Flexible and Dynamic Consent-Capturing

Muhammad Rizwan Asghar[1,2] and Giovanni Russello[1]

[1] Create-Net, Italy
{asghar,russello}@create-net.org
[2] University of Trento, Italy

Abstract. Data usage is of great concern for a user owning the data. Users want assurance that their personal data will be fairly used for the purposes for which they have provided their consent. Moreover, they should be able to withdraw their consent once they want. Actually, consent is captured as a matter of legal record that can be used as legal evidence. It restricts the use and dissemination of information. The separation of consent capturing from the access control enforcement mechanism may help a user to autonomously define the consent evaluation functionality, necessary for the automation of consent decision. In this paper, we present a solution that addresses how to capture, store, evaluate and withdraw consent. The proposed solution preserves integrity of consent, essential to provide a digital evidence for legal proceedings. Furthermore, it accommodates emergency situations when users cannot provide their consent.

Keywords: Consent Management, Consent Evaluation, Consent Automation, Consent-Capturing, Consent-Based Access Control

1 Introduction

Data usage is of great concern for a user owning the data. Users want assurance that their personal data will be fairly used for the purposes for which they have provided their consent. Moreover, they should be able to withdraw their consent once they want. Actually, consent is captured as a matter of legal record that can be used as legal evidence. It restricts the use and dissemination of information which is controlled by the user owning the data. For the usage of personal data, organisations need to obtain user consent, strictly regulated by the legislation. This forces organisations to implement not only the organisational compliance rules but also the legislation rules to access the user data. Currently, the law enforcement agencies are forcing organisations to collect users consent every time their data is accessed where users can decide not to provide consent or they may withdraw their consent any time they want[3]. In the electronic environment, consent can be determined by the automatic process without the explicit involvement of the user owning the data, also known as a *data subject*. This enables enormous amount of data to be processed in a automated and faster manner where consent is captured at the runtime.

[3] http://www.bloomberg.com/news/2011-03-16/facebook-google-must-obey-eu-data-protection-law-reding-says.html

1.1 Motivation

The motivation behind consent-capturing is from the real-world, where a user intervention is reduced as much as possible. Let us consider the healthcare scenario where a patient provides his/her written consent to the hospital. Later on, the hospital staff refers to this written consent in order to provide access to patient's records. For instance, if a nurse needs to access patient's record, she needs first to make it sure that she has access and patient has provided his/her consent for a nurse. In the healthcare scenario, we can realise access controls at two different levels. At the first level, the access controls are enforced by the care/service provider while at the second level, it is enforced by the patient, where a patient provides his/her consent to the first level in order her data to be accessed. In short, consent is a mean to control the access on the personal data. There are two obvious questions which needs to be addressed when we capture the notion of consent in an automated manner. First, how to capture and store the consent. Second, how to evaluate consent from the written consent.

Unfortunately, traditional access control techniques, such as Role-Based Access Control (RBAC) [12], fail to capture consent. However, there are only a few access control techniques [1, 7] that can capture consent but they tightly couple the access control enforcement mechanism with the consent-capturing. The separation of consent-capturing from the enforcement mechanism may help a user to autonomously define the consent evaluation functionality, necessary for regulating the automation of consent decision. The main drawback of state-of-the-art consent-capturing schemes [4, 9, 10, 14] is that consent-capturing mechanism is too rigid as they consider the consent with a predefined set of attributes and it is not possible to take the contextual information into account in order to provide consent. This contextual information may include time, location or other information about the requester, who makes an access request. In short, the existing consent-capturing mechanisms are not expressive enough to handle the real-world situations. Moreover, the existing access control techniques do not address how to ensure transparent auditing while capturing the consent. The transparent auditing could be required for providing digital evidence in the court.

