Data Distribution and Encryption Modelling for PaaS-enabled Cloud Security[1]

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 specifying effective security controls for sensitive data that are manipulated by 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 materialise 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.). Its second part outlines an adequate access control scheme, one which hinges upon policies that take into account the dynamic and heterogeneous nature of cloud environments. This mainly refers to context-aware access control and it has been thoroughly discussed in [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 application 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

978-1-5090-1445-3/16/$31.00 ©2016 IEEE (CloudSPD'16)

[1] © © 2016 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

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 I: Details of Data Distribution elements

Class Taxonomy Levels	Proper-ties	Description
Symmetric		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]).
	hasSymmetricKeySize	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)
	hasSymmetricBlockSize	This property associates Symmetric with a positive integer that expresses how long is the fixed length string of bits on which the cypher operates (e.g. 128 bits).
Asymmetric		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]).
	hasAsymmetricKeySize	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)
	hasCurve	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.
Hybrid		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.

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 I 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.

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 II.

TABLE II: DETAILS OF DATA FRAGMENTATION ELEMENTS

Class Taxonomy Levels		Properties	Description
Relational Data Fragmentation			This class refers to different types of possible fragmentation of a relational database.
		hasPrivacyConstraint	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).
	Horizontal Fragmentation		This class refers to the fragmentation of a relational database across its rows.
		hasFragRow	This property associates a Horizontal Fragmentation instance with a non-negative integer that denotes the row number on the basis of which a Table should be fragmented.
	Vertical Fragmentation		This class refers to the fragmentation of a relational database across its columns.
		hasFragColumn	This property associates a Vertical Fragmentation instance with a string that denotes the column based on which a Table should be fragmented.
	Mixed Fragmentation		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.
		hasFragRow	This property associates Horizontal Fragmentation with a non-negative integer that denotes the row based on which a Table should be fragmented.
		hasFragColumn	This property associates Vertical Fragmentation with a string that denotes the column based on which a Table should be fragmented.
non-Relational Data Fragmentation			This Class refers to different types of possible fragmentation of a non-relational database [11].
	Sharding		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 [11].
	Replication		Replication class refers to the details of copying data/aggregates across multiple servers, so each bit of data can be found in multiple places [11].
	Master-slave replication		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 [11].
	Peer-to-peer replication		Peer-to-peer subclass (of Replication) refers to the details of replication that allow writes to several nodes and synchronization between them [11].

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 and strength 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 in order to protect its privacy. (iii) Access Control policies that determine when to permit, or deny, access to sensitive data by taking into account a set of contextual attributes linked with the actor requesting the access. Both the DE and DFD policies entail security controls that are enforceable during the bootstrapping phase of a cloud application. On the other hand, Access Control policies entail security controls that are enforceable during application execution time.

In order to facilitate application developers in articulating *effective* security policies, our PaaS solution is underlain – for each supported policy kind – by an *ontological model* that

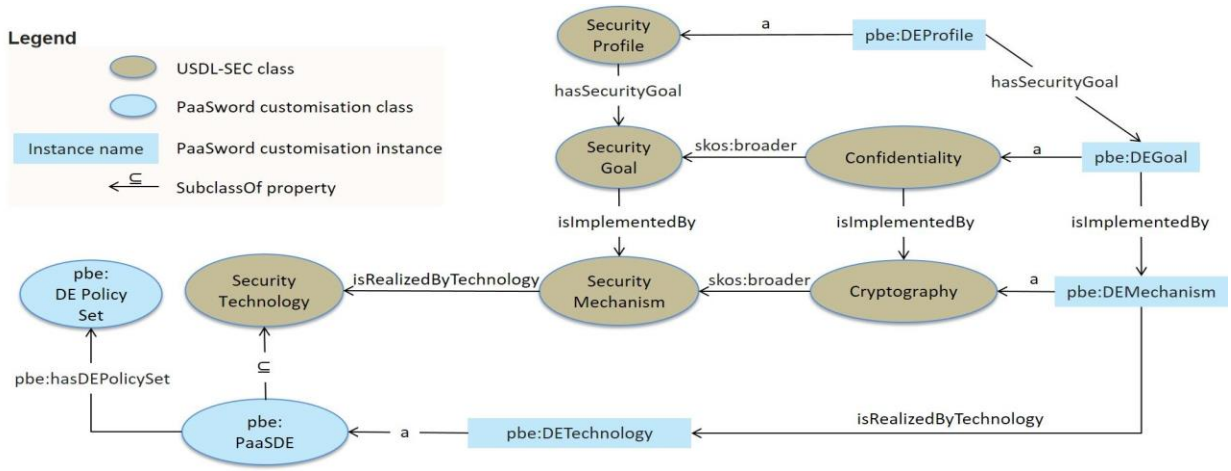

Fig. 1: USDL-SEC customisation (only classes and properties used in this paper are depicted)

exhibits the following characteristics: (i) It is based on a framework of pertinent concepts, and their interrelations, that capture the knowledge artefacts that are required for specifying 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, one that unravels the representation of a policy from the actual code that is employed for enforcing it, bringing about the following benefits: (i) It permits the extension and instantiation of the concepts and properties of the aforementioned framework independently of the code of the application. The aim of such an extension/instantiation is to ground the framework to the particular needs of a given application. (ii) It formulates an adequate basis for reasoning generically, i.e. orthogonally to the code of the application, about the correctness of the policies, hence about the effectiveness of the security controls that these policies 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 [4].

