SIUV: A Smart car Identity management and Usage control system based on Verifiable credentials

Ali Hariri[1,2], Subhajit Bandopadhyay[1,3], Theo Dimitrakos[1], Bruno Crispo[2], Muttukrishnan Rajarajan[3], and Athanasios Rizos[1]

[1] Munich Research Center, Huawei Technologies Duesseldorf GmbH, Munich, Germany, {name.surname}@huawei.com
[2] Department of Computer Science and Information Engineering, University of Trento, Trento, Italy{name.surname}@unitn.it
[3] Institute for Cyber Security, School of Mathematics, Computer Science and Engineering, City, University of London, London, United Kingdom{name.surname}@city.ac.uk

Abstract. The automotive industry is witnessing an accelerated growth in digital innovations that turn modern vehicles into digital systems. Such innovations include services that entertain, support and guide drivers and passengers such as fleet management, autonomous driving, shared mobility and private renting. This makes the security of modern vehicles a crucial concern as they have evolved into cyber-physical and safety-critical systems. Therefore, a continuous and stateful monitoring of the usage of vehicular resources has become a paramount requirement. Indeed, several Identity and Access Management (IAM) frameworks have been proposed in the automotive field, but context awareness and continuity of control remain overlooked. To address these challenges, we present a stateful smart-car IAM that is based on Usage Control (UCON) and Verifiable Credentials (VCs) to protect in-car resources. The system uses Attribute Based Access Control (ABAC) policies to issue privileges to subjects (i.e. drivers, passengers or applications) according to their credentials and claims. The issued privileges are then used to decide whether to grant or deny access to in-car resources. Furthermore, the system continuously monitors subject claims, resource attributes and environmental conditions (e.g. location or time). Hence, if a change occurs, the system re-evaluates policies and updates or revokes issued privileges and usage decisions accordingly. In addition, the system enforces the Principle of Least Privilege (PoLP) by only asking for the minimum privileges that are required to make a decision about a particular action. Finally, we describe the architecture and implementation of the proposed IAM, discuss the evaluation results, and define future directions.

Keywords: UCON · IAM · Automotive · Smart Car · Verifiable Credentials · Principle of Least Privilege · ABAC · ALFA · XACML · Ed25519

1 Introduction

The exponential growth of the Internet of Vehicles (IoV) innovations, like autonomous driving, driver assistance, vehicle connectivity, fleet management, infotainment, shared mobility, and online car applications has recently gained a notable attention in the automotive industry. Original Equipment Manufacturers (OEMs), together with an increasing number of third-party software providers, are expected to deliver such services and applications in smart cars. Such services and innovations are supported by the integration of smart sensors as well as the shift from electromechanical to interconnected software-based systems in modern vehicles, which is turning modern vehicles into sources of valuable information. Although such services improve the convenience and experience of passengers, they make modern vehicles a target for cyber-attacks and hence introduce a variety of security, safety and privacy threats and risks [8][6].

Many white hat hackers and security researchers have demonstrated different attacks on modern vehicles. For instance, Miller and Charlie [17] performed an attack that allowed them to gain access to critical physical systems such as steering and braking systems. They proved that the attack can be executed remotely without any physical interaction with the vehicle, and that many vehicles are vulnerable. As a result, few millions of cars had to be recalled from the market. Similarly, Dürrwang et al. [10] exploited a vulnerability that allows an unintended deployment or a prevention of deployment of airbags. More recently, Wouters et al. [23] uncovered a vulnerability that allowed them to clone a key fob of a Tesla Model S in less than two minutes. Researchers have also studied the state of the art of IoV security and defined open problems and challenges that need to be addressed [21][3]. Both [3] and [21] state that context awareness is an important aspect in vehicle access control solutions that remains overlooked. This happens because the physical and environmental conditions of the vehicle are crucial and must be taken into consideration upon making authorisation decisions. They also emphasise that the Principle of Least Privilege (PoLP) is another important challenge that must be addressed in order to mitigate insider threats and eliminate the risk of unintended or malicious use of unnecessary capabilities. Furthermore, since vehicles are mobile systems, their environmental conditions change continuously as they move. Thus, access to vehicular resources must be monitored and controlled continuously to ensure a correct, safe and secure usage as circumstances change. Different in-car resources are manufactured by different OEMs that are not in full control of third-party applications and cannot predict their behaviour. Consequently, such applications cannot be trusted unconditionally, but their attributes must be continuously verified.