1.2 Research Contributions

In this paper, we present how to capture, store and evaluate consent from the written consent. We consider the written consent as a consent-policy where a data subject indicates who is permitted to access his/her data. In the proposed solution, the consent evaluation functionality can be delegated to a third party. The advantage of this delegation is to separate the access control enforcement mechanism from the consent-capturing. This research is a step towards the automation of the consent-capturing. The automation do not only captures consent dynamically but also increases efficiency as compared to providing consent requiring data subject's intervention. Moreover, the proposed solution enables a data subject to withdraw his/her consent. Moreover, it treats emergency situations when a data subject cannot provide his/her consent. Last but not least, the integrity of consent is preserved and a log is maintained by system entities to provide a digital evidence for legal proceedings.

1.3 Organisation

The rest of this paper is organised as follows: Section 2 describes the legal requirements in order to capture consent. Section 3 reviews the related work. Section 4 presents the proposed solution. Section 5 focuses on the solution details. We follow with a discussion about availability, confidentiality and increased usability in Section 6. Finally, Section 7 concludes this paper and gives directions for the future work.

2 Legal Requirements to Capture Consent

Consent is an individual's right. In fact, consent can be regarded one's wish to provide access on one's personal information. Legally, a data subject should be able to provide, modify or withhold statements expressing consent. The given consent should be retained for the digital evidence. Generally, the paper-based consent is considered valid once signed by the data subject. In some countries, specific legislation may require the digital consent to be signed using the digital signature. In other words, an electronically signed consent can be considered equivalent to the manually signed paper-based consent.

According to article 2(h) of the EU Data Protection Directive (DPD) [5], consent is defined as: *"'the data subject's consent' shall mean any freely given specific and informed indication of his wishes by which the data subject signifies his agreement to personal data relating to him being processed.*" This definition clearly indicates three conditions, i.e., the consent must be *freely given*, *specific* and *informed*.

- **Freely given:** A consent can be considered free if captured/provided in the absence of any coercion. In other words, it is a voluntary decision of a data subject.
- **Specific:** A consent can be considered specific when it is captured/provided for a dedicated purpose. Moreover, one consent is for one action which is specified to the data subject. Therefore, one consent cannot be used for any other purposes.
- **Informed:** The consent is not considered valid until a data subject is provided with the necessary information. The data subject must be provided with information to understand all benefits and drawbacks of giving and not giving consent. This information must be accurate and given in a transparent, clear and understandable manner.

In legal terms, consent is often a positive response. However, a data subject can express a negative response as it can be regarded as the right of a data subject to express his/her desire for not sharing a certain piece of data.

3 Related Work

In RBAC [12], each system user is assigned a role and a set of permissions are granted to that role. A user can get access on data based on his/her role. RBAC is motivated by the fact that users in the real-world make the decisions based on their job functions within an organisation. The major drawback of RBAC is that it does not take into considerations the user consent for providing the access.

In the British Medical Association (BMA) policy model [1], access privileges for each medical record are defined in the form of Access Control Lists (ACLs) that are managed and updated by a clinician. The main goal of the BMA model is to capture consent, preventing multiple people in obtaining access to large databases of identifiable records. The shortcoming is that the ACLs are not flexible and expressive enough for defining the access. Moreover, the consent structure is not discussed. In Provision-Based Access Control (PBAC) [7], access decisions are expressed as a sequence of provisional actions instead of simple permit or deny statements. The user consent can be captured if stated within the access policy. In both techniques [1] and [7], the consent-capturing mechanism is tightly coupled with the access control enforcement. This restricts the possibility of consent-capturing in an automated manner.

Cassandra [3], a role-based trust management language and system for expressing authorisation policy in healthcare systems, captures the notion of consent as a special role to inform the user that his/her data is being accessed. Cassandra has been used for enforcing the policies of the UK national Electronic Heath Record (EHR) system. The requirement of consent-capturing notion as a special role adds an extra workload because in most of the situations, the consent can be implicitly derived if a user has allowed the system to do so.

