Metamodel for Continuous Authorisation and Usage Control

Ali Hariri[1,2][0000−0002−1985−6496], Amjad Ibrahim[2], Theo Dimitrakos[2,3], and Bruno Crispo[1]

[1] Department of Information Engineering and Computer Science, University of Trento, Trento, Italy
{name.surname}@unitn.it
[2] German Research Center, Huawei Technologies Düsseldorf GmbH, Munich, Germany
{name.surname}@huawei.com
[3] School of Computing, University of Kent, Canterbury, United Kingdom
{n.surname}@kent.ac.uk

Abstract. Access control has been traditionally used to protect data and privacy. Traditional access control models (e.g., ABAC, RBAC) cannot meet modern security requirements as technologies spread over heterogeneous and dynamic environments that need continuous monitoring. Modern models such as Usage Control (UCON) introduced the concept of continuous authorisation that has a lifecycle consisting of a series of phases through which the authorisation passes during its lifetime. However, such models assume a fixed lifecycle for all authorisations, so they cannot satisfy emerging technologies (e.g., smart vehicles, zero-trust, data flow), which require various and fine-grained lifecycles. Researchers have extended existing models to meet such requirements, but all solutions remain restrictive, as they are specially tailored for specific use-cases. In this paper, we propose an extensible model for continuous authorisations and usage control. The model enables its users to customise and dynamically configure the authorisation lifecycle as required by the use-case. This adds a layer of abstraction, forming a metamodel that can be instantiated into different flavours of continuous authorisation models, each addressing specific requirements. We also show that the authorisation lifecycle can be modelled as Deterministic Finite Automaton (DFA) and expressed in a structured language used by an evaluation engine to dynamically enact and manage the lifecycle. We layout the building blocks of the proposed metamodel and devise future research directions.

Keywords: UCON · Usage Control · Continuous Authorisation · Authorisation Lifecycle · Extensible Access Control

1 Introduction

With the rise of the ubiquitous Internet (e.g., smartphones, Internet of Things (IoT) and social networks), data security and user privacy have become major

concerns fueled by privacy-related incidents such as the Cambridge Analytica scandal [8, 20]. Access control has been traditionally used to protect data and privacy. It evolved from coarse-grained models such as Mandatory Access Control (MAC) and Role Based Access Control (RBAC) to more flexible and fine-grained models such as Attribute Based Access Control (ABAC). However, these models assume that access rights do not change during access, so they do not react to situation changes. For this reason, Park and Sandhu [18] proposed the Usage Control (UCON) model as a generalisation that extends preceding models with attribute mutability and continuous control. The novelties of UCON enable it to revoke access when a change occurs and the security policy is no longer satisfied, which is required in long-lasting and dynamic authorisations such as cloud and IoT [4, 15, 16].

UCON introduced the concept of continuous authorisation that spans over a period of time *within a session*, during which the security policy is re-evaluated and the authorisation decision is updated accordingly. A continuous authorisation passes through different phases during its lifetime, and different rules apply at each phase. We designate the set of such phases and the transitions between them as the *authorisation lifecycle*. UCON employs a fixed authorisation lifecycle that consists of three phases only, so all UCON authorisations always have the same lifecycle.

Governance initiatives and regulations, together with the accelerating growth of emerging technologies and use-cases have introduced more complicated authorisation requirements that cannot be met with UCON for the following reasons:

1. UCON enacts continuity *only while access is in progress*, while new technologies require continuous monitoring *before* granting authorisation and/or *after* revoking it. For instance, Zero-Trust Architectures (ZTAs) rely on continuous and behavioural authentication as well as security scoring [21], which requires monitoring before granting authorisation. Similarly, safety-critical applications such as smart vehicles and smart grids require monitoring after revoking authorisation to ensure that safety measures are taken [11].
2. UCON *revokes* and *terminates* authorisations when the security policy is no longer satisfied. However, some use-cases need to *temporarily suspend* the authorisation rather than completely revoking it. For example, changes in user behaviour or security score may not necessarily entail a termination of authorisation, but rather *withholding* it to allow the user to improve their security score [9].
3. UCON's lifecycle is static, coarse-grained and limited to three phases only, but new technologies and privacy laws necessitate finer-grained authorisation lifecycles based on different contexts or security threats. For example, big data lifecycle incorporates five different phases (i.e., data collection, retention, transfer, access and destruction) [14, 22].