To address the aforementioned challenges, we propose SIUV: a context-aware Identity and Access Management (IAM) system for smart cars. It is based on the Usage Control (UCON) model that allows context-aware, stateful and continuous access control, and supports dynamic and codeless behaviour based on policies. The system also enforces the PoLP and uses Verifiable Credentials (VCs) to authenticate subjects (i.e. drivers, passengers and in-car applications) and verify their claims. Accordingly, the **contributions** of this work are as follows:

(1) We introduce Usage Control System Plus (UCS+), an optimised, efficient and modular implementation of the UCON model. (2) We introduce a stateful and dynamic Security Token Service (STS), based on UCS+, that accepts and verifies VCs and issues privileges according to UCON policies. The STS monitors environmental conditions and subject attributes, re-evaluates policies if a change occurs, and revokes privileges if they become invalid. (3) We define and implement the "allApplicable" rule combining algorithm that allows UCON policies to be used for issuing privileges rather than making access decisions. (4) We propose a vehicular IAM architecture that relies on a centralised STS and distributed UCS+ instances to protect in-car resources, enforce the PoLP and support zero-trust principles.

The rest of the paper is organised as follows: Section 2 outlines related work and presents the theoretical and technical background. We describe the details of the proposed IAM system and the implementation in Section 3. An example use case as well as the evaluation of the system are discussed in Section 4. Finally, we define future directions and draw conclusions in Section 5.

2 Background

This section outlines related work and presents the theoretical and technical background of Smart-car Identity management and Usage control system based on Verifiable credentials (SIUV).

2.1 Related Work

Hamad et al. [12] introduced a policy-based authentication and authorisation framework for vehicular communications. They used a trust management model as the core of their framework. When an application needs to connect to a remote entity, it makes a request to the trust management system. The system evaluates relevant policies and makes a decision based on the context. The framework introduces a performance overhead over network connections, but this only affects new connections as the policies are only evaluated during the setup phase. This means that the framework does not enforce continuous control and does not take contextual changes into consideration once a connection is established.

A dynamic access control model for modern vehicles based on dynamic Mandatory Access Control (MAC) is proposed in [14]. The model uses a state machine to control the flow of information between vehicular components, and enforces the PoLP and the separation of duties. Although the model supports dynamic access control, it does not address context awareness and continuity of control, since access control decisions do not change after they have been taken.

Kim et al. [16] developed a decentralised access control framework by integrating an Attribute Based Access Control (ABAC) module in AUTOSAR [2] platform. They focused on protecting Electronic Control Unit (ECU) diagnostic interfaces from unauthorised access. Accordingly, diagnostic Controller Area Network (CAN) messages are considered as subjects and their properties are

used as subject attributes, while ECU properties as environmental attributes. Their framework lacks context awareness and continuity of control as it only works with a predefined set of attributes and enforces atomic access control.

Likewise, Gupta and Sandhu [11] proposed an extended and layered access-control-oriented architecture for the IoV. They also provided an authorisation framework that secures the dynamic intra- and inter-communications between vehicles and vehicle components. They discussed and described how different access control models can be enforced at the different layers of the architecture. This framework also lacks continuity of control, it does not enforce the PoLP.

Ammar et al. [1] developed an end-to-end Role Based Access Control (RBAC) mechanism that protects vulnerable On-Board Diagnostics-II (OBD-II) ports. These ports impose a serious security risk as they can be used to install software on ECUs and to access the internal vehicle network. Thus, access to vehicle's resources through the vulnerable OBD-II ports is regulated by the proposed mechanism according to RBAC policies. The mechanism is AUTOSAR-compliant and can be installed on existing vehicles without any modifications.

