DataPAL: Data Protection and Authorization Lifecycle framework

Subhajit Bandopadhyay[***], Theo Dimitrakos[*†], Yair Diaz[*], Ali Hariri[*‖],
Tezcan Dilshener[§‡], Antonio La Marra[¶], Alessandro Rosetti[¶],
[*] *German Research Center*, *Huawei Technologies Duesseldorf GmbH*, Munich, Germany, {name.surname}@huawei.com
[†] *School of Computing*, *University of Kent*, Canterbury, UK
[‡] *European Business School University*, Munich, Germany, {name.surname}@euruni.edu
[§] *IT All Software Solutions*, Munich, Germany, {inital.surname}@it-all-sw-sol.com
[¶] *Security Forge*, Pisa, Italy, {namesurname}@security-forge.com
[‖] *Department of Computer Science and Information Engineering, University of Trento*, Trento, Italy, {name.surname}@unitn.it
[**] *Institute for Cyber Security, School of Mathematics, Computer Science and Engineering,*
City, University of London, London, United Kingdom, {name.surname}@city.ac.uk

***Abstract*—This paper introduces a new model for handling data privacy throughout data lifecycle via the introduction of a policy profile using the Abbreviated Language For Authorization (ALFA) policy language.**
Our approach extends previous models in three complementary ways: (1) By introducing Administration and Delegation Profile (ADP) in ALFA policy where users and companies can restrict the scope of access/usage policies related to data as well as specify a chain of custody for data (moreover such an approach eases up the tasks of handling users' consent); (2) Thanks to our framework Usage Control System Plus (UCS+) users can monitor the usage of data and revoke its usage upon specific conditions or at will; (3) By introducing new states for policy evaluation, i.e. Admissible/NotAdmissible to filter out those applicable policies that were unauthorized in the first place.

***Index Terms*—Data Privacy, GDPR, Usage Control, Authorization, Data Protection**

I. Introduction

Over the past years, the protection of user privacy and the proper handling of user data has become a major concern with increasing importance [1]. This is caused by the rise of ubiquitous internet (e.g., smart devices, social networks, etc.) that generate, collect and share huge amounts of personal data such as location information, video feeds and user behaviour [2]. Furthermore, these concerns were fueled by many privacy-related incidents in which personal data was abused and exploited, such as the most famous scandal the Cambridge Analytica [3].

As a response to the need for protecting citizens' privacy and ensure the appropriate use of their data, laws such as the General Data Protection Regulation (GDPR), have been introduced [1]. In these days, many initiatives like the GAIA-X project[1] have been launched in order to enforce policies to control data access and usage, as well as to guarantee that smart systems are privacy-preserving and compliant with regulations.

One of the privacy protection principles that must be supported in such systems is known as intervenability, which offers the ability for users to intervene (e.g. withdraw their data) in case inconsistencies occur with their data. In addition, it must be guaranteed that the users are informed about how their data will be used and their consent allowing or denying the processing of their data must be obtained. Also, it must be possible for users to withdraw their consent even after data processing has already started.

In our previous work [4], we presented a session-dependent authorization framework that provides an enhanced, expressive and flexible data protection support in big data environments. The framework organises authorizations in a hierarchy that represents dependencies between them. As such, if an authorization was revoked due to an attribute update, all subsequent and dependent authorizations would be revoked. This allows the system to end any data processing operations if the corresponding authorization conditions do not hold anymore.

In another work, we introduced Usage Control System Plus (UCS+) [5], a novel trust-aware continuous authorization framework for Internet of Things (IoT). The framework is a fusion of Attribute Based Access Control (ABAC) [6] policy engine and a Trust Level Evaluation Engine (TLEE). It uses a policy language known as the ALFA [7] and extends it with trust-level evaluation formulae, and supports contextualised and continuous monitoring of attributes as well as re-evaluation of policies.

In this paper, we extend UCS+ and ALFA to support Administration and Delegation Profile (ADP) in ALFA policy language by incorporating administrative policies which are particular policies to define administration and delegation. More importantly, we introduce a new policy model that uses administrative policies to support user consent and intervenability, namely the ADP. The model allows data owners to

This work is part of a Huawei R&D project in cooperation with Security Forge.

[1] https://www.gaia-x.eu/

978-1-6654-2742-5/21/$31.00 ©2021 IEEE

define data-centric policies to be enforced in each and every step of the lifecycle of their data (i.e. collection, retention, treatment and deletion), giving them full control over their data and privacy. An example of data lifecycle is represented in Figure 1 where the various stages of collection, access, usage, storage, transfer and deletion are highlighted.