In this paper, we propose an extensible model for continuous authorisation and usage control, and lay out its building blocks as well as future research directions. The model enables finer-grained, flexible and dynamic authorisation lifecycles by allowing its users to define any number and sequence of phases

as required. As such, the model can be instantiated into different flavours of concrete models, each specialised for a specific use-case, thus sustaining emerging applications as well as future use-cases. The foundation of our model is modelling the authorisation lifecycle as a Deterministic Finite Automaton (DFA) such that states represent phases. The DFA is then expressed in a structured language that can capture contextual information and define DFA transitions accordingly.

The rest of this paper is organised as follows: Section 2 describes background and related work; In Section 3, we define the authorisation lifecycle and its phases, and show how it can be modelled as a DFA; We introduce our model and explain how it enables user-defined lifecycles in Section 4; We outline preliminary work and future directions in Section 5, and conclude in Section 6.

2 Background and Related Work

Park and Sandhu [18] introduced UCON to extend access control with mutability of attributes and continuity of control. Unlike preceding models, UCON assumes that attribute values may change during usage of resources, so it specifies that policies must be continuously evaluated while access is in progress. UCON classifies decision predicates as *pre* and *ongoing* such that: *pre* predicates are evaluated when an access request is made to decide whether to grant or deny access, while *ongoing* predicates are re-evaluated whenever an attribute value changes while access is in progress. UCON also incorporates obligations, which are mandatory actions that must be performed by the subject to gain authorisation, and they are also classified as *pre* and *ongoing* obligations. Moreover, UCON adds optional procedures defined in security policies to update attribute values. Attribute updates are classified as *pre*, *ongoing* and *post* updates such that *pre* updates are enacted before granting access, *ongoing* updates are enacted while access is in progress, and *post* updates are performed after the end of access.

UCON revokes access when the security policy is no longer satisfied, which may disrupt user interactions in highly dynamic environments such as pervasive computing. For this reason, Almutairi and Siewe [1] extended UCON with the ability to adapt to changing contexts rather than revoking authorisation altogether. They added a phase designated as "adapting" to reflect the stage at which the authorisation session adapts to a changing context. They also specify their model as a DFA that describes how an access request is handled. The DFA, however, is coarse-grained and cannot be changed to support use-cases that require a different DFA.

Similarly, Dimitrakos et al. [9] extended UCON with trust awareness by involving trust level evaluation in authorisation decisions. Their model allows subjects to improve their trust level instead of denying authorisation if the trust level is low. This was achieved by adding a counter of re-evaluations such that the authorisation remains on hold until the subject improves their trust or until the re-evaluation counter expires. While this does add continuity to all phases, it remains restrictive to their use-case and the three traditional phases

of UCON. Moreover, their model cannot capture the context of re-evaluations, so the counter may expire due to re-evaluations unrelated to trust level, and consequently not allowing the subject to improve their trust.

In [6], Conti et al. introduced a context-aware access control model for Android devices. Their model monitors contextual information and reacts to changes by revoking a running application or service. While their model is not based on UCON, it provides similar novelties. The model has similar limitations to UCON as it does not cover monitoring before granting and after denying access, and it does not include a fine-grained authorisation lifecycle.

Bandopadhyay et al. [3] leveraged UCON to handle data privacy throughout the data lifecycle. Their framework allows users to intervene and revoke the consent to use their data. Although data lifecycle consists of five stages, they used UCON's three phases only. This limits the expressibility of the model and the ability to capture specific contexts related to the data lifecycle stages.

Likewise, other works [2, 4, 5, 13, 15, 16] have also built on UCON to enable continuous authorisation. Nonetheless, they remain restrictive to their corresponding use-cases, and limited to UCON's coarse-grained three phases only (i.e., *pre*, *ongoing* and *post*).