2.2 Access and Usage Control

ABAC [13] is one of several access control models used to protect digital resources. It provides more flexibility and finer-grained control than preceding models as it manages access rights based on attributes of subjects, resources and the environment. ABAC's evaluation semantics, architecture, administration and policy language are defined in a comprehensive standardised reference model known as the eXtensible Access Control Markup Language (XACML) [19]. Although ABAC is fine-grained and flexible, it does not support continuity of access and mutability of attributes during evaluation. For this reason,the UCON model was proposed by Park and Sandhu [20] as a generalisation that goes beyond ABAC and other models to support context awareness, mutability of attributes and continuity of control. UCON continuously monitors attribute values and re-evaluates policies when a change occurs in order to guarantee that access rights still hold whilst usage is still in progress. The model categorises policy decision predicates as "*pre*", "*ongoing*" and "*post*". "*pre*" predicates are evaluated when an access request is made and before access starts; "*ongoing*" predicates are evaluated during usage and particularly when an attribute changes; and "*post*" predicates when access has ended or has been revoked. In addition, UCON introduces obligations and advice; such that obligations are actions that must be fulfilled, whereas advice refers to actions that are recommended. The novelties of UCON make it an excellent baseline for dynamic applications, such as IoV, where the context changes continuously and resource usage is long-lived.

2.3 Abbreviated Language For Authorisation (ALFA)

Abbreviated Language For Authorisation (ALFA) [18] is a pseudocode domain-specific policy language that maps directly to XACML without adding any new semantics. It is much less verbose than XACML and thus more human readable

and shorter in size. Like XACML, ALFA is expressed in a nested hierarchy of elements, namely policy sets, policies and rules. A policy set consists of one or more policy sets or policies, and a policy consists of one or more rules. Each element contains a target clause that determines the applicability of the element according to attribute values. An element may also include obligation or advice expressions that define actions to be taken after evaluation. Each rule incorporates a decisions (i.e. *permit* or *deny*) and may include a condition expression that refines the applicability. Condition expressions are complex expressions such as functions, mathematical operations or comparison operators. Like XACML, ALFA also relies on combining algorithms that resolve conflicts when multiple policies or rules are applicable. Combining algorithms are defined in policy sets and policies using the *apply* keyword. A non-inclusive generic example of an ALFA policy is shown in Listing 1.1.

Listing 1.1. Non-inclusive example of an ALFA policy

```
policy policyName {
    target clause Attributes.attribute1 == ''value''
    apply firstApplicable
    rule ruleName {
        target clause Attributes.attribute2
        condition Attributes.attribute3 + Attributes.attribute4 == 2
        permit
        on permit {
            obligation obligationName {
                Attributes.attribute5 = 5
} } } };
```

2.4 Verifiable Credentials (VCs)

Verifiable Credentials are digital identifiers that provide cryptographic proofs to support the validity and reliability of a claim. Issuers can issue VCs about any subject and make claims that are illustrated with some contextual information about the claim value. The W3C VC data model [22], which we use in our work, specifies the roles of claims, credentials, presentations etc. to help conform to a common structure. We construct these claims as privileges, which will enable a subject to use authorised car resources and communicate with other subjects which requires continuous information flow. If we consider a VC as a presentation graph, it could consist of different credentials which are linked to each other contextually and packaged together to give a unique presentation by the holder. Such a VC is cryptographically verifiable and tamper-resistant. The set of claims that a VC will make on behalf of the issuer or multiple issuers can refer to a common or multiple subjects too, thus making the case for multiple identity properties be used for a distinct purpose. The VC information flow allows an issuer to issue credentials to the holder. The credentials are stored in a wallet and the holder decides whether to present the information in the form of a credential or presentation to the verifier. Every credential or presentation comes attached with a proof mechanism which helps the verifier to verify the authenticity of the VC with the Verifiable Data Registry (VDR).

3 Smart-car Identity management and Usage control system based on Verifiable credentials (SIUV)

In this section, we describe SIUV, a vehicular identity and access management system based on UCON and VCs.