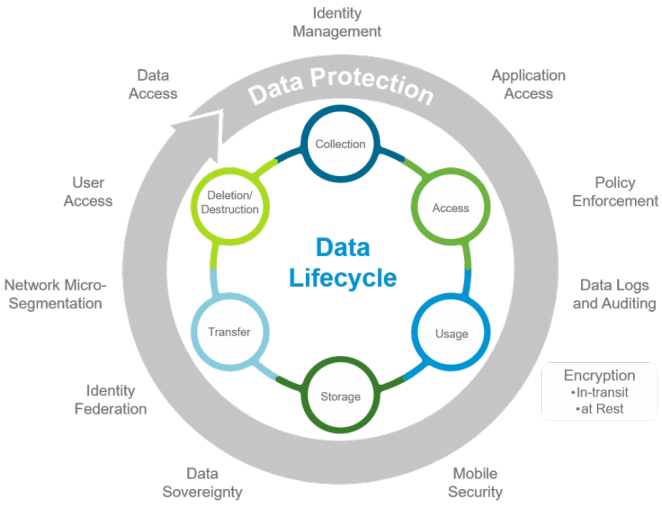

Fig. 1. Data Lifecycle [8]

It also supports separation of concerns and shared responsibility. In addition, the model helps in compliance by using root of trust administrative policies to encode regulations that define what data owners and data processors can do about the data. Thanks to UCS+, administrative policies can be re-evaluated when a change occurs and authorization decisions can be updated accordingly.

The rest of the paper is organized as follows: Section II presents the related work. In Section III We outline the architecture of UCS+ and in Section IV, we introduce our policy model. We describe the policy language used, the administration and delegation profile, the policy editing and evaluation workflows, the data lifecycle policies and then provide an example use-case. Section V refers to our prototype implementation and discusses the results of the performance analysis in realistic cloud settings. Finally, we conclude in Section VI by summarizing the main innovations and discussing future research directions.

II. Related Work

Meis and Heisel [9] specified a taxonomy of software requirements that can guide engineers to identify intervenability requirements for a specific software system and presented the taxonomy as an extensible metamodel using UML class diagram. The authors classified intervenability into two main requirements: data subject interventions and supervisory authority interventions. The interventions of supervisory authorities are: suspending data flows, ordering a ban of data processing, or the erasure or rectification of data. The interventions of data subjects are: withholding or withdrawing consent, reviewing or challenging the accuracy and completeness of data, disapproving specific data processing operations, and requesting a copy of data. The consequences of such interventions include the suspension of data flows, accessing or copying data by data subjects, amendment, correction or erasure of data, and restriction of data processing. As an important aspect of data protection in the entire data processing lifecycle, intervenability has been one of the core properties of our DataPAL framework.

Chadwick et al. [10] introduced a solution for supporting dynamic delegation of authority (DOA) between users in multiple administrative domains. Their solution is focused only on dynamic delegation of user-role assignment in which users delegate their roles to other users. They specifically introduced a new component designated as the Credential Validation Service (CVS) to work alongside the Policy Decision Point (PDP) and support DOA. The purpose of the CVS is to evaluate a credential validation policy to validate a set of subject credentials issued by multiple attribute authorities including delegates. As such, the CVS returns a set of validated subject attributes to be used by the PDP for the evaluation of an authorization policy. The CVS uses a top-level trusted credential validation policy that defines delegation conditions and which credential issuers and sub-policies can be trusted. With the help of administrative policies as stated in ADP, we are able to control the authorization lifecycle and support dynamic delegation of authority in our framework.

The work of Daoudagh and Marchetti [11] defines a GDPR-based lifecycle for authorization systems that assures the by-design compliance of the developed access control systems with GDPR. It also introduces an integrated environment that automatically enforces the lifecycle. The authors refer to the eXtensible Access Control Markup Language (XACML) standard in their work, however, the lifecycle is generic and can be used in any ABAC implementation. The defined lifecycle consists of eight steps starting with defining the context and use-case from a GDPR point of view. The subsequent steps include gathering authorization requirements according to GDPR regulations, identifying required attributes, authoring authorization policies that encode GDPR's provisions, testing and verifying policies and access control mechanisms to ensure that they meet the GDPR regulations, deploying policies and mechanisms, and running access reviews to assess the implemented solutions. We ensure regulatory compliance in view of data protection since administrative policies would be able to enforce policy evaluation as and how contexts keep changing in a dynamic data processing scenario.

To support user-centric access control, Preuveneers and Joosen [12] present a dynamic granular access control solution that works on top of existing authorization frameworks such as OpenID Connect and User Managed Access (UMA) 2.0. The solution supports the delegation of authorization decisions to multiple stakeholders in continuously evolving federations of cloud and edge microservices. This was achieved by adding a delegation microservice that intervenes with UMA 2.0 protocol flow to identify the relevant permissions and scopes

and redirect the client to the correct authorization server. The delegation microservice makes delegation decisions based on resource owners and requesting parties using delegation policies written in the Open Policy Agent (OPA) language. The solution leverages microservice technologies to support flexibility and scalability and to adhere to the security needs of different stakeholders in microservice federations.