3 The Authorisation Lifecycle

We define the authorisation lifecycle as the series of phases through which an authorisation session passes during its lifetime from initial request to enforcement to revocation or termination. The lifecycle of a continuous authorisation includes a temporal aspect as one or more phases span over a period of time. In contrast, authorisations in traditional access control do not span over time, so their lifecycle phases are momentary and atomic (i.e., policy evaluation is performed only once). The authorisation lifecycle depends on the use-case and the suitable access control model. For instance, UCON includes three phases only, such that the lifecycle starts with the *pre* phase then moves to the *ongoing* phase and ends with the *post* phase. The authorisation lifecycle in emerging use-cases incorporates phases other than UCON's three phases. For example, data flow lifecycle includes five phases: *data collection*, *data retention*, *data access*, *data transfer* and *data destruction*. Other applications may require a lifecycle with phases for guiding users through the necessary steps to gain authorisation (e.g., Multi-Factor Authentication (MFA), solving CAPTCHAs, improving security score). In this section, we describe the properties of the authorisation lifecycle and show how it can be modelled as a DFA.

3.1 Lifecycle Phases

We define a phase as a particular step that characterises the authorisation at a particular point in its lifetime. This, however, does not mean that a phase describes the context of the authorisation neither does it describe the subject or the object. Thus, a phase must not be a decision factor in the security policy

or the evaluation function (e.g, must not be an attribute in ABAC or a role in RBAC). It must only be a characteristic within the scope of the authorisation itself, describing the state or the progress of the authorisation. For example, UCON's *pre* phase indicates that an access request has been made but access has not been granted yet, while the *ongoing* phase denotes that access has been granted and is in progress. Another example is the *retention* phase in data flow, which indicates that data will be stored for a period of time as part of the data usage authorisation, thus data retention policies apply. The series of phases and transitions between them constitute the authorisation lifecycle. They portray the nature of the authorisation and define a scheme or a profile for enacting it. The lifecycle determines how the authorisation should progress throughout its lifetime as different events occur and the evaluation decision changes.

3.2 Classification of Decision Predicates

Since phases reflect the different stages in the authorisation, different rules apply at each phase. Thus, decision predicates must be classified according to the phases of the authorisation lifecycle such that each class is evaluated only when the lifecycle enters the corresponding phase. Such classification is a mapping from the set of decision predicates to the set of phases. Policy evaluation may participate in driving the lifecycle such that the evaluation outcome in each phase determines the next phase in the lifecycle. For instance, the lifecycle may specify that if the policy evaluation outcome of *phaseA* is *permit*, then the authorisation must move to *phaseB*. Transitions between phases may also be influenced by events other than the evaluation decision such as contextual events.

3.3 Modelling the Lifecycle as a DFA

The authorisation lifecycle consists of a set of phases and transitions between them. For this reason, it can be modelled as a DFA such that DFA states represent phases, and the transition function together with the inputs represent the events that drive the lifecycle. We use the same formal DFA definition [7] to define the lifecycle as the following 5-tuple $(Q, \Sigma, \delta, q_0, F)$ such that:

- Q is a finite set of phases;
- Σ is a finite set of events and inputs (e.g., evaluation decisions, contextual events) used by the transition function to drive the lifecycle;
- δ is a transition function $\delta : Q \times \Sigma \rightarrow Q$;
- q_0 is an initial phase;
- F is a finite set of terminal phases.

The authorisation lifecycle must have the same properties as a DFA, so the following properties must be met when specifying a lifecycle in order to ensure its correctness:

- The lifecycle must have one and only one initial phase.
- The lifecycle must have at least one exit or final phase.

– All phases must be reachable.
– There must be one and only one transition for every unique input at each phase.
– The lifecycle must be deterministic, so the same events at a specific phase must always result in the same transition.

Examples of lifecycle DFAs are shown in Section 5.

4 The Model

In this section, we describe the proposed model and how it enables finer-grained and configurable authorisation lifecycless.

4.1 Model Components

The model consists of six elements and two functions as shown in Figure 1. The evaluation function evaluates decision predicates, which rely on attributes and the current lifecycle phase, then it outputs an authorisation decision and directives. The transition function takes the evaluation outcome, attributes, and the current phase as inputs, and evaluates lifecycle controllers to determine the next phase. These components compose authorisations and drive their lifecycles. They are further described in this section.