3.1 Usage Control System Plus (UCS+)

We introduce UCS+, a modular and optimised UCON framework that conserves a full ABAC baseline model and uses ALFA policy language. UCS+ introduces context and continuity of access by adding an authorisation session that incorporates an implicit temporal state. The temporal state is captured by classifying policy rules as "*pre*", "*ongoing*" and "*post*" rules. This is achieved using an internal attribute that is updated when the session transitions between these three states. More importantly, the modularity of UCS+ allows additional evaluators to be incorporated. For instance, a trust level evaluator can be added to UCS+ to evaluate the level of confidence in collected attributes or to evaluate the level of trust upon authenticating a subject. UCS+ leverages the publish/subscribe pattern to maximize concurrency between policy parsing, attribute retrieval and policy evaluation. This maximizes performance and minimizes dependency on low network latency or high availability of computational resources, while improving the ability to upgrade or substitute component services and migrate to a distributed deployment where necessary. Figure 1 illustrates the architecture of UCS+, which consists of 8 major components as follows: **Context Handler (CH)** is the core component that receives access requests, handles authorisations and manages their workflow. **Message Bus** supports communications between components using the Publish/Subscribe (Pub/Sub) pattern. **Policy Enforcement Point (PEP)** is the interface of Usage Control System (UCS)+ and the component that protects resources. It creates access requests, invokes the CH and enforces decisions. **Policy Decision Point (PDP)** is the component that evaluates policies and makes access decisions. **Policy Administration Point (PAP)** stores and manages policies and is used by the PDP to retrieve applicable policies. **Policy Information Point (PIP)** defines where to find attributes and how to monitor them. **PIP Registry** manages PIPs and defines which PIPs are responsible for which attributes. **Session Manager (SM)** manages and keeps track of all ongoing sessions to support the continuity of control. **Attribute Table (AT)** is a cache of attribute values and other metadata. **Obligation Manager (OM)** handles and manages policy and rule obligations.

3.2 SIUV Architecture

To support zero-trust principles, we introduce an STS that always verifies subject credentials and claims, and exchanges them with contextualised and short-lived privileges(i.e. expire after the scope of usage). We also introduce localised UCS+ instances that protect localised resources using privileges issued by the STS. This

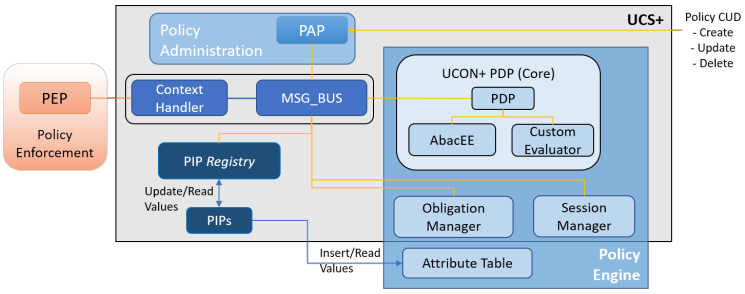

Fig. 1. UCS+ architecture

allows the IAM to enforce the PoLP and to avoid verifying the same credentials every time a subject needs to access a resource. The details of the STS, the localised UCS+ instances and other components of the architecture are shown in Figure 2 and described in the following subsections.

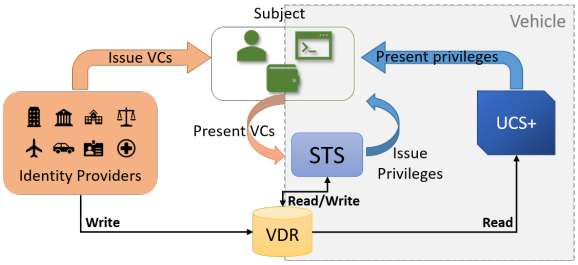

Fig. 2. Architecture of the proposed IAM system

Identity Providers (IdPs): IdPs are authorities that issue and assert claims about drivers, passengers, applications or in-car resources. They issue such claims in the form of VCs, which include proofs that guarantee their integrity. An example of an issuer is the department of motor vehicles that issues driving licenses, or an OEM that asserts claims about in-car components and services.

Subject: A subject could be a driver, a passenger or a software application that needs to access and use in-car resources, and is represented by a digital wallet that holds the corresponding VCs. The digital wallet is managed and controlled by the subject, but its operations can be automated (e.g. the wallet can automatically present VCs when needed). It interacts with the issuers, the STS and UCS+ instances to request and present VCs and privileges.

VDR: The VDR is a distributed hashtable that holds revocation lists and issuers' public keys. The VDR may also be used to store other relevant information like metadata about proofs or schemas and structures of VCs.

STS: The STS is the core component of the system that grants and denies privileges to subjects. It verifies and decodes VCs presented by subjects and issues contextualised privileges in exchange. The STS is based on UCS+, thus it uses ALFA policies to define how to issue privileges according to the context and to subject attributes. The STS also monitors subject and context attributes and re-evaluates policies when a change occurs. If privileges that were already issued are not valid anymore, the STS revokes them and updates the VDR.

The combination of VC and STS protects against any possible identity theft since the privileges issued by the STS are protected from manipulation. This also allows *unlinkability*, thus protecting against any sharing instances of these credentials be linked. This maximizes privacy for the same wallet which shares credentials many times with the STS and thereby removing any chance for the STS to identify any subject information from the credential sharing instances. Since a VC is cryptographically verifiable and rejects any manipulation, they are not vulnerable to any phishing attack.