Usage-based access control [16] aims to provide dynamic and fine-grained access control. In the usage-based access control, a policy is defined. This policy is based on attributes of the subject, target and environment. The attributes are continuously monitored and updated. If needed, a policy can be enforced repeatedly during a usage session. Attributes can be used for capturing information related to the consent while the access session is still active. If the information changes, such as the user revokes the consent, the access session is terminated. Russello *et al.* [11] propose a framework for capturing the context in which data is being accessed. Both approaches [16] and [11] have major limitation of tightly coupling the access control enforcement mechanism with the consent capturing.

Ruan and Varadharajan [10] present an authorisation model for handling e-consent in healthcare applications. Their model supports consent delegation, denial and consent inheritance. However, it is not possible to capture the contextual information in order to provide the consent. In e-Consent [4], the consent can be identified in four different forms including, *general consent, general consent with specific denial, general denial with specific consent* and *general denials*. They provide some basics of consent and associated security services. They suggest to store consent in the database. However, it is not clear how they can express the consent rules and evaluate the request against those rules in order to provide the consent. O'Keefe, Greenfield and Goodchild [9] propose an e-Consent system that captures, grants and withholds consent for access to manage electronic health information. Unfortunately, the consent-capturing mechanism is static, restricting the possibility of expressive consent rules.

Jin *et al.* [6] proposes an authorisation framework for sharing EHR, enabling a patient to control his/her medical data. They consider three types of consents, i.e., *break-glass consent*, *patient consent* and *default consent*. The *break-glass consent* has the highest priority, representing emergency situations, while the *default consent* has the lowest one. The *patient consent* is captured if no *default consent* is provided. They

store consent in the database. Unfortunately, the consent-capturing mechanism is not expressive enough to define the real-world consent rules.

Verhenneman [13] provides a discussion about the legal theory on consent and describes the lifecycle of consent. The author provides a legal analysis in order to capture consent. However, the work is theoretical without any concrete solution. Wuyts *et al*. [14] propose an architecture to integrate patient consent in e-health access control. They capture consent as Policy Information Point (PIP), where consent is stored in the database. The shortcoming of storing consent in the database is that it limits the data subject to rely on only pre-defined of attributes with a very limit expressivity to define the consent rules. While in our proposed solution, we do not consider a fix set of pre-defined attributes, instead we dynamically capture the attributes in the form of contextual information.

The existing research on access control lacks in providing a user to autonomously define the consent evaluation functionality, necessary for regulating the automatic collection of consent. That is, it requires investigation how to capture consent independent of the access control enforcement mechanism. Moreover, it is not clear how to provide transparent auditing while giving (or obtaining) the consent so that later on a forensic analysis can be performed by an investigating auditor.

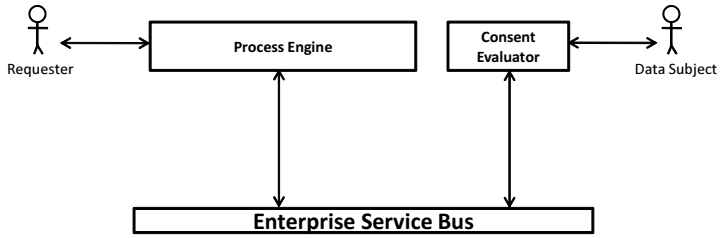


Fig. 1. System architecture of the proposed solution

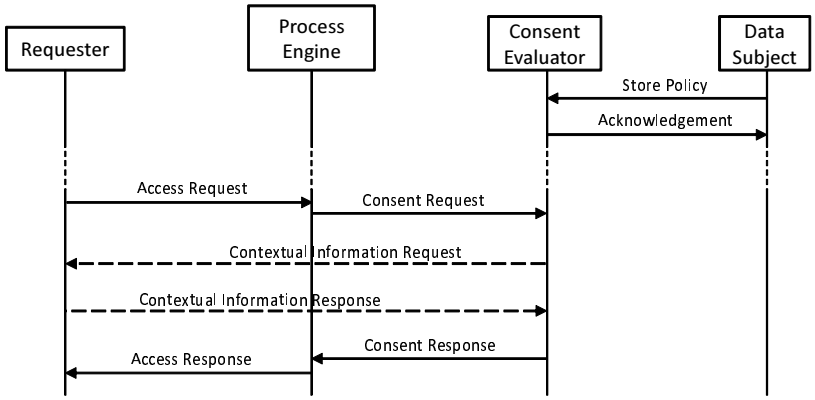
4 The Proposed Approach

Before presenting the details of the proposed solution, it is necessary to discuss the system model which is described as follows:

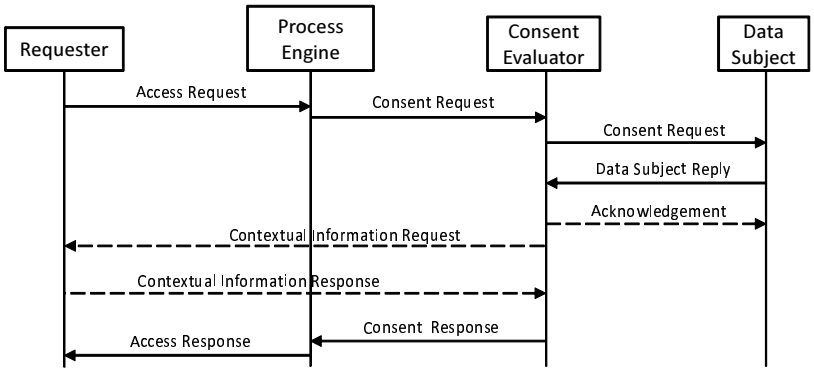
4.1 System Model

In this section, we identify the following system entities:

– **Data Subject:** Data Subject is the user whose consent is being captured. A Data Subject interacts with the Consent Evaluator to manage consent-policy corresponding to a resource. Furthermore, a Data Subject interacts with the Requester using the enterprise service bus to collect the contextual information.



(a) when a consent-policy is already stored



(b) when a data subject replies dynamically

Fig. 2. Capturing consent in the proposed solution

- **Requester:** A Requester is a user that sends access request to the Process Engine. If required, a Requester sends contextual information to the Data Subject. This information may include, but not limited to, Requester's role, Requester's name, Requester's location, time, etc.
- **Process Engine:** It is the data controller who is responsible to enforce access controls. It works as a bridge between a Requester and a Data Subject. Once required, it sends consent request to the Consent Evaluator and receives back the consent.
- **Consent Evaluator:** The Consent Evaluator is an entity responsible to manage and evaluate consent-policy. When requested, it provides consent back to the Process Engine. This is an additional entity that is not present in the traditional access control systems.

The main idea is to have two levels of access controls. The first level of access controls to define access control policies which could be defined by the Data Subject, the data controller or both. While, the second level of access controls are defined by the

Data Subject in order to authorise someone to get his/her consent in an automated manner. The first level of access controls express either consent is required or not. The second level of access controls are managed by the Data Subject. In other words, each Data Subject defines consent-policies in order to indicate who is authorised to get his/her consent. Moreover, a Data Subject may withdraw his/her consent by deleting the corresponding consent-policy. In this paper, our main focus is to elaborate the second level of access controls.

Figure 1 shows the abstract architecture of the proposed solution. The Process Engine is responsible to handle the first level of access controls defining who can access the resource while the Consent Evaluator is responsible to capture and store the consent of a Data Subject. The Consent Evaluator is maintained for each Data Subject. Once the Process Engine identifies that consent is required, it interacts with the Consent Evaluator via the enterprise service bus.

Figure 2 illustrates how system entities interact with each other. We can distinguish between two cases for capturing consent. The first case is when consent can be stored statically while the second case is when consent is captured dynamically.