A. DE Policy Model

1) DE Policy Rules. Following the XACML standard [12], a DE policy comprises one or more *rules* that may be abstractly described in terms of the template of Table III.

TABLE III: DE RULE TEMPLATE

[controlled object**]** is encrypted with [cryptographic type]

This template defines the structure, in terms of relevant attributes, to which all DE rules in our PaaS framework adhere. The first attribute, namely *controlled object*, identifies the sensitive data object which is to be protected. Its values are drawn 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:CryptoType` 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 Fig. 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 linked to the rules that it comprises via the `pbe:hasDERule` property. Inspired by the XACML standard, DE policies are grouped into *policy sets*. A policy set is modelled as an instance of the class `pbe:PolicySet` (see Fig. 2). A policy is associated with its encompassing policy set by means of the `pbe:belongsToPolicySet` property. A policy set may hierarchically comprise one or more other DE policy sets. As depicted in Fig. 2, this is captured by rendering the `pbe:belongsToPolicySet` property applicable to DE policy sets too.

B. DE Policies in Linked USDL

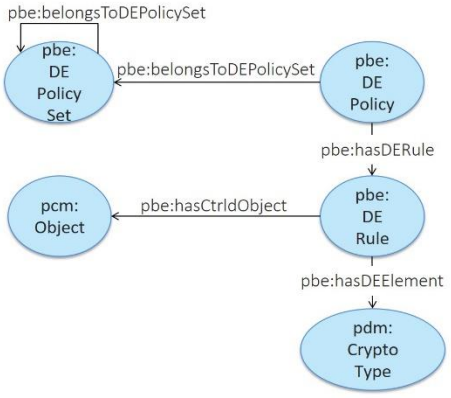

Fig. 2: DE ontological model

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 USDL-SEC [13] – Linked USDL's security profile. By drawing upon Linked USDL, we avoid the use of custom ontologies for capturing data encryption policies (see Section V for a brief review of such ontologies). Instead, our approach is founded on a diffused and lightweight ontological framework. Moreover, the adoption of Linked USDL offers the following benefits [14]: (i) Linked USDL is based on widely-used RDF(S) vocabularies (e.g. FOAF, GoodRelations, SKOS) and can be easily extended to include other relevant existing, or new, ontologies. It therefore promotes knowledge-sharing and increases the reusability, interoperability and generality of our approach. This is particularly important for our work as it facilitates seamless integration with the vocabularies outlined Section III. (ii) Through the different profiles that it encompasses, Linked USDL provides a holistic and generic solution able to accommodate a wide range of relevant business details in.

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 Fig. 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].

The DE policy model constitutes itself a particular security profile to which a cloud application may adhere. In this respect, it is formulated as an instance of the `SecurityProfile` class, namely `pbe:DEProfile` (see Fig. 1). A security profile is linked, by virtue of the object property `hasSecurityGoal`, with a security goal that forms an instance of the `SecurityGoal` class. For DE policies, the security goal is *confidentiality*. This is depicted in Fig. 1 by relating the instance `pbe:DEProfile` with a particular instance (in this case `pbe:DEGoal`) of the `Confifentiality` class via the property `hasSecurityGoal`. The `Confidentiality` class is a sub-concept of `SecurityGoal`. Confidentiality is achieved through a suitable data encryption mechanism. To this end, USDL-SEC provides the concept `SecurityMechanism` which represents such a mechanism. More specifically, it offers the class `Cryptography` (which is a sub-concept of `SecurityMechanism`), an instance of which (in this case `pbe:DEMechanism`), represents the DE mechanism that our PaaS framework provides. This instance is linked to the `pbe:DEGoal` instance by virtue of the property `isImplementedBy`.

The DE mechanism modelled by the instance `pbe:DEMechanism` is implemented by a concrete underlying security technology. USDL-SEC provides the concept `SecurityTechnology` for the specification of such a technology. In our approach, the DE mechanism is implemented by the DE technology offered by the proposed PaaS framework. This is captured by the introduction of the `pbe:PaaSDE` subclass (see Fig. 1) and the instance `pbe:DETechnology` which models this DE technology. This instance is linked to the DE mechanism via the property `isRealizedByTechnology`. The `pbe:PaaSDE` subclass is related to the class `pbe:DEPolicySet` via the `pbe:hasDEPoliceSet` property. This essentially signifies 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 a 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 IV.