Plappert et al. [13] introduced a vehicular privacy monitoring and privacy-aware data access system that informs and allows users to control sensitive data flows. The system allows users to define their own privacy settings then monitors and controls data flows accordingly and in compliance with the GDPR. It uses privacy enhancing technologies to modify data according to user policies. The system includes a human machine interface module that allows the user to review privacy-sensitive data flows and to adjust privacy settings to control the data flows. The privacy settings are then transformed into XACML policies that are used by other modules to control data flows. The monitoring module intercepts data flows between in-vehicle resources and third-party applications or external services, and evaluates XACML policies to decide how to handle the intercepted data flows.

In our work, we support user-centric data usage policies for controlling flow of information among different parties and we continuously monitor policy attribute values for state change. Depending upon the policy, a re-evaluation could be triggered which can change or deny access to data for resource owners.

III. Architecture

Our system UCS+ supports continuous context monitoring, usage revocation and policy re-evaluation, and enables secure data sharing and control. This makes it a perfect fit to address the objectives highlighted in the Introduction section. Namely, intervenability and consent, where users are able to define, via policy, how their data can be collected, used or transferred and eventually how each action should be performed. UCS+ is implemented according to a message-centric integration architecture inspired by microservices. It leverages in-line context enrichment and the Publish/Subscribe (Pub/Sub) pattern in order to maximise concurrency between policy parsing, policy evaluation and attribute retrieval. As such, UCS+ enhances performance and reduces the need for low network latencies or high computational resources. It also improves the ability to upgrade or substitute components and services and to migrate to a distributed deployment where necessary.

UCS+ incorporates the core components of a standard Usage Control (UCON) framework, as well as new components that support the additional functionalities and optimisations of UCS+. The architectural components are shown in Figure 2 and briefly described as follows:

- `Context Handler (CH)` is the core component of the framework, acting as message enricher and manager of the operative workflow.
- `Message Bus (MSG BUS)` is a sub-component of the CH that implements and supports the Pub/Sub communication paradigm.
- `Session Manager (SM)` is a vital component enabling the continuity of control as well as the storage of information about all active sessions.
- `Obligation Manager (OM)` handles and manages policy and rule obligations.
- `Attribute Table (AT)` is an auxiliary component that manages and caches attributes. It is needed in faulty environments where attribute values are not available right away when queried.
- `Attribute Retriever (AR)` is an auxiliary component in charge of querying and updating attribute values.
- `Policy Information Point (PIP)` defines where to find attributes and how to monitor them and collect their values.
- `PIP Registry` manages PIPs and defines which PIPs are responsible for which attributes.
- `Policy Enforcement Point (PEP)` receives evaluation requests (PEPA), interacts with the CH to evaluate them, and returns an evaluation result (PEPO).
- `PDP` is the component that evaluates policies matching a specific request, and returns one of the following possible decisions:
  - `PERMIT`;
  - `DENY`;
  - `NOT APPLICABLE`: Decision cannot be taken due to semantic reasons, i.e. no rules in the policy are matched; or
  - `INDETERMINATE`: Decision cannot be taken since either the policy or the request are malformed.

  The PDP leverages ABAC Expression Evaluator (AbacEE) to perform ABAC policy evaluation, and supports auxiliary evaluators. As such a dedicated evaluator that supports ADP has been implemented and plugged inside PDP with minimal effort.
- `Policy Administration Point (PAP)` is a management interface used to create, read, update and delete policies. It stores and manages policies and is used by the PDP to retrieve applicable policies.

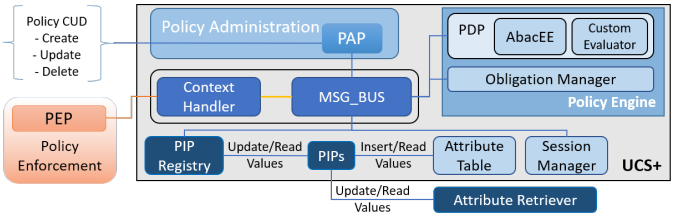

Fig. 2. Architecture with interconnections

IV. Policy Model

A. Administration and Delegation Profile (ADP)

ADP [14] is a profile introduced in XACML 3.0 to address use-cases in which administration of policy issuance or delegation of policy authority is required. For instance,

system administrators or resource owners may want to restrict and control who can write policies about their resources. As another example, a resource owner may want to delegate their authority to someone else while they are away or on vacation. To support such use-cases, the ADP allows administrators to write policies about other policies forming trees that start with top level policies designated as root of trust. In the scope of this paper, we refer to controller policies (i.e. policies that control other policies) as *Administrative Policies*. On the other hand, we designate controlled policies (i.e. access policies) as *Usage Policies*. As such, administrative policies define who is authorized to write other administrative policies or specific usage on data, resources or services.

