Supplementary Technical Report

Motivating Example: UCON+ in a Smart Home Environment

Mohammad Asmand Joneghani
Leila Samimi-Dehkordi

November 16, 2025

Abstract

This supplementary report presents the complete motivating example used in the paper "SmartUcon: A Model-Driven Framework for Secure and Privacy-Aware Usage Control in Smart Environments." A smart lock system in a smart home is modeled using the UCON+ framework, demonstrating its three-phase control, trust management, obligations, advice, and delegation. The full ALFA code is provided to highlight the complexity of manual policy authoring.

Contents

0.1 Introduction and Scenario

To illustrate the practical challenges of usage control in smart environments, consider a smart lock system as a representative example. In such environments, maintaining continuous security assurance and preventing unauthorized data exposure are critical concerns [4]. Traditional UCON models primarily focus on the ongoing phase of usage, providing continuous monitoring but offering limited support for pre-authorization checks, post-usage revocation, or trust-based decision-making. Moreover, they provide little support for adaptive threat response or privacy-aware policy enforcement [1, 6]. UCON+, in contrast, introduces hierarchical policy structures, three-phase control (pre, ongoing, post), trust management, and delegation mechanisms, making it more suitable for highly dynamic IoT environments such as smart homes.

UCON+ extends this with:

- Three-phase control (pre, ongoing, post)
- Hierarchical policies
- Trust management
- Delegation

0.1.1 Scenario: Smart Lock Access Control

The UCON+ architecture introduces a dynamic access control model with three distinct, interconnected phases (see Figure 1): Pre-Authorization, Ongoing, and Post-Authorization. As an advanced extension of the UCON model, UCON+ begins with a Pre-Authorization phase where access requests are evaluated against a comprehensive set of attributes, including user identity, location proximity, time constraints, and a behavioral trust score. This ensures that authorization decisions are not only context-aware but also security-driven, minimizing the risk of unauthorized access [5]. If these conditions are met, initial access is granted, and the event is logged.

The system then transitions to the Ongoing phase, where it continuously monitors contextual conditions. If any anomaly or policy violation occurs, access is securely revoked, and all events are recorded for accountability and audit purposes. Access may be revoked if any condition is violated, triggering automated responses such as notifications to the homeowner and adjusting the lock's operational mode. Finally, the Post-Authorization phase ensures secure and verifiable revocation of all access rights after a designated period, with any subsequent access attempts flagged as suspicious via threat detection and documented in a persistent log. This end-to-end monitoring cycle contributes to continuous information security and trust maintenance within the smart home ecosystem [3]. This holistic approach, from initial verification to post-access threat analysis, provides a more robust and adaptive security framework for smart environments than traditional models.