In the first case, shown in Figure 2(a), a Data Subject stores the consent-policy and gets back an acknowledgment. Once a Requester sends an access request, the Process Engine identifies if the access control policy corresponding to the requested resource requires any consent. If the Data Subject consent is required, the Process Engine generates and sends the consent request to the Consent Evaluator. Since the consent-policy is already stored, the Consent Evaluator does not need to interact with the Data Subject. However, the Consent Evaluator may collect contextual information from the Requester in order to evaluate the consent-policy. After the consent-policy has been evaluated, consent is sent from the Consent Evaluator to the Process Engine. Finally, the Process Engine evaluates the access policy and sends the access response back to the Requester.

In the second case, shown in Figure 2(b), we assume that the consent-policy is not already stored as we considered in the above case. Once a Requester makes an access request, the Process Engine identifies if the access control policy corresponding to the requested resource requires any consent. If the Data Subject consent is required, the Process Engine generates and sends the consent request to the Consent Evaluator. Since the consent-policy is not stored, the Consent Evaluator needs to interact with the Data Subject. For this purpose, the Consent Evaluator forwards the consent request to the Data Subject. The Data Subject may reply with consent, consent-policy or both. In case if the Data Subject reply includes the consent-policy, the Consent Evaluator sends an acknowledgment back to the Data Subject; moreover, the Consent Evaluator may collect contextual information from the Requester in order to evaluate the consent-policy. After the consent-policy has been evaluated, consent is sent from the Consent Evaluator to the Process Engine. However, if the Data Subject reply includes consent, then the Consent Evaluator does on to do evaluation. Once the Process Engine receives consent, it evaluates the access control policy and sends the access response back to the Requester.

In the proposed solution, it is possible to provide a Data Subject with a tool for the consent management; this not only enables the administration of the consent-policy but also supports the inspection service to check who has obtained consent automatically.

In other words, the system maintains a log at the Consent Evaluator side to facilitate a Data Subject with both the administration of consent-policies and inspection of consent log. The consent log may be provided as digital evidence.

4.2 Consent-Policy Types

The consent-policy in the proposed solution refers to the written consent. When a Requester needs to verify whether he or she has consent in order to access the data, he or she refers to the written consent, which is consent-policy in the proposed solution. The purpose of the Consent Evaluator is to store the consent-policy and evaluate the consent decision, whenever requested.

The consent-policy may require a set of attributes in order to evaluate consent. The consent-policy attributes include, but not limited to, the following:

- Request date
- Request time
- Requester name
- Requester role
- Requester location
- Requester age
- Access reason
- Access specification

There are two types of consent-policy.

Open Policy The open policy can be categorised further into two types. One is *black-list* while the other is *white-list*. The black-list associated with an attribute limits access to a resource for Requesters holding that attribute with values in the list. For instance, in Table 1, a black-list of Requesters (i.e., Requester name attribute) is maintained to limit the access on resource R_1. On the other hand, the white-list associated with an attribute permits access to a resource only for Requesters holding that attribute with values in the list. For instance, in Table 1, a white-list of Requesters (i.e., requester role attribute) is maintained to permit access to resource R_2.

Complex Policy The complex policy is the one that may involve conditional expressions in order to provide consent. Typically, a conditional expression is evaluated against the Requester attributes. These conditional expressions are evaluated by the Consent Evaluator. If the Requester attributes satisfy all the conditional expressions in the consent-policy, the consent is *Yes* and *No* otherwise. For instance, in Table 1, a Requester can access to resource R_5 if the time of request is between 8:00 hrs and 17:00 hrs, and the Requester is located in HR-ward.