TABLE IV: 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 `pdm:DataFragmentation` and `pdm:DataDistribution` 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 `pdfd:DFDPolicy` and `pdfd:DFDPolicySet` classes 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 Fig. 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 that 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 utilises the policies included in the DFD policy sets. A more elaborate account of the DFD policy model can be found in [15].

V. Related Work

A number of formalisms that advocate the use of markup languages for expressing policies have been proposed [12], [16]. Although they succeed in disentangling the representation of policies from the code used by an application for enforcing them, these formalisms lack any semantic agreement beyond the organisations that created them. Any interoperability thus hinges on custom vocabularies shared between 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 potential relations between policies (e.g. contradicting policies, subsuming policies, etc.); (iii) it leads to reasoning about policy compliance that is dependent upon the particular vocabularies in which the rules of the reasoning are expressed; (iv) it hinders the performance of rule-based policy governance. In order to overcome these disadvantages, the works in [17] and [18] propose semantically-rich approaches to the representation of policies. These generally embrace ontologies 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 custom ontologies for capturing these attributes, the model was founded upon the USDL-SEC profile [13]. Our model forms part of the PaaSword framework – a security-by-design framework that aspires to develop security mechanisms for increasing the trustworthiness of cloud applications [3]. The next steps of this work include the development of dedicated mechanisms that make use of this modelling framework in order to aid developers in defining suitable 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

[1] Group, T. T., 2013. The Notorious Nine. Cloud Computing Top Threats in 2013. Cloud Security Aliance (CSA).

[2] The Notorious Nine Cloud Computing Top Threats in 2013, Cloud Security Alliance, 2013.

[3] Y. Verginadis, A. Michalas, P. Gouvas, G. Schiefer, G. Hübsch and I. Paraskakis, "PaaSword: A Holistic Data Privacy and Security by Design Framework for Cloud Services," in *Proceedings of the 5th International Conference on Cloud Computing and Services Science (CLOSER 2015)*, Lisbon, Portugal, 2015.

[4] S. Veloudis, Y. Verginadis, I. Patiniotakis, I. Paraskakis and G. Mentzas. Context-aware Security Models for PaaS-enabled Access Control. *6th International Conference on Cloud Computing and Services Science (CLOSER 2016)*, Rome, Italy, April 23-25, 2016.

[5] NIST Computer Security Division's (CSD) Security Technology Group (STG) (2013). "Block cipher modes". Cryptographic Toolkit. NIST. Retrieved April 12, 2013.

[6] Daemen, Joan; Rijmen, Vincent (March 9, 2003). "AES Proposal: Rijndael" (PDF). National Institute of Standards and Technology. p. 1. Retrieved 21 February 2013.

[7] P. Prasithsangaree & P. Krishnamurthy (2003). "Analysis of Energy Consumption of RC4 and AES Algorithms in Wireless LANs" (PDF). Archived from the original (PDF) on 3 December 2013.

[8] Diffie, W.; Hellman, M.E. (November 1976). "New directions in cryptography". IEEE Transactions on Information Theory 22 (6): 644–654. doi:10.1109/TIT.1976.1055638. ISSN 0018-9448.

[9] Koblitz, N. (1987). "Elliptic curve cryptosystems". Mathematics of Computation 48 (177): 203–209. doi:10.2307/2007884. JSTOR 2007884.

[10] Zimmermann, Philip (1995). PGP Source Code and Internals. MIT Press. ISBN 0-262-24039-4.

[11] Sadalage, P. J., Fowler, M. (2012). NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley.

[12] eXtensible Access Control Markup Language (XACML) Version 3.0. 22 January 2013. OASIS Standard. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

[13] Linked USDL, http://www.linked-usdl.org/

[14] Cardoso, J., Pedrinaci, C., Leidig, T.: Linked USDL: a Vocabulary for Web-scale Service Trading. *In 11th Extended Semantic Web Conference (ESWC)* (2014)

[15] PaaSword Deliverable 2.2. https://www.paasword.eu/deliverables/

[16] Damianou, N., Dulay, N., Lupu, E., Sloman, M.: The Ponder Policy Specification Language. In Sloman, M., Lobo, J., Lupu, E. (eds.) *Proceedings of the International Workshop on Policies for Distributed Systems and Networks* (POLICY '01), pp. 18--38, Springer-Verlag, London (2000)

[17] Uszok, A., Bradshaw, J., Jeffers, R., Johnson, M., Tate, A., Dalton, J., and Aitken, S.: KAoS Policy Management for Semantic Web Services. IEEE Intel. Sys. 19, 4, 32--41 (2004)

[18] Nejdl, W., Olmedilla, D., Winslett, M, Zhang. C.C.: Ontology-Based policy specification and management. In Gómez-Pérez, A. and Euzenat, J. (eds.) ESWC'05, pp. 290-302, Springer-Verlag, Berlin, Heidelberg (2005)