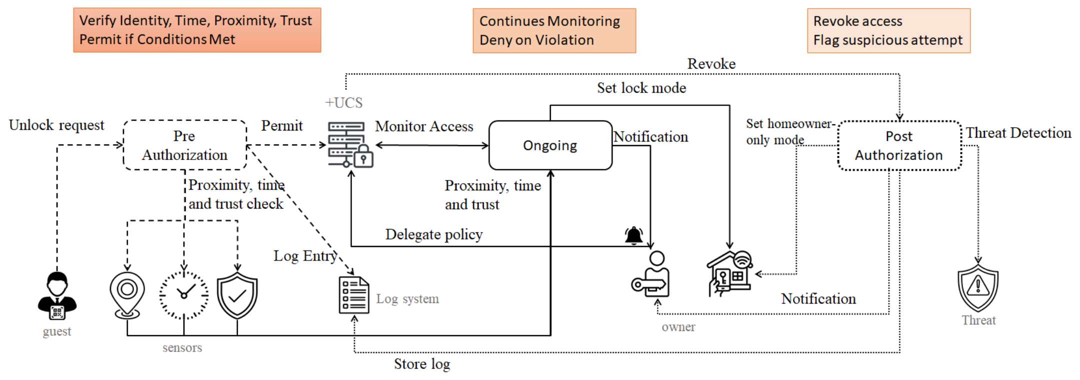

Figure 1: Control Flow of UCON+ Phases for Smart Lock

While the smart lock example demonstrates the expressive power of UCON+, it also reveals a critical limitation: writing such policies directly in a low-level policy language like ALFA is non-trivial. From a security engineering perspective, manual ALFA authoring may also introduce inconsistencies or vulnerabilities if conditions or obligations are incompletely defined [2]. ALFA requires precise syntax, knowledge of attributes, conditions, obligations, and advice, which can be overwhelming for non-specialists such as system administrators or domain experts in health-care, transportation, or smart homes. Even for experienced developers, manual specification of complex UCON+ policies is error-prone and time-consuming, especially when policies must be continuously adapted to changing contexts. This complexity highlights the need for a higher-level abstraction: a Domain-Specific Modeling Language that simplifies policy specification, validates models against a metamodel, and automatically generates executable ALFA code.

In the following, we present the ALFA code for each phase of the smart lock scenario to emphasize the complexity and low-level nature of manual policy specification.

0.2 UCON+ Initialization

A smart lock is installed on the main entrance of a home and connected to the IoT network. The lock should only be opened by authorized individuals (e.g., family members or temporary guests), and access should be granted under specific contextual conditions (e.g., time, location, or trust level). In Listing 1, the following concepts are initialized:

- **Subject:** A guest invited for a one-week stay.
- **Resource:** The smart lock on the main entrance door.
- **Action:** The unlock operation requested by the guest.
- **Environment:** A smart home with location sensors, time constraints, and identity recognition systems.
- **Objective:** Ensure that the guest can open the lock only during specific hours (e.g., 8:00 AM–8:00 PM) and only when the homeowner is nearby.

```
namespace smart.home {
  // ---- Subject ----
  attribute subject.userId {
    category = subject;
    id = "urn:smart.home:subject:userId";
    type = string;}

  attribute subject.role {
    category = subject;
    id = "urn:smart.home:subject:role";
    type = string;}

  attribute subject.trustLevel {
    category = subject;
    id = "urn:smart.home:subject:trustLevel";
    type = double;}

  // ---- Resource ----
  attribute resource.deviceId {
    category = resource;
    id = "urn:smart.home:resource:deviceId";
    type = string;}

  attribute resource.deviceType {
    category = resource;
    id = "urn:smart.home:resource:deviceType";
    type = string;}

// ---- Action ----
  attribute action.actionId {
    category = action;
    id = "urn:smart.home:action:actionId";
    type = string;}

  // ---- Environment ----
  attribute environment.currentTime {
    category = environment;
    id = "urn:smart.home:environment:currentTime";
    type = time;}

  attribute environment.ownerProximityMeters {
    category = environment;
    id = "urn:smart.home:environment:ownerProximityMeters";
    type = double;}

  attribute environment.accessNotAfter {
    category = environment;
    id = "urn:smart.home:environment:accessNotAfter";
    type = dateTime;}

  attribute environment.now {
    category = environment;
    id = "urn:smart.home:environment:now";
    type = dateTime;}
```

Listing 1: Initialization of Smart Lock Scenario

0.3 UCON+ Operation

UCON+ operates in three phases—pre-authorization, ongoing, and post-authorization—enhanced with advanced capabilities. The following paragraphs illustrate these phases using the smart lock example.

0.3.1 Pre-Authorization Phase

Listing 2 demonstrates the pre-authorization phase:

- **Attribute Verification:**
  - Guest identity is verified via the smart home application (e.g., QR code or biometric authentication).
  - *Trust level* of the guest is assessed through a trust evaluation system analyzing past behavior.
  - Environmental conditions are checked: Is the current time within 8:00 AM–8:00 PM? Is the homeowner within 5 meters of the door (via location sensors)?
- **Decision:** If all conditions (identity, trust, time, location) are satisfied, initial access is granted.
- **Obligation:** The guest must log their entry (e.g., confirm via the application).

```
policyset SmartLockPolicies {
  apply permitOverrides

  target
    clause resource.deviceType == "smart-lock";

  policy GuestUnlock {
    apply denyOverrides

    target
      clause action.actionId == "unlock"
        and subject.role == "guest"
        and subject.userId == "Guest1";

    rule PreAuth {
      effect Permit
      condition
        subject.trustLevel >= 0.70
        and environment.currentTime >= time("08:00:00")
        and environment.currentTime <= time("20:00:00")
        and environment.ownerProximityMeters <= 5.0
        and environment.currentDate <= date("2025-08-23"); // Example
            access period check
      obligation log-entry on Permit;}
  }}
```