Table 1. Consent-policy storage at the consent evaluator side

Resource ID	Consent-Policy Type	Contextual Information	Policy Description
R_1	Open	Requester-name	Black-list: {Alice, Bob, Charlie}
R_2	Open	Requester-role	White-list: {Nurse, Doctor}
R_3	Complex	Request-time	Condition: 8:00 hrs $\leq$ Time $\leq$ 17:00 hrs
R_4	Complex	Requester-location	Condition: Location=HR-Ward
R_5	Complex	Request-time, Requester-location	Condition: 8:00 hrs $\leq$ Time $\leq$ 17:00 hrs AND Location=HR-Ward
$\vdots$	$\vdots$	$\vdots$	$\vdots$

5 Solution Details

For each of the resource, requiring consent of the Data Subject, the Consent Evaluator stores the corresponding consent-policy. Table 1 illustrates how consent-policies are stored at the Consent Evaluator. Each row in Table 1 corresponds to the consent-policy per resource. For each resource, the Consent Evaluator stores the consent-policy type (that is, open or complex), parameters (that is, contextual information) required in order to evaluate the consent-policy and the description providing further details of the consent-policy. In case of the open type, the policy description provides the information about the type of access list, either black-list or white-list. While in case of the complex type, the policy description expresses the conditional expressions required to be fulfilled in order to provide consent as *Yes*.

5.1 Communication Messages

For both static and dynamic consent-capturing, the detail of each message, shown in Figure 2, is described as follows:

Store Policy When a Data Subject desires to store his/her consent, he/she sends *Store Policy* message to the Consent Evaluator. Each Data Subject has his/her own Consent Evaluator. This message includes resource ID, consent-policy type, a list of attributes used in the consent-policy and the consent-policy. Upon receiving this message, a Consent Evaluator stores this information, as shown in Table 1.

Acknowledgment We can distinguish between two different cases which are based on how the consent-policy is stored. If the consent-policy is stored statically, then after the *Store Policy* message, the *Acknowledgment* message is sent back to th Data Subject. In case if the consent-policy is stored dynamically then it is sent after the *Data Subject Reply*. In both cases, if the consent-policy is stored successfully then the *Acknowledgment* will include *OK*. Otherwise, it will include the error message with the error details.

Access Request A Requester sends the *Access Request* to the Process Engine in order to request access to the resource. The *Access Request* includes Requester's URI, target resource ID, the access operation, date and time.

Consent Request Whenever the Process Engine identifies in the access control policy that the Data Subject consent is required, it sends the *Consent Request* to the Consent Evaluator. The *Consent Request* contains the Requester's URI, target resource ID, the access operation, date and time. Here, we can see that both the *Access Request* and the *Consent Request* contain the same information, since the consent-policy is managed by the Data Subject while the access control policy does not necessarily need to be managed by the Data Subject.

Data Subject Reply In case of dynamic consent capturing, this message is sent from the Data Subject to the Consent Evaluator in response to the Consent Request. It may contain the consent response, the consent-policy or both. The Consent Evaluator takes actions based on this message. That is, the Consent Evaluator either forwards the consent response or evaluates the consent-policy. In case if it contains both the consent response and the consent-policy then the evaluation can be skipped. In case if a Data Subject replies with consent then it may be accomplished synchronously or asynchronously using a handheld device such as PDA or mobile device. Alternatively, consent can also be provided by email.

Contextual Information Request After the consent-policy has been found against the requested resource, the Consent Evaluator needs to collect the contextual information by sending the *Contextual Information Request* to the Requester that is identified by his/her URI. The *Contextual Information Request* contains all the parameters required for the consent-policy corresponding to the requested resource.

Contextual Information Response The Requester replies the *Contextual Information Request* with the *Contextual Information Response*. The *Contextual Information Response* contains all the parameters requested in the *Contextual Information Request*. The contextual information may be collected in multiple round trips of request and response.

Consent Response After the Consent Evaluator has evaluated the consent-policy against the contextual information, the *Consent Response* is sent back to the Process Engine. In case if the contextual information of the Requester satisfies the consent-policy corresponding to the requested resource, the *Consent Response* contains the consent. Otherwise, it contains an error message.

Access Response Finally, the Process Engine sends the *Access Response* to the Requester. For the successful access response, it is necessary that the Consent Evaluator replies with the required consent.