According to the ADP, a Usage Policy will not be enforced unless it is explicitly authorized by Administrative Policies. Thus, a Usage Policy can be *Admissible* if its enforcement is authorized by all upper layer policies, or *NotAdmissible* if one of the Administrative Policies in the hierarchy cannot be considered. This is expressed in the evaluation outcomes truth table illustrated in Table I, which shows that whenever Administrative Policies are admissible, a final result is obtained. On the other hand, if one of the Administrative Policies is NotAdmissible, then another policy needs to be found. Indeed, in some cases, multiple policies may be Admissible and have a Permit/Deny result. For this reason, a so-called policy reduction algorithm is used. It is of vital importance to manage inconsistencies and to guide editing of administrative policies because otherwise all evaluations will end up in a NotAdmissible state. To achieve this, once an administrative policy is added, compliance with the above layers can be checked and suggestions for improvements may be provided. Therefore editing a full-fledged policy with ADP resembles the one shown in Listing 1.

TABLE I
EVALUATION OUTCOMES

Usage Policy	Administrative Policy	Result
Permit	Admissible	Permit
Deny	Admissible	Deny
Permit	NotAdmissible	Find other policy
Deny	NotAdmissible	Find other policy

```
<given a root policy defined by authority / organization>
user writes custom policy Pa
synthetic request is crafted from Pa
request is evaluated against Pa
    select administrative policy Padmin for Pa
        for each Pi in Padmin
            create administrative request Ra using R and Pi
            evaluate Ra against Pi
            if deny
                <show modifications>
            else
                <allow policy>
```

Listing 1. Evaluation and Reduction process

B. Extended Policy Language

UCS+ uses ALFA [7] as a baseline policy language. ALFA is a pseudocode domain-specific policy language that maps directly to XACML [15] without adding any new semantics. It is much less verbose than XACML, which makes it more human readable and shorter in size allowing faster parsing and evaluation. ALFA adheres to the same hierarchy as XACML where decision predicates are expressed in rules that are nested under policies, which are in turn nested under policy sets. ALFA policy sets may also enclose other policy sets, and rules must evaluate into either *permit* or *deny*. Like XACML, ALFA relies on combining algorithms to resolve conflicts between sibling rules or policies. ALFA also allows the use of functions, such as regular expression, string concatenation and others, which helps to further refine applicability of policies and rules. Accordingly, an ALFA policy must adhere to the non-inclusive structure shown in Listing 2.

In our previous work [5], we extended ALFA by classifying policy rules as "*pre*", "*ongoing*" and "*post*", adding an implicit temporal state to authorizations. In this work, we introduce another extension to ALFA to support ADP. We specifically introduced the "`PolicyIssuer`" keyword, which identifies the author of the policy, and is used to create an administrative request. In addition, we reserved the following attribute categories to be used for ADP:

- `Attributes.delegate` attribute category used in administrative requests to carry the attributes of the issuer of the policy which is being reduced.
- `Attributes.delegation-info` attribute category used in administrative requests to carry information about the reduction in progress, such as the decision being reduced.
- `Attributes.delegated` category used to carry information about the situation which is being reduced.

These extensions support the policy model and policy editing workflow described in the following subsection.

```
policy <Title> {
  target clause
          Attributes.<AttributeName> == <AttributeValue>
  apply <CombiningAlgorithm>
  rule <RuleName> {
    target clause
          Attributes.<AttributeName> == <AttributeValue>
    condition <ConditionExpression>
    <decision (permit or deny)>
    on <decision> {
      <instruction (obligation)> <ObligationName> {
        <Attribute> = <Value>
}}}}
```

Listing 2. Structure of an ALFA policy

C. Policy Editing and Evaluation Workflow

Using the extensions we added to ALFA, we introduce the following policy editing workflow, which is also demonstrated in the example workflow shown in Figure 3. The *Chief Information Security Officer (CISO)* is in charge of creating

top-level root-of-trust administrative policies that scope organisational, regulatory and business constraints. For instance, the CISO specifies the actions, resources, usage purposes and other attributes that Data Subject (DS) and other actors are allowed to control in their policies. *DS* (i.e. data owner) is delegated by the CISO, and is in charge of creating administrative policies that delegate authority to other actors (e.g. managers). DS policies define who can write usage policies about their data, and what kind of actions and data processing operations can these usage policies control. Finally, managers write usage policies about the data and define the constraints for data usage and processing. In summary, the CISO group delegates to the DS group, and the DS group in turn delegates to the managers group, which permits or denies staff from accessing data. An example of an administrative and usage policies is shown in Listing 3, where the CISO policy allows all data subjects to write policies about any action performed on their data. Similarly, the data subject policy allows any manager to write policies about CRUD operations performed on the data. Finally, the staff policy is a usage policy that allows staff member Bob to read the data.