Listing 2: Guest Unlock Policy - Pre-Authorization

While this ALFA specification clearly expresses the required policy, it also demonstrates the difficulty of writing such rules manually: multiple attributes, conditions, and obligations must be explicitly defined and carefully maintained.

0.3.2 Ongoing Phase

Listing 3 shows the ongoing phase:

- **Continuous Monitoring:**
  - If the guest attempts to open the lock at 9:00 PM (outside permitted hours), access is revoked.

  - If the homeowner leaves the vicinity, guest access is temporarily suspended.
  - If the guest's trust level decreases (e.g., due to attempted unauthorized access to another device), access is terminated.

- **Automated Response:** The system may notify the homeowner or switch the lock to "homeowner-only" mode.

```
      rule Ongoing_TimeWindowBreak {
        effect Deny
        condition
          environment.currentTime < time("08:00:00")
          or environment.currentTime > time("20:00:00");
        advice notify-owner on Deny;
        obligation set-homeowner-only-mode on Deny;}

      rule Ongoing_OwnerAway {
        effect Deny
        condition environment.ownerProximityMeters > 5.0;
        advice notify-owner on Deny;
        obligation set-homeowner-only-mode on Deny;}

      rule Ongoing_TrustDrop {
        effect Deny
        condition subject.trustLevel < 0.70;
        advice notify-owner on Deny;
        obligation set-homeowner-only-mode on Deny;}
    }
```

Listing 3: Ongoing Access Control Rules

These rules illustrate the strength of UCON+ in enabling continuous enforcement during the entire usage session. However, they also highlight the complexity of writing multiple conditions, obligations, and advice statements in ALFA.

0.3.3 Post-Authorization Phase

Listing 4 defines the post-authorization policy:

- After the guest's access period ends (e.g., after one week), UCON+ fully revokes access.
- **Post-actions:** The lock returns to its default (homeowner-only) mode, and all access logs are stored.
- Any attempt by the guest to open the lock after the expiry is flagged as suspicious using *threat intelligence*, and the homeowner is alerted.

```
policy PostAuthorization {
  apply denyOverrides

  target
    clause subject.role == "guest"
      and action.actionId == "unlock";

  rule ExpiredAccess {
    effect Deny
    condition environment.now > environment.accessNotAfter;
    obligation revoke-guest-token on Deny;
    obligation set-homeowner-only-mode on Deny;
    obligation store-access-log on Deny;
    advice flag-suspicious on Deny
  }}
```

Listing 4: Post-Authorization Policy

This phase ensures that access rights are securely revoked once the guest's authorization period ends and that all subsequent attempts are properly logged and flagged. While this provides strong guarantees, the manual specification of revocation, log storage, and suspicious activity detection in ALFA requires multiple low-level constructs.

0.4 Obligations and Advice

Listing 5 shows how obligations and advice can be incorporated into UCON+ policies. In this example, access is granted only if the subject's trust level exceeds a defined threshold. Upon access, an obligation `log-entry()` is triggered, and an advice is issued to notify the owner.

```
namespace smart.home.obl {

  obligation log-entry on Permit {
    assignment { id = "urn:obl:type"; value = "access-log" }
    assignment { id = "urn:obl:event"; value = "unlock" }
  }

  obligation revoke-guest-token on Deny {
    assignment { id = "urn:obl:action"; value = "revoke" }
    assignment { id = "urn:obl:who"; value = "guest" }
  }

  advice notify-owner on Deny {
    assignment { id = "urn:advice:channel"; value = "push" }
    assignment { id = "urn:advice:severity"; value = "warning" }
  }

  advice flag-suspicious on Deny {
    assignment { id = "urn:advice:tag"; value = "threat-intel:follow-up
       " }
  }
}
```

Listing 5: Obligations and Advice

Obligations and advice extend UCON+ beyond simple authorization decisions by attaching actions to both permit and deny outcomes.