5.2 Consent Withdrawal

In the proposed architecture, a Data Subject may withdraw his/her consent any time he/she wants. This can be accomplished by deleting the consent-policy stored on the Consent Evaluator. In other words, the resource entry will be deleted from Table 1 by the Data Subject. This maps well to the real-world situations. Consider the healthcare scenario where a patient has provided his/her consent in order to provide access on his/her medical data for the research purpose. Every time the access is made, the consent

will be checked. Let suppose that the patient calls the care-provider to withdraw his/her consent. After the withdrawal, the patient medical data cannot be accessed anymore. The same is the situation in the proposed solution. After a Data Subject has defined the consent-policy, consent can be provided by the Consent Evaluator. However, once the Data Subject wants to withdraw, the consent-policy is deleted by the Consent Evaluator. After the consent-policy has been deleted, no consent can be provided until the Data Subject provides the new consent or the consent-policy.

5.3 Integrity of Consent

Let us assume that the PKI is already in place, where each entity, including the Requesters and Data Subjects, has a private-public key pair. For providing the integrity, the consent-policy can be signed with the signing (or private) key of the Data Subject while the contextual information is signed with the signing key of the Requester. Once a Requester provides the contextual information, first an integrity check is performed to verify if the information is not altered or forged by an adversary. In case if a dispute occurs, the log of entities can be inspected in order to investigate the matter. Technically, the signature will be checked on the transmitted data. The digital signature not only guarantees integrity but also ensures non-repudiation and unforgeability of consent.

5.4 Emergency Situations

In case of emergency, the *Emergency Response Team* may provide consent on the behalf of the Data Subject. Let us consider the healthcare scenario where a patient is in emergency condition, such as heart-attack or some similar situation. In this situation, if a doctor needs any consent for which the patient has not defined any consent-policy then the *Emergency Response Team* or a legal guardian may provide consent on the behalf of the patient. Moreover, just like the paper-based consent, the consent-policy of minors or one who is mentally incapable can be provided by a legal guardian.

5.5 Digital Evidence

Once consent is requested by a Process Engine, the request should be logged. Not only this but also the Consent Evaluation should log once the consent is provided to the Process Engine. Since the contextual information is provided by a Requester to the Consent Evaluator, a Requester should also log the information requested by a Consent Evaluator. This would prevent repudiation of all the involved entities. These logs can be provided to the court as digital evidence.

5.6 Data Subject Tool

The proposed solution may provide a Data Subject with a tool to manage the consent-policies. Furthermore, a Data Subject may be provided with an inspection tool for observing who has obtained consent in an automated manner.

5.7 Implementation Overview

For evaluating the consent-policy against the Requester's attributes, we may consider the widely accepted policy-based framework proposed by IETF [15], where the Consent Evaluator manages the Policy Enforcement Point (PEP) (only the for the second level of access controls defined in the consent-policy) in order to provide the consent. The Consent Evaluator also manages Policy Decision Point (PDP) for making decision about the consent either *Yes* or *No*. The consent-policy store can be realised as the Policy Administration Point (PAP). In the proposed solution, the request is sent by the Process Engine while the Requester can be treated as a PIP. For representing the consent-policy, we may consider XACML policy language proposed by OASIS [8].

5.8 Performance Overhead

The performance overhead of the proposed solution is $O(m+n)$, where m number of conditional predicates in the consent-policy while n is the number contextual attributes required to evaluate the consent-policy in order to provide consent.

6 Discussion

This section provides a discussion about how the proposed solution may provide security properties including availability and confidentiality. Moreover, this section also gives a brief overview about how to increase the usability for a Data Subject.

6.1 Availability and Confidentiality

For providing availability, the Consent Evaluator can be considered in the outsourced environment, such as the cloud service provider. If the Consent Evaluator is managed by a third party service provider, then there is a threat of information leakage about the consent-policy or the contextual information. In order to provide confidentiality to the consent-capturing in outsourced environments, we may consider ESPOON (Encrypted Security Policies in Outsourced Environments) proposed in [2].

