSDE subclass is associated, through the property pbe:hasDEPolicySet, with the class pbe:DEPolicySet (the top concept of the DE policy model of Section IV.A). This essentially captures the fact that the DE mechanism is realised through the policies encompassed in one or more DE policy sets.

C. DFD Policy Model

The DFD policy model is derived in manner analogous to the one described in Sections IV.A and IV.B above. More specifically, the classes and properties of this model ontologically express the DFD rule template of Table 4.

Table 4: DFD rule template.

| [controlled object] is fragmented with [fragmentation scheme] | and distributed with [distribution scheme] |

This template comprises the attributes controlled object, fragmentation scheme and distribution scheme. The first attribute is the same as the one in the DE rule template. The second and third attributes identify, respectively, the data fragmentation and distribution schemes that are to be applied to the controlled object. They draw, respectively, their values from the classes pdfd:DFDObject and pdfd:DFDPolicySet of the Data Distribution and Encryption model outlined in Section III. DFD policies and policy sets are modelled analogously to DE policies and policy sets, i.e., as instances of the classes pdfd:DFDPolicy and pdfd:DFDPolicySet respectively.

Finally, the DFD model is incorporated into the ontological framework offered by USDL-SEC in a manner symmetric to the one described in Section IV.B for the DE model. More specifically, the Confidentiality class of Figure 1 is replaced by the Privacy class as now the security goal is privacy. Similarly, the Cryptography class is replaced by the class pdfd:DFD which accommodates the DFD mechanism which achieves this security goal. In addition, the classes pbe:PaaSDE and pbe:DEPolicySet are now replaced by the classes pdfd:PaaSDFD and pdfd:DFDPolicySet respectively to convey the fact that the DFD mechanism is realised through the policies encompassed in one or more DFD policy sets. For a more elaborate account of the DFD policy model, the interested reader is referred to [15].

V. RELATED WORK

A number of formalisms that advocate the use of markup languages for the declarative representation of policies have
been proposed [12], [16]. Although they succeed in disentangling the representation of policies from the code employed by an application for enforcing them, these formalisms lack any form of semantic agreement outside the confines of the organisations that created them. Any interoperability thus hinges on ad-hoc vocabularies that are shared by the various stakeholders that participate in an interaction. This inevitably brings about the following disadvantages: (i) it restricts the portability and reusability of policies; (ii) it hampers the determination of inter-policy relations; (iii) it leads to ad-hoc reasoning about policy compliance, which is perplexed with the particular vocabularies that are utilised for expressing the rules according to which the reasoning takes place; (iv) it hinders the performance of rule-based policy governance.

In order to overcome these disadvantages, a number of semantically-rich approaches to the specification of policies have been proposed [17], [18]. These generally embrace OWL for capturing the knowledge artefacts that underlie policies. In [17], the authors propose KAoS – a general-purpose framework for managing, monitoring and enforcing a wide range of policies. In [18], the authors propose POLICYTAB for facilitating trust negotiation in Semantic Web environments. The aforementioned semantically-enhanced approaches rely on bespoke, non-standards-based, ontologies for the representation of policies. This by definition restricts the generality, hence the portability and reusability of the policies that they represent. In contrast, our reliance on Linked USDL raises this restriction whilst bringing about the advantages outlined in Section IV.B.

VI. CONCLUSIONS AND FUTURE WORK

We have presented a novel Data Distribution and Encryption Model for PaaS-enabled Cloud Security. This model conceptualises all attributes that must be considered for designing and bootstrapping the fragmentation, distribution and encryption of sensitive data that are to be stored in cloud infrastructures. In order to avoid the use of bespoke ontologies for the representation of such knowledge, the model was expressed in Linked USDL’s SEC profile [13]. Our model will be used as part of the PaaSWord framework – an envisaged security-by-design framework that will provide storage protection mechanisms for increasing the trust of users on cloud applications [3]. The next steps of this work include the development of dedicated mechanisms that make use of this modelling framework for guiding a developer in defining a set of Data Access Object annotations in the persistence layer of cloud applications.

ACKNOWLEDGMENT

The research leading to these results has received funding from the EU’s H2020 programme under grant agreement No 644814, the PaaSWord project (www.paasword.eu).

REFERENCES