Another key concept is the Automated Delegation Profile (ADP), where the homeowner delegates access policy management to the system. Listing 6 presents an ALFA snippet for defining delegation in UCON+. Here, the rule permits a `Homeowner` to delegate access to a `SmartLock`, and creates a delegation profile as an obligation.

```
 policyset AdminDelegation {
   apply denyOverrides

   policy OwnerDelegatesGuest {
     apply denyOverrides

     target
       clause subject.role == "owner"
```

```
        and resource.deviceType == "smart-lock";

    rule DelegateGuestAccess {
      effect Permit
      condition subject.userId == "Homeowner1";}

    rule RevokeGuest {
      effect Permit
      obligation revoke-guest-token on Permit;}
  }}
```

Listing 6: Delegation Policy

Delegation further exemplifies the flexibility of UCON+, allowing homeowners to transfer control responsibilities to the system or to other users.

The smart lock example highlights the main differences between UCON and UCON+. While the original UCON model mainly focuses on the ongoing phase of access and lacks robust mechanisms for pre- and post-authorization monitoring, trust evaluation, and delegation, UCON+ extends these capabilities by covering all three phases, integrating trust and threat intelligence, and supporting delegation management. Although these features significantly increase expressiveness and suitability for IoT environments, they also make policy specification in low-level languages such as ALFA complex and error-prone. This motivates the adoption of a Domain-Specific Modeling Language, which enables intuitive, graphical modeling of UCON+ policies. A DSML abstracts away low-level syntax, ensures validation against a metamodel, and automatically generates executable ALFA code, thus bridging the gap between the expressive but technically demanding UCON+ framework and the practical needs of real-world system designers.

0.5 Conclusion

The smart lock example demonstrates UCON+'s expressiveness but also the **high complexity** of manual ALFA authoring. This motivates **SmartUcon**: a DSML for graphical modeling and automated code generation.

Bibliography

[1] Abdulgader Almutairi and François Siewe. CA-UCON: a context-aware usage control model. In Tomoko Yonezawa and Waltenegus Dargie, editors, *CASEMANS@Ubicomp '11: The 5th ACM International workshop on context-awareness for self-managing systems, Beijing, China, 17 September 2011*, pages 38–43. ACM, 2011.

[2] Athareh Fatemian, Bahman Zamani, Marzieh Masoumi, Mehran Kamranpour, Behrouz Tork Ladani, and Shekoufeh Kolahdouz Rahimi. Automatic generation of xacml code using model-driven approach. In *2021 11th International Conference on Computer Engineering and Knowledge (ICCKE)*, pages 206–211, 2021.

[3] Giacomo Giorgi, Antonio La Marra, Fabio Martinelli, Paolo Mori, Athanasios Rizos, and Andrea Saracino. Exploiting if this then that and usage control obligations for smart home security and management. *Concurr. Comput. Pract. Exp.*, 34(16), 2022.

[4] Zhaoyang Han, Liang Liu, and Zhe Liu. An efficient access control scheme for smart lock based on asynchronous communication. In *Proceedings of the ACM Turing Celebration Conference - China, ACM TUR-C 2019, Chengdu, China, May 17-19, 2019*, pages 61:1–61:5. ACM, 2019.

[5] Ali Hariri, Amjad Ibrahim, Bithin Alangot, Subhajit Bandopadhyay, Antonio La Marra, Alessandro Rosetti, Hussein Joumaa, and Theo Dimitrakos. Ucon+: comprehensive model, architecture and implementation for usage control and continuous authorization. In *Collaborative Approaches for Cyber Security in Cyber-Physical Systems*, pages 209–226. Springer, 2023.

[6] Rong Jiang, Xue Chen, Yimin Yu, Ying Zhang, and Weiping Ding. Risk and ucon-based access control model for healthcare big data. *J. Big Data*, 10(1):104, 2023.