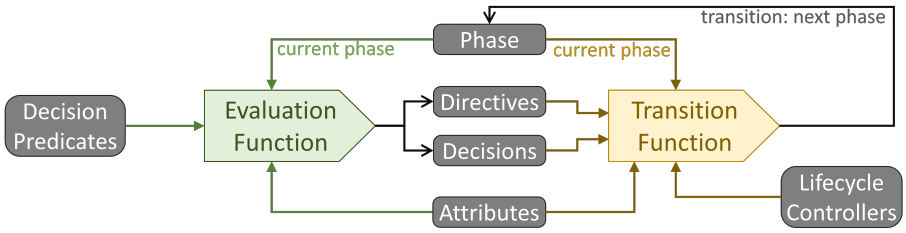

Fig. 1. Components of the proposed model

Attributes Attributes are characteristics about the subject, object or environment, such as a role, file type or temperature. We assume that all attributes are mutable and may change during the authorisation. A change may occur as a consequence of the authorisation process itself, or due to administrative actions or environmental changes. For instance, an increase in temperature is an environmental change, while updating the `last_modified` date of a file may be a consequence of an authorisation that permitted writing to the file. When an attribute value changes, a re-evaluation of decision predicates must be triggered.

Lifecycle Phases We consider all authorisations to have a lifecycle as described in Section 3. Thus, lifecycle phases are a core component of the model, defined and described in Section 3.1.

Decision Predicates Decision predicates constitute security policies. They are logical predicates over attributes, and are evaluated against a request to determine whether an authorisation can be granted, given the values of the attributes of interest. Decision predicates must be classified according to lifecycle phases, as described in Section 3.2, such that they are evaluated only when the authorisation lifecycle is in the phase under which they are classified. Taking data flow as an example, decision predicates that are classified as *retention* predicates, must only be evaluated when the lifecycle is in the *retention* phase (i.e., data is being stored). Decision predicates must be re-evaluated whenever an attribute value changes, and the authorisation decision must be updated accordingly. Moreover, when the authorisation lifecycle transitions to a new phase, decision predicates classified under the new phase must be evaluated even if no attribute updates occurred.

Decisions Decisions refer to the evaluation outcome of decision predicates, which determines whether authorisations can be granted or not. Our model extends the classical two-valued decisions (i.e., *permit*, *deny*) to n-valued decisions to cover additional values such as *error*, *admissible* or *not admissible*.

Directives Directives are mandatory or optional actions that need to be performed as part of the authorisation, including subject obligations, attribute updates, and other procedures defined by policies. For example, a directive may specify that a smart lock must be unlocked if the authorisation to enter a building is granted. This allows taking proactive actions as part of the authorisation.

Lifecycle Controllers Lifecycle controllers are rules that drive the authorisation lifecycle and define transitions between phases. They are if-else statements that determine the next phase of the lifecycle based on its current phase, the authorisation decision and other conditions (e.g., about attributes or directives). As such, lifecycle controllers must be able to capture contextual information and drive the lifecycle accordingly. A pseudo-code example of a lifecycle controller is shown in Listing 1.1 Lifecycle controllers must be specified by the system administrator only, and policy authors must classify their policies based on the phases defined in the lifecycle controllers. Thus, administrators configure the enforcement system with a set of authorisation lifecycles each targeting a specific technology or use-case, while policy authors use these lifecycles by defining policies for each of the corresponding phases.

Evaluation Function The evaluation function is the component that makes authorisation decisions (e.g., *permit*, *deny*), and determines directives. It takes

Listing 1.1. Lifecycle expressed as if-else statements

```
If currentPhase is ''phaseA'' and authDecision is ''permit'' and directives are enforced
  Then change phase to ''phaseB''
Else If currentPhase is ''phaseA'' and authDecision is ''permit''
  Then change phase to ''phaseC''
Else change phase to ''exit''
```

the current lifecycle phase as input, then evaluates decision predicates that are classified under the current phase by retrieving and evaluating attributes. The output of the evaluation outcome is an authorisation decision combined with a set of directives. The evaluation function must be invoked whenever the phase changes or an attribute value is updated.