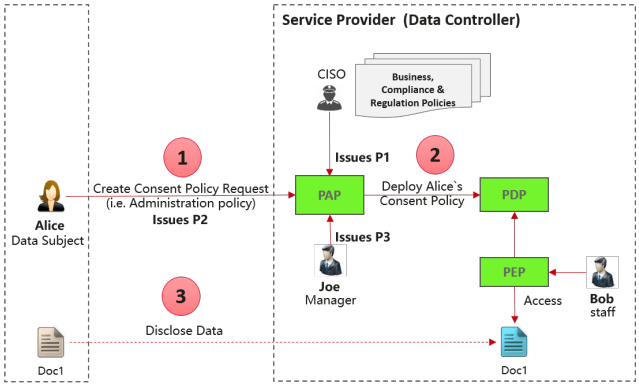

Fig. 3. Policy editing workflow

The policy evaluation flow can be summarized in two different steps: reduction and combination. Reduction determines whether a usage policy was written by an authorized personnel and whether the evaluation outcome can be considered or not. This is achieved by creating an evaluation tree having as root node a root-of-trust policy and as edges the results of applicable policies, then finding the branches that can reach the root-of-trust. In the combination step, the PDP combines all valid results using combining algorithms defined in the policies. This evaluation flow is described in Listing 4. The ADP defines a maximum depth of delegations in order to limit number of administrative policies to be evaluated. Accordingly, if the number of nodes of a path in the reduction graph exceeds the maximum delegation depth, the path will be discarded. The maximum delegation depth is not supported in this version of our framework. However, it will be added in future iterations.

For both administrative and usage policies, we use the *denyUnlessPermit* combining algorithm due to its deterministic and restrictive nature. This algorithm is restrictive because its default outcome is always *deny*, unless there is an explicit applicable permit rule. This also eliminates all indeterministic outcomes, such as *indeterminate*, in cases where an attribute value is missing or a condition is false. Therefore, if an administrative policy evaluates into *indeterminate*, the corresponding branch would be discarded from the overall delegation tree. Nonetheless, we intend to analyse the use of different combining algorithms and their effects and risks on the delegation tree and authorization decisions.

```
policyset l3_ad_policy_set {
  apply denyUnlessPermit
  policy ciso_retain {
    target clause
      regex(Attributes.delegated.action, "\/action\/(.*)") &&
      regex(Attributes.delegate.subject, "\/dataSubjects\/(.*)")
    apply denyUnlessPermit
    rule r1 {
      permit
    }
  }
  policy dataSubject_retain {
    policyIssuer { Attributes.subject = "/dataSubjects/Carol" }
    apply denyUnlessPermit
    target clause
      regex(Attributes.delegated.action, "\/action\/datalifecycle\/(.*)")
        &&
      regex(Attributes.delegate.subject, "\/employees\/manager\/(.*)")
    rule r2 {
      permit
    }
  }
  policy staff_retain {
    policyIssuer {Attributes.subject = "/employees/manager/Joe"}
    apply denyUnlessPermit
    target clause
      Attributes.subject == "/employees/staff/Bob" &&
      Attributes.action == "/action/datalifecycle/retain"
      ...
  }
}
```

Listing 3. Example of administrative policies

D. Data Protection and Retention Policies

The above model has enabled a new approach to data privacy. This is because CISOs can write administrative policies that state the characteristics of delegated actors, such as data owners, and the aspects on which these actors can write policies. Furthermore, data owners can state possible data manipulation or anonymization techniques to be used whenever their data are used, adding an extra level of protection to better guarantee user privacy. Since administrative policies can be stored in remote or separate storage or systems, e.g., individual wallets, data owners have the capability of modifying them as they wish. Data owners can eventually further delegate some aspects, if they want to, to data managers.