Localised UCS+ instances: UCS+ instances are used to protect resources such as domain controllers or ECUs. However, they do *not* control the low level behaviour of such components as this imposes a safety risk. Rather, they only control access and usage of high level APIs, services and functions exposed by such components. UCS+ instances rely on ALFA policies to make access decisions according to environmental and resource attributes as well as privileges issued by the STS. They also monitor such attributes and privileges and re-evaluate policies upon changes then update the access decision accordingly. UCS+ instances also enforce the PoLP by asking only for the minimum set of privileges required for the requested action. Specifically, when a UCS+ instance receives an access request, it selects applicable policies and determines the required privileges to evaluate them, then it asks the wallet for these privileges only.

To enforce the PoLP, we introduced a slight modification to the physical architecture by making the PEP act as the Attribute Manager (AM) of the PIP. Therefore, the PIP invokes the PEP to collect the required privileges and the PEP, in turn, requests these privileges from the wallet. Accordingly, the flow of enforcing the PoLP is shown in Figure 3 and described as follows: (1) the subject sends a request to the PEP; (2) the PEP creates an access request and invokes the CH; (3) the CH determines the required attributes to evaluate the applicable polices and invokes the corresponding PIPs; (4) if the required attribute is a privilege, the PIP asks the PEP for the attribute; (5) the PEP asks the wallet for the required privileges and the wallet presents the privileges if they exist; (6) the PEP verifies that the privileges were issued by the STS, parses them and sends the value back to the PIP; (7) finally, the PIP returns the values to the CH, which invokes the PDP to evaluate the policy and make a decision.

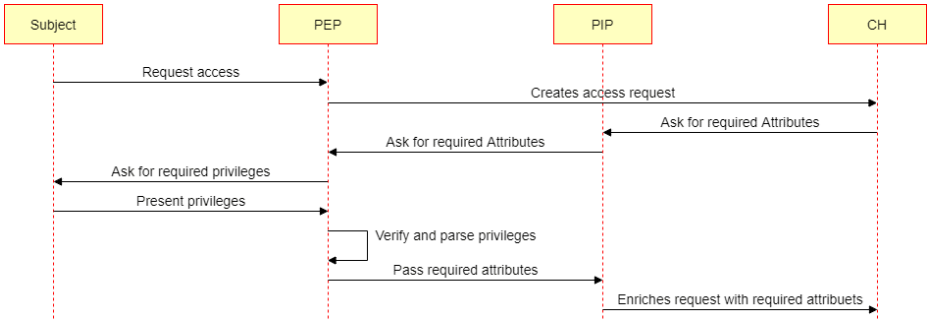

Fig. 3. Sequence diagram of enforcing the PoLP

3.3 STS Policies

In a typical STS, a particular set of privileges may be bound to specific subject attributes. Thus, all subjects that have these attributes will always have the same privileges regardless of the context. In addition, issued privileges do not change even if subject attributes change, which is why tokens issued by an STS are short-lived. This usually results in subjects being either overprivileged or underprivileged. To solve such issues, we use UCS+ as the basis of the STS. This allows the STS to issue different privileges in different situations, and to revoke or update issued privileges if subject or environmental attributes change. This is accomplished using ALFA policies that determine what privileges must be issued according to subject attributes and environmental conditions. We particularly use rule obligations to determine the privileges that must be issued if the corresponding rule applies. ALFA policies include combining algorithms that determine which rule or policy should be enforced when multiple rules or policies apply in a given context. This is fundamental for access decision requests because it is unreasonable to enforce multiple decisions especially if they are contradictory. However, this is not the case for STS policies and rules, because if multiple rules apply in a particular context, all corresponding privileges must be issued. For this reason, we defined and implemented a new combining algorithm, called "*allApplicable*". The algorithm bundles a list of obligations from all the applicable rules and policies at the time of evaluation. The bundled list of obligations is included in the decision that is returned to the PEP by UCS+.