Transition Function The transition function is the component that drives the lifecycle and changes the phase. It takes the current phase as input, then evaluates lifecycle controllers to determine the next phase in the lifecycle. Lifecycle controllers may include conditions about attributes or about the outcome of the evaluation function, so attributes, decision and directives are taken as input in the transition function. The transition function must be invoked right after every evaluation by the evaluation function.the The transition function may also be invoked upon attribute updates, if the lifecycle relies on attributes and needs to react to changes.

4.2 Evaluation Flow

When an authorisation request is created, the evaluation starts by calling the transition function to select the appropriate lifecycle. The transition function evaluates lifecycle controllers and sets the first phase in the lifecycle. Thereafter, the evaluation function is called to evaluate decision predicates classified under the selected phase, and evaluation outcome is enforced. The transition function is then invoked again to determine determine whether the lifecycle must transition to another phase or remain in the same phase. If the phase does not change, the system waits for attribute updates or user input (e.g., end authorisation), upon which the evaluation function is invoked again. Otherwise, decision predicates classified under the new phase are evaluated and the new decision and directives are enforced. The evaluation of decision predicates is again followed by calling the transition function and so on until the end of the authorisation. To sum up, every time a transition to a different phase occurs, the evaluation function will be invoked and the authorisation decision will be updated, then the transition function will be called again to check whether the phase must change. As an example, Figure 2 shows a flow chart describing the evaluation flow of a UCON authorisation using our model.

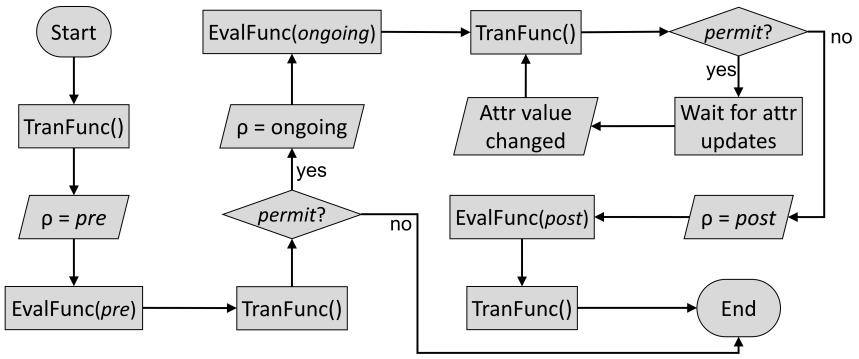

Fig. 2. Evaluation flow of a UCON authorisation

4.3 Authorisation Process

We use the term *Authorisation Process* to refer to all events, evaluations and other elements belonging to the lifetime of a single authorisation. An authorisation process is initiated by an *Authorisation Request*, and ends when its lifecycle reaches an exit phase (e.g., lifecycle of safety-critical use-cases specifies that the authorisation process ends upon completion of safety measures not upon revocation). Since the authorisation processes start with a request, each authorisation process is associated with, and uniquely identified by, the request that started it. An authorisation request is a query to check whether a *specific subject* is allowed to exercise a *specific right* on a *specific object*. Thus, it can be defined by the tuple $Rq := (s, o, r)$, where s is the subject, o is the object, and r is the access right. While the focus of this paper is on authorisations only, our objective is to cover policy-based systems that incorporate other lifecycles, such as the credential exchange system we introduced in [10]. The request in such systems is not a query about a *subject* exercising a *right* on an *object*, but rather a query about the capabilities that a specific subject has within a specific system and context. For this reason, we define the request as an assignment to a subset of attributes, which can be subject, object and/or environmental attributes. Let A be the set of all attributes and V be the set of all possible values for A. Also, let A_R be the subset of attributes that are present in the request such that $A_R \subset A$. We define the request as $Rq : A_R \to V$ a partial mapping from A_R to V representing the assignment of values to the request attributes.

5 Preliminary and Future Work

In this section, we describe our preliminary work to realise our model as well as example lifecycles. We also lay out a roadmap of future actions to finalise the model and address open problems.

5.1 Groundwork

We introduced, in [10], an optimised implementation[4] of UCON using Abbreviated Language For Authorisation (ALFA) [17] as a baseline policy language. In this work, we modified our implementation to decouple the authorisation lifecycle from the authorisation process allowing the lifecycle to be configurable. We added a component that evaluates lifecycle controllers and changes the phase accordingly. ALFA rules incorporate condition and obligation expressions, which can be used to express if-else statements, so we used ALFA to express lifecycle controllers as shown in the example in Listing 1.2. Complete examples are also hosted on GitLab[5].