Data lifecycle policies are, afterall, usage control policies, and as such, their rules can be classified by the UCS+ phases (i.e. "*pre*", "*ongoing*" and "*post*"). Modeling data lifecycle policies according to UCS+ phases allows data owners to define usage policies related to different stages of a usage

```
<given a policy set PS and a request R>
evaluate R against PS
find applicable policies Papp
for each applicable policy Pa in Papp
    if deny
        continue
    if PolicyIssuer is absent
        then combine
    else
        [selection step] : select administrative policy Padmin for Pa
            for each Pi in Padmin
                create administrative request Ra using R and Pi
                evaluate Ra against Pi
                if deny
                    no edge
                else
                    add edge
                    if !policyIssuer
                        potential path is found
                    else
                        go to selection step with Pi as Pa

combine edge results with PS combining algorithm
```

Listing 4. Evaluation and Reduction process

session, giving them full control over the lifecycle of their data, from collection to treatment to destruction (Figure 1). For instance, UCS+ phases can be used to control the data lifecycle as follows:

- *pre*: The data owner needs to provide consent for new data before it is collected and accessed for the first time. First-time collection and access consent may be recurrent over a time period.
- *ongoing*: Once data is collected and first-time accessed; consent defines the conditions of data usage, storage, transfer, and etc.
- *post*: Finally, once data is no longer used or access/usage is violated, consent defines the conditions of archival, deletion, destruction, and etc. of the data as well as any associated clean-up tasks.

An example of such policy is shown in Listing 5. This makes the system GDPR-compliant and reduces the need to trust and rely on third parties for proper data management and security. EU regulations, such as electronic IDentification, Authentication and trust Services (eIDAS) [16], go exactly in this direction, that is guaranteeing data owners full control on their data and to, eventually, explicitly and dynamically authorize each and every attempt to use their data [1]. By adding an additional layer of administrative delegation policy, one can also take into account regulations and norms stated by EU or other authorities [1]. This is because it is possible to have a first layer where regulation is defined, another where each user states their own administrative policy and finally usage policies are defined.

E. Use Case

In a typical healthcare scenario, a patient shares with hospital personnel many of their personal and sensitive information: address, income, insurance, diseases, whether they have donated an organ or blood, etc. The patient needs to

```
policy staff_retain {
    policyIssuer {Attributes.subject = "/employees/manager/Joe"}
    target clause
      Attributes.subject == "/employees/staff/Bob" &&
      Attributes.action == "/action/datalifecycle/retain" &&
      Attributes.dataRecord.id == "entryData" &&
      Attributes.dataRecord.owner == "dataSubject" &&
    Attributes.purpose == "userExperienceOptimization"
  apply denyUnlessPermit
  rule pre {
    target clause Attributes.ucs.step.pre
    condition Attributes.dataRecord.userConsent
    permit
    on permit {
      obligation create {
        dataRecord = Attributes.dataRecord.id
      }
    }
  }
  rule ongoing {
    target clause Attributes.ucs.step.ongoing
    condition
      Attributes.dataRecord.userConsent and
      Attributes.dataRecord.encrypted and
      Attributes.dataRecord.expiryDate > Attributes.currentTime
    permit
  }
  rule post {
    target clause Attributes.ucs.step.post
    deny
    on deny {
      obligation delete {
        msg = "Record deleted"
        dataRecord = Attributes.dataRecord.id
      }
      obligation notify {
        dataRecord = Attributes.dataRecord.id
        hasUserConsented = Attributes.dataRecord.userConsent
        isEncrypted = Attributes.dataRecord.encrypted
        isDataExpired =
          (Attributes.dataRecord.expiryDate < Attributes.currentTime)
      }
    }
  }
}
```

Listing 5. Example of data lifecycle policy

properly protect such information from disclosure or from misusage to protect their own privacy. To achieve this, the patient can define an administrative policy that states how their data should be managed by the organisation and how the organisation's policies will be handled. Such a profile does not require to define a full-fledged policy, rather the patient can delegate someone else as a doctor or a relative. The beauty of using the aforementioned profile is the high degree of flexibility it provides to people in different conditions. To mitigate abuse, patient is delegated by the hospital itself which is the so-called root of trust. In the hospital, for example, a policy states that patients' data can be shared in a medical équipe. On the other hand, our policy model allows the patient to state that no sharing policies are allowed on their data. Therefore even if the hospital's policy can be permitted, the flow ends up in a NotAdmissible state, thus data sharing will be denied.

V. Performance

To evaluate the viability of the proposed policy management and data protection model, we carried out two different sets of performance evaluation experiments on a machine equipped with 16 GB of RAM and *Intel Core i7 ®9750H @ 2.60GHz* CPU, running Ubuntu Linux operating system. In the first set of experiments we have evaluated the performance of the PDP in standalone mode. In this experiment set, first test was conducted as a control reference, in which only a usage policy was evaluated without any administrative policies. In the other tests, the number of administrative policies included in the evaluation was increased from one up to three policies. The usage policy that was used in the experiment uses four string attributes, while each of the administrative policies uses five string attributes. The experiments were conducted without any optimisation of the reduction algorithm described in Listing 4.