6.2 Increased Usability

In order to increase usability for defining a consent-policy, a Data Subject can be provided with a drag-and-drop policy definition tool for a pre-defined set of parameters of the contextual information. Moreover, a full-fledged pre-defined set of consent-policies can also be provided to the Data Subject.

7 Conclusions and Future Work

In this paper, we have proposed an architecture capture and manage consent. The proposed architecture enables the data controller to capture the consent in an automated manner. Furthermore, consent can be withdrawn any time a Data Subject wishes to do so. In the future, we are planning to implement and evaluate the performance overhead incurred by the proposed solution. In case if a Data Subject associates multiple consent-policies with a resource, then consent resolution strategies needs to investigated.

Acknowledgment

This work is supported by the EU FP7 programme, Research Grant 257063 (project Endorse).

References

1. R.J. Anderson. A security policy model for clinical information systems. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pages 30 –43, May 1996.
2. Muhammad Rizwan Asghar, Mihaela Ion, Giovanni Russello, and Bruno Crispo. ESPOON: Enforcing encrypted security policies in outsourced environments. In *The Sixth International Conference on Availability, Reliability and Security*, ARES'11, 2011.
3. M.Y. Becker and P. Sewell. Cassandra: distributed access control policies with tunable expressiveness. In *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*, pages 159 – 168, 2004.
4. Enrico Coiera and Roger Clarke. e-Consent: The design and implementation of consumer consent mechanisms in an electronic environment. *Journal of the American Medical Informatics Association : JAMIA*, 11(2):129–140, 2004.
5. European Communities. Directive 95/46/ec of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, nov 1995. `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2001:008:0001:0022:EN:PDF`.
6. Jing Jin, Gail-Joon Ahn, Hongxin Hu, Michael J. Covington, and Xinwen Zhang. Patient-centric authorization framework for sharing electronic health records. In *Proceedings of the 14th ACM symposium on Access control models and technologies*, SACMAT '09, pages 125–134, New York, NY, USA, 2009. ACM.
7. Michiharu Kudo. Pbac: Provision-based access control model. *International Journal of Information Security*, 1:116–130, 2002. 10.1007/s102070100010.
8. OASIS. extensible access control markup language (xacml) version 2.0, February 2005. `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf`.
9. Christine M. O'Keefe, Paul Greenfield, and Andrew Goodchild. A decentralised approach to electronic consent and health information access control. *Journal of Research and Practice in Information Technology*, 37(2), 2005.
10. Chun Ruan and Vijay Varadharajan. An authorization model for e-consent requirement in a health care application. In Jianying Zhou, Moti Yung, and Yongfei Han, editors, *Applied Cryptography and Network Security*, volume 2846 of *Lecture Notes in Computer Science*, pages 191–205. Springer Berlin / Heidelberg, 2003.
11. G. Russello, Changyu Dong, and N. Dulay. Consent-based workflows for healthcare management. In *Policies for Distributed Systems and Networks, 2008. POLICY 2008. IEEE Workshop on*, pages 153 –161, 2008.
12. R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-based access control models. *Computer*, 29(2):38 –47, February 1996.
13. Griet Verhenneman. Consent, an instrument for patient empowerment? In *Proceedings of the 49th FITCE Congress*, 2010.
14. Kim Wuyts, Riccardo Scandariato, Griet Verhenneman, and Wouter Joosen. Integrating patient consent in e-health access control. *IJSSE*, 2(2):1–24, 2011.
15. R. Yavatkar, D. Pendarakis, and R. Guerin. A Framework for Policy-based Admission Control. RFC 2753 (Informational), January 2000. `http://www.ietf.org/rfc/rfc2753.txt`.

16. Xinwen Zhang, Masayuki Nakae, Michael J. Covington, and Ravi Sandhu. A usage-based authorization framework for collaborative computing systems. In *Proceedings of the eleventh ACM symposium on Access control models and technologies*, SACMAT '06, pages 180–189, New York, NY, USA, 2006. ACM.