Listing 1.2. Example of lifecycle rule expressed in ALFA

```
policy lifecycle {                                                                    1
    ...                                                                               2
    rule ruleAtoB {                                                                   3
        target Attributes.session.phase == "phaseA"                                   4
        condition Attributes.session.decision == "permit"                             5
        permit                                                                        6
        on permit {                                                                   7
            obligation obligationAtoB { Attributes.session.phase = "phaseB" }         8
    }}                                                                                9
    ...                                                                               10
};                                                                                    11
```

5.2 Example Lifecycles

UCON We constructed UCON using our model by modelling its lifecycle as the DFA shown in Figure 3. The lifecycle starts with the *pre* phase when access is requested. If the evaluation outcome of *pre* rules is *deny*, the authorisation ends. Otherwise, the lifecycle moves to the *ongoing* phase and remains there as long as re-evaluations of *ongoing* rules result in *permit*. Once an *ongoing* re-evaluation results in *deny*, the lifecycle moves to *post* where it enacts *post* rules and exits. We expressed UCON's lifecycle using ALFA, which is hosted on GitLab [5], alongside other artefacts.

Extended UCON As aforementioned, emerging technologies require continuous control before granting and after revoking authorisation. They may also need to temporarily suspend authorisations rather than completely revoking them. Such requirements cannot be met with UCON because it enacts continuity in the *ongoing* phase only. For this reason, we introduce an extended version of UCON by modifying its DFA as shown in Figure 4 and constructing it using our metamodel. The extended lifecycle does not terminate the authorisation

[4] Implementation of UCON model by Huawei German Research Centre
[5] https://gitlab.com/HaririAli/AuthorisationMetamodel

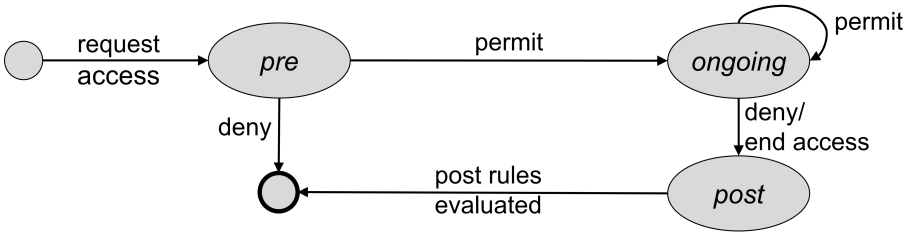

Fig. 3. Lifecycle of a UCON authorisation

process unless all directives (e.g., safety measures) are completed. It also does not revoke access but rather puts it *on hold* as shown in the DFA, which solves the problem of re-evaluation counters introduced by [9]. The lifecycle can be also further extended to capture contextual information and add finer-grained transitions (e.g., move to *on-hold* only if *deny* was caused by trust-level changes; revoke access otherwise). This demonstrates the extensibility of our model, and how it can enact different lifecycles to address different requirements.

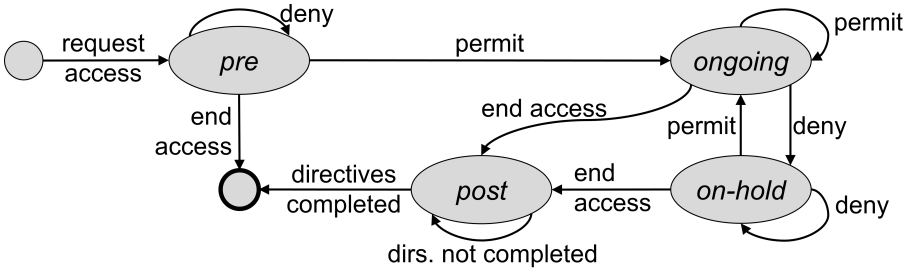

Fig. 4. Lifecycle of extended UCON authorisation

Data Flow Control Data protection and data-centric authorisation is another example that can be covered by our model. Regulations such as the General Data Protection Regulation (GDPR) define a data lifecycle that consists of five phases, namely *collection*, *retention*, *access*, $transfer$ and *destruction*, where different policies and rules apply at each phase For instance, *retention* policies may specify that data can be retained for up to three months or until the data subject withdraws consent. Subsequently, the lifecycle must move to the *destruction* phase, where policies specify that the data must be deleted and derivative data must be anonymised. We model the data-flow lifecycle as a DFA shown in Figure 5, which can be expressed and enacted by our model. The data-flow lifecycle may be more complicated than the provided DFA example and it may require capturing contextual or other information.