3.4 Implementation

We provide an optimised and highly efficient UCS+ implementation tailored for embedded systems and written in C++. The implementation is modular as new components can be added and existing ones can be extended. We redesigned and implemented the ABAC evaluator in the PDP to use ALFA as the baseline policy language. This is because ALFA is much more compact than XACML and takes much less time to be parsed and evaluated. We modified our implementation

into two sub-frameworks to support the functionalities of the STS and localised UCS+ instances. For the STS, we extended the PEP by adding functions that support issuing, verifying and parsing VCs. The STS PEP also understands and enforces the obligations used in STS policies to issue privileges. In addition, we extended the PIP of the STS to monitor the VDR and re-evaluate the corresponding policies if a VC is revoked. As for the localised UCS+ instances, we extended both the PEP and the PIP to support the PoLP as described in Section 3.2. The extended PIP collects privileges from the extended PEP, which exposes an interface to be used by the PIP to ask for privileges. When the PEP's interface is invoked, the PEP communicates with the wallet to request the privileges.

The W3C VC model [22] specifies that every VC has to have a proof mechanism to support the verifiability of the credentials. In our work, we have used an embedded proof mechanism, and the VC "proof" field mentions the mechanism in place. We use Ed25519 [4], a twisted Edwards curve digital signature algorithm, based on elliptic curve cryptography. As of today, Ed25519 is the most popular instance of EdDSA and is based on the Edwards Curve25519. Although there are many variants of Ed25519-original [4] such as NIST [7], IETF [15] etc. that specifies some refined security properties, we use the Ed25519 instance by LibSodium [9], a widely popular cryptographic library. Brendel et al. [5] discusses the game-based definitions of the security properties of the Ed25519 signature scheme and provably defines Ed25519-LibSodium [9] to be more resilient against key substitution attacks as well as message bound security than other Ed25519 instances [4][7][15].

4 Experimental Evaluation

4.1 Use Case

One possible use case of the proposed IAM system is travelling between the borders of two countries where driving rules are different. For instance, the minimum age to drive is 17 in Denmark and 18 in Sweden. Thus, we assume a 17-year-old driver with a valid driving license in Denmark. The driver presents the license and ID credentials to the STS in the car and the STS issues a "canDrive" privilege. This privilege is then used by localised UCS+ instances to grant access to the necessary components that allow the driver to drive the car. As soon as the car crosses the borders to Sweden, the location attribute is updated and as a result, the STS re-evaluates relevant policies. In this case, the STS finds that the "canDrive" privilege is not valid anymore, so it revokes it. This demonstrated in the non-inclusing policies of Listing 1.2 The localised UCS+ instances receive the revocation update from the VDR, and revoke access to the protected components in a safely manner. However, policies of localised UCS+ instances may include oligations that can activate autopilot or take safety-related actions upon the revocation of privileges as shown in Listing 1.3. This use case demonstrates that our IAM supports both centralised and distributed control. Centralised control is performed by the STS that monitors the global context and verifies credentials that are relevant to all resources. Distributed control is enforced by

localised UCS+ instances that monitor local contexts and make decisions relevant to the protected resources only. Our IAM allows the car to monitor both global and local contexts and react accordingly. Alternative solutions that only support centralised control cannot react to changes that are relavant to a particular resource. On the other hand, solutions that only support distributed control introduce a performance overhead because global context and attributes would have to be monitored by all access control nodes.

Listing 1.2. STS policies

```
policySet privileges {
    apply allApplicable
    policy driver {
        target clause Attributes.isRegistered
        apply firstApplicable
        rule driver_dk {
            target clause Attributes.license.expiry > time.now()
                        and Attributes.license.issuer == ''borger.dk''
                        and Attributes.location == ''Denmark''
            condition Attributes.age > 17 and Attributes.ucs.step == ''ongoing''
            permit
            on permit {
                obligation canDrive {
                    command = ''issue_privileges''
                    canDrive = true
        }}}
        rule driver_dk_eu {
            target clause Attributes.license.expiry > time.now()
                        and Attributes.license.issuer == ''borger.dk''
            condition Attributes.age > 18 and Attributes.ucs.step == ''ongoing''
            permit
            on permit {
                obligation canDrive {
                    command = ''issue_privileges''
                    canDrive = true
        }}}
        rule revoke {
            condition Attributes.ucs.step == ''post''
            deny
            on deny {
                obligation canDrive {
                    command = ''revoke_privileges''
                    canDrive = false
    }}}};
    policy wiperControl { ... };
    };
```