The results are shown in Figure 4 which illustrates the time required to fully evaluate a usage policy with different administration levels. It can be noticed that the required time is negligible and increases linearly, as expected, when the number of administrative policies increases. To evaluate an administrative policy, a new request has to be crafted, which introduces an overhead to the usage policy evaluation. However, this is one of the aspects that can be further optimised by exploiting caching, parallel evaluation or other policy features to make the system faster.

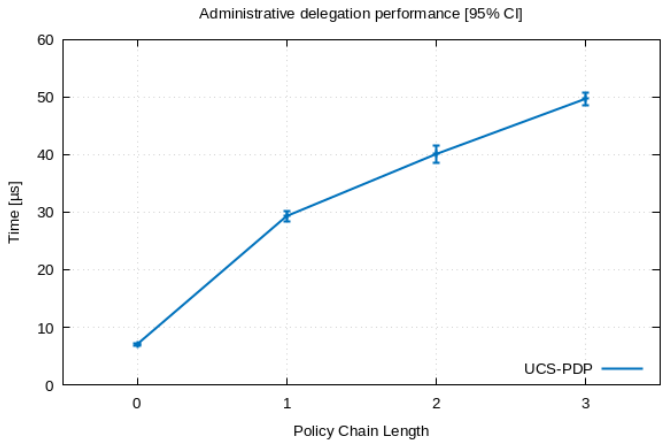

Fig. 4. Administrative delegation performance impact on PDP evaluation

In addition to the conducted experiments, we also measured the overall full flow evaluation of a usage policy with an increasing number of administrative policies. The policies used in this measurement are the same policies used in the aforementioned experiments.e Figure 5 illustrates the results and shows that the performance does not vary significantly when the number of administrative policies increases. The results also show that the overall time required to evaluate a usage policy and to reach the root-of-trust administrative policy is no more than 0.5ms. This shows that the overhead introduced by our model can be tolerated although further optimisations are also possible.

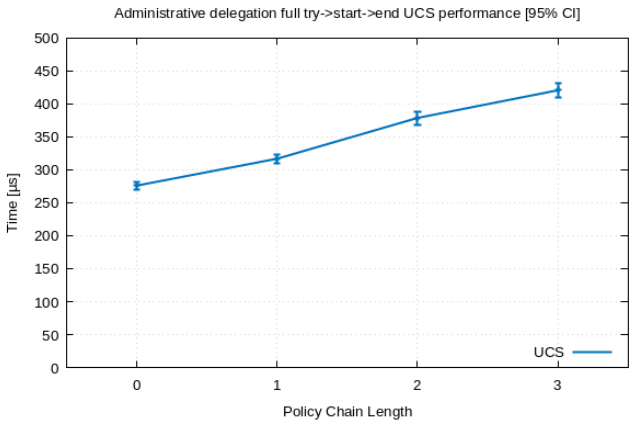

Fig. 5. Administrative delegation performance impact on UCS full flow

VI. Conclusion

In this work, we presented a new approach for the handling of data protection and privacy requirements. The approach consists of a policy model that leverages on the policy structures of both the Administration and Delegation Profile (ADP) and our framework Usage Control System Plus (UCS+) models. The ADP model was adapted and re-purposed in order to support data protection use-cases. The adaptation enables the specification and systematic hierarchical scoping of data protection requirements in the following order: the Data Processor at management level (e.g., compliance/regulatory) by means of administrative policies, the Data Subject (e.g., consent, usage purpose) by means of further constrained administrative policies, and by the Data Processor at operational level (e.g., business workflows) by means of further constrained access policies. The ALFA policy language syntax was also extended to support the ADP as defined by the XACML specification. In addition, the temporal state classification of ALFA policy rules, introduced in UCS+, was used to control the different phases of the data lifecycle. This was achieved by mapping the *pre*, *ongoing* and *post* phases respectively to data collection, data usage (e.g. access, processing, etc.) and data destruction/deletion.

In our future work, we aim at improving the performance of administrative policy evaluation by caching administrative requests or by evaluating them in parallel to usage requests. This should flatten the performance results curve even more while increasing the number of administrative policies. Such optimisations will take into account that the same administrative policy may be applied on multiple data sets. We also plan to support the maximum delegation depth as defined by the ADP, and to introduce the implicit temporal state (i.e. *pre*, *ongoing* and *post*) in administrative policies, allowing administrative authorization sessions. Moreover, we intend to analyse the interplay between the combining algorithms of both administrative and usage policies, as well as their effect on the final evaluation decision. Finally, we intend to explore and verify the viability of using our policy model in other

scenarios such as IoT applications.