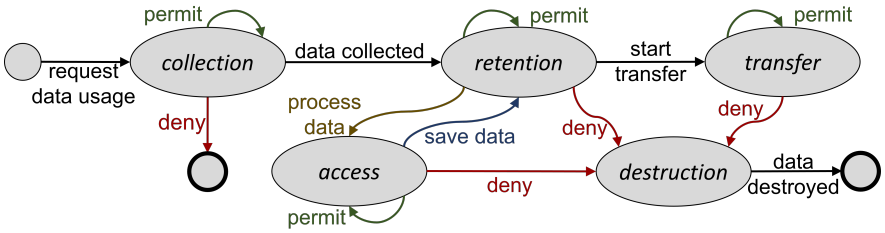

Fig. 5. Lifecycle of a data flow authorisation

5.3 Future Directions

To complete our work and further develop the model, we plan to conduct a formal analysis and formalise the model. We also plan to carry out more thorough and extensive case studies of the above examples, and to explore other use-cases. This allows us to identify corner cases that have not been addressed. Finally, we aim to solve three open problems identified and described below.

Firstly, we defined the authorisation request in Section 4.3 as an assignment of values to a subset of attributes designated as *request attributes*. Request attributes must not be arbitrary as they should reflect a complete and legitimate evaluation request. However, we cannot predetermine what kind of attributes must be request attributes as this changes from one use-cases to another. Therefore, the model must add a layer of abstraction that allows users to express what constitutes a legitimate request.

Secondly, our model assumes that *all* attributes are *mutable* including request attributes. However, value changes in some request attributes may reflect a significant change in the request, which makes authorisation process incoherent as it would not relate consistently to its previous phases and states anymore. An example of such changes is a value change in the `location` attribute when the *subject is expected to be immobile*. This obviously does not apply to all request attributes, as value changes in other request attributes do not necessarily indicate a significant change in the request context. For example, a value in change in `user_balance` would be acceptable as it does not reflect a significant change in the request. We can argue that identifier attributes (e.g., subject ID, object ID) must always be fixed in all use-cases, but such argument cannot always be valid for other attributes. Therefore, a thorough analysis and study is needed to devise a systematic method for specifying which request attributes must be fixed in a particular use-case.

Thirdly, we used ALFA to express lifecycle controllers as if-else statements. However, ALFA was not designed for this purpose and its syntax and semantics do not fully support expressing DFAs. For this reason, we plan to consider and study other languages such as Amazon States Language and Drools [12, 19], or perhaps to devise new language for this purpose if needed.

6 Conclusion

We introduced the concept of authorisation lifecycle, which refers to the different phases in the lifetime of an authorisation. We also proposed a metamodel for continuous authorisation that enables fine-grained and customisable authorisation lifecycles. The foundation of our model is modelling the lifecycle as a DFA describing the lifecycle phases and transitions between them. By defining different DFAs, the model can be instantiated into different versions with different lifecycles each addressing specific requirements. We described preliminary work and specified future directions that include solving the identified open problems, extensive case studies, as well as formal analysis and formalisation of the model.