4.2 Test Cases

We evaluated the prototype on a computer running Ubuntu 20.04LTS Linux OS with 32 GB of RAM and Core i7-9850H CPU. Specifically, we measured the time required to evaluate STS and UCS+ policies as well as the overhead cost of an increasing number of attributes. We also measured the time required to issue and verify VCs as well as the effect of the number of claims on the issuance and

Listing 1.3. STS policies

```
policy engineControl {                                                          1
    target clause Attributes.resourceId == ''engine''                           2
    rule drive {                                                                3
        target clause Attributes.api == ''engineAPI'' and Attributes.action == ''startEngine''4
        condition Attributes.canDrive == true and Attributes.ucs.step == ''ongoing''  5
        permit                                                                  6
    }                                                                           7
    rule revoke {                                                               8
        condition Attributes.ucs.step == ''post''                               9
        deny                                                                    10
        on deny {                                                               11
            obligation safeStop {                                               12
                command = ''autopilot''                                         13
                action = ''safe_stop''                                          14
    }}}};                                                                       15
```

verification time. Finally, we evaluated the performance penalty of enforcing the PoLP especially with an increasing number of privileges. The results are shown in Figure 4 and are discussed in the following subsections.

Performance of Policy Evaluation in UCS+ To evaluate the performance of UCS+, we ran five tests with an increasing number of attributes that need to be collected by PIPs. It necessary to note that this usually depends on the PIPs and how they collect attribute values. For instance, if a PIP needs to retrieve an attribute value over the network, then the network delay would affect the policy evaluation time. However, UCS+ caches attribute values in the AT, so only policy evaluations that need an uncached attribute value would be affected by such overhead. For this reason, we only measure the time required for policy evaluation when the attributes are already cached. We ran each test 1000 rounds and observed a standard deviation of 19.2µs. Table 1 and Figure 4 show the average time required for policy evaluation in each test. The results demonstrate that our implementation is lightweight, very efficient and highly optimised, which make it suitable for embedded and safety-critical systems like smart vehicles.

Performance Evaluation of Issuing and Verifying VCs We ran five different test cases, in which the VC contains 1, 3, 5, 10 and 20 claims. In each case, we ran the test 1000 rounds and computed the average time and the standard deviation. The observed standard deviation was 13.8µs and the results are shown in Table 2. The results show that the number of claims does not have a significant effect on the efficiency of issuing and verifying VCs. In addition, they show that the time required to issue or verify a typical VC is less than 0.5ms. This is because we used Ed25519 digital signatures, which are faster than any EcDSA or EdDSA instances in terms of signature generation and verification.

Performance Evaluation of the PoLP Enforcement The implementation of the PoLP includes back and forth communication between the wallet and

the PEP, and between the PEP and the PIP. Undoubtedly, this introduces a performance overhead especially because the PIP asks for privileges sequentially and synchronously. In order to measure this overhead, we conducted 4 test cases with an increasing number of privileges required for an action. The average time to enforce the PoLP was computed along with the standard deviation after running each test case 1000 times. We noted a standard deviation of 44.7µs. The results presented in Table 3 show that the overhead increases significantly and linearly as the number of required privileges increases. However, the number of privileges required for a specific action is not expected to exceed 3 privileges, so such overhead can be tolerated.

Table 1. Average time to issue and verify a VC as function of the number of claims

Number of attributes	**1**	**3**	**5**	**10**	**20**
Average time to evaluate a policy (in µs)	9	11	18	27	29

Table 2. Average time to issue and verify a VC as function of the number of claims

Number of claims	**1**	**3**	**5**	**10**	**20**
Average time to issue a VC (in µs)	86	97	105	136	196
Average time to verify a VC (in µs)	115	119	125	142	172

Table 3. Average time to enforce the PoLP as function of the number of privileges

Number of privileges	**1**	**3**	**5**	**10**
Average time to enforce the PoLP (in µs)	300	952	1554	3136

5 Conclusion and Future Work

This work is a preliminary effort with several areas for improvement. For instance, in order to regulate the disclosure of credentials and privileges, we plan to investigate the addition of a negotiation protocol between the wallet and other components. We also intend to use JSON-LD, to interpret and serialize linked data in JSON and be able to support signature sets, signature chaining etc. Using selective disclosure scheme and predicate proofs is much safer than disclosing raw information while still maintaining ownership of the claims. Future directions in this work should aim at providing anonymity, pseudonymity, and maximizing privacy of the holder