REFERENCES

[1] E. Parliament, "REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016," 2016. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=OJ:L:2016:119:FULL&from=EN

[2] IotaComm, "how does IoT affect Big Dta?" 2020. [Online]. Available: https://www.iotacommunications.com/blog/iot-big-data/

[3] H. Davies, "Ted Cruz using firm that harvested data on millions of unwitting Facebook users," *the Guardian*, vol. 11, p. 2015, 2015. [Online]. Available: https://www.theguardian.com/us-news/2015/dec/11/senator-ted-cruz-president-campaign-facebook-user-data

[4] G. Baldi, Y. Diaz-Tellez, T. Dimitrakos, F. Martinelli, C. Michailidou, P. Mori, O. Osliak, and A. Saracino, "Session-dependent Usage Control for Big Data," *J. Internet Serv. Inf. Secur.*, vol. 10, no. 3, pp. 76–92, 2020. [Online]. Available: https://doi.org/10.22667/JISIS.2020.08.31.076

[5] T. Dimitrakos, T. Dilshener, A. Kravtsov, A. L. Marra, F. Martinelli, A. Rizos, A. Rosetti, and A. Saracino, "Trust Aware Continuous Authorization for Zero Trust in Consumer Internet of Things," in *19th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2020, Guangzhou, China, December 29, 2020 - January 1, 2021*, G. Wang, R. K. L. Ko, M. Z. A. Bhuiyan, and Y. Pan, Eds. IEEE, 2020, pp. 1801–1812. [Online]. Available: https://doi.org/10.1109/TrustCom50675.2020.00247

[6] V. C. Hu, D. Ferraiolo, R. Kuhn, A. R. Friedman, A. J. Lang, M. M. Cogdell, A. Schnitzer, K. Sandlin, R. Miller, K. Scarfone *et al.*, "Guide to attribute based access control (abac) definition and considerations (draft)," *NIST special publication*, vol. 800, no. 162, 2013.

[7] OASIS, "Abbreviated language for authorization Version 1.0," 2015. [Online]. Available: https://www.oasis-open.org/committees/download.php/55228/alfa-for-xacml-v1.0-wd01.doc

[8] D. Williams, "Data Protection Lifecycle," [Pinterest post]. [Online]. Available: https://www.pinterest.ca/pin/444871269435969291/

[9] R. Meis and M. Heisel, "Understanding the Privacy Goal Intervenability," in *Trust, Privacy and Security in Digital Business - 13th International Conference, TrustBus 2016, Porto, Portugal, September 7-8, 2016, Proceedings*, ser. Lecture Notes in Computer Science, S. K. Katsikas, C. Lambrinoudakis, and S. Furnell, Eds., vol. 9830. Springer, 2016, pp. 79–94. [Online]. Available: https://doi.org/10.1007/978-3-319-44341-6_6

[10] D. W. Chadwick, S. Otenko, and T. Nguyen, "Adding support to XACML for multi-domain user to user dynamic delegation of authority," *Int. J. Inf. Sec.*, vol. 8, no. 2, pp. 137–152, 2009. [Online]. Available: https://doi.org/10.1007/s10207-008-0073-y

[11] S. Daoudagh and E. Marchetti, "A Life Cycle for Authorization Systems Development in the GDPR Perspective," in *Proceedings of the Fourth Italian Conference on Cyber Security, Ancona, Italy, February 4th to 7th, 2020*, ser. CEUR Workshop Proceedings, M. Loreti and L. Spalazzi, Eds., vol. 2597. CEUR-WS.org, 2020, pp. 128–140. [Online]. Available: http://ceur-ws.org/Vol-2597/paper-12.pdf

[12] D. Preuveneers and W. Joosen, "Towards Multi-party Policy-based Access Control in Federations of Cloud and Edge Microservices," in *2019 IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops 2019, Stockholm, Sweden, June 17-19, 2019*. IEEE, 2019, pp. 29–38. [Online]. Available: https://doi.org/10.1109/EuroSPW.2019.00010

[13] C. Plappert, D. Zelle, C. Krauß, B. Lange, S. Mauthöfer, J. Walter, B. Abendroth, R. Robrahn, T. von Pape, and H. Decke, "A privacy-aware data access system for automotive applications," in *15th ESCAR Embedded Security in Cars Conference*, 2017.

[14] E. Rissanen, H. Lockhart, and T. Moses, "XACML v3.0 Administration and Delegation Profile Version 1.0," *Committee Draft*, vol. 4, 2014. [Online]. Available: https://docs.oasis-open.org/xacml/3.0/xacml-3.0-administration-v1-spec-en.html

[15] OASIS, "eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01," 2017. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.html

[16] Council of European Union, "Council regulation (EU) No. 910/2014 electronic IDentification, Authentication and trust Services for electronic transactions in the internal market and repealing Directive 1999/93/EC," 2014. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2014.257.01.0073.01.ENG