References

1. Almutairi, A., Siewe, F.: Ca-ucon: a context-aware usage control model. In: Proceedings of the 5th ACM International Workshop on Context-Awareness for Self-Managing Systems. pp. 38–43 (2011)
2. Baldi, G., Diaz-Tellez, Y., Dimitrakos, T., Martinelli, F., Michailidou, C., Mori, P., Osliak, O., Saracino, A.: Session-dependent Usage Control for Big Data. J. Internet Serv. Inf. Secur. **10**(3), 76–92 (2020)
3. Bandopadhyay, S., Dimitrakos, T., Diaz, Y., Hariri, A., Dilshener, T., Marra, A.L., Rosetti, A.: DataPAL: Data Protection and Authorization Lifecycle framework. In: 2021 6th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM). IEEE (2021)
4. Carniani, E., D'Arenzo, D., Lazouski, A., Martinelli, F., Mori, P.: Usage control on cloud systems. Future Generation Computer Systems **63**, 37–55 (2016)
5. Cirillo, F., Cheng, B., Porcellana, R., Russo, M., Solmaz, G., Sakamoto, H., Romano, S.P.: Intentkeeper: Intent-oriented data usage control for federated data analytics. In: 2020 IEEE 45th Conference on Local Computer Networks (LCN). pp. 204–215. IEEE (2020)
6. Conti, M., Crispo, B., Fernandes, E., Zhauniarovich, Y.: Crêpe: A system for enforcing fine-grained context-related policies on android. IEEE Transactions on Information Forensics and Security **7**(5), 1426–1438 (2012)
7. Daniele Micciancio: Deterministic Finite Automata (DFA) (2015), https://cseweb.ucsd.edu/classes/sp15/cse191-e/lec1.html
8. Davies, H.: Ted Cruz using firm that harvested data on millions of unwitting Facebook users. the Guardian **11**, 2015 (2015), https://www.theguardian.com/us-news/2015/dec/11/senator-ted-cruz-president-campaign-facebook-user-data
9. Dimitrakos, T., Dilshener, T., Kravtsov, A., La Marra, A., Martinelli, F., Rizos, A., Rosetti, A., Saracino, A.: Trust Aware Continuous Authorization for Zero Trust in Consumer Internet of Things. In: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). pp. 1801–1812 (2020). https://doi.org/10.1109/TrustCom50675.2020.00247
10. Hariri, A., Bandopadhyay, S., Rizos, A., Dimitrakos, T., Crispo, B., Rajarajan, M.: Siuv: A smart car identity management and usage control system based on verifiable credentials. In: IFIP International Conference on ICT Systems Security and Privacy Protection. pp. 36–50. Springer (2021)
11. Humayed, A., Lin, J., Li, F., Luo, B.: Cyber-physical systems security—a survey. IEEE Internet of Things Journal **4**(6), 1802–1831 (2017)

12. Inc., A.W.S.: Amazon States Language (2021), https://docs.aws.amazon.com/step-functions/latest/dg/concepts-amazon-states-language.html
13. Katt, B., Zhang, X., Breu, R., Hafner, M., Seifert, J.P.: A general obligation model and continuity: enhanced policy enforcement engine for usage control. In: Proceedings of the 13th ACM symposium on Access control models and technologies. pp. 123–132 (2008)
14. Koo, J., Kang, G., Kim, Y.G.: Security and privacy in big data life cycle: A survey and open challenges. Sustainability **12**(24), 10571 (2020)
15. La Marra, A., Martinelli, F., Mori, P., Rizos, A., Saracino, A.: Introducing usage control in MQTT. In: Computer Security, pp. 35–43. Springer (2017)
16. La Marra, A., Martinelli, F., Mori, P., Saracino, A.: Implementing usage control in internet of things: a smart home use case. In: 2017 IEEE Trustcom/BigDataSE/ICESS. pp. 1056–1063. IEEE (2017)
17. OASIS: Abbreviated language for authorization Version 1.0 (2015), https://bit.ly/2UP6Jza
18. Park, J., Sandhu, R.: The UCONABC usage control model. ACM Transactions on Information and System Security (TISSEC) **7**(1), 128–174 (2004)
19. Proctor, M.: Drools: a rule engine for complex event processing. In: International Symposium on Applications of Graph Transformations with Industrial Relevance. pp. 2–2. Springer (2011)
20. Punagin, S., Arya, A.: Privacy in the age of pervasive internet and big data analytics–challenges and opportunities. International Journal of Modern Education & Computer Science **7**(7) (2015)
21. Rose, S.W., Borchert, O., Mitchell, S., Connelly, S.: Zero trust architecture (2020)
22. Ruan, W., Xu, M., Jia, H., Wu, Z., Song, L., Han, W.: Privacy compliance: Can technology come to the rescue? IEEE Security & Privacy **19**(4), 37–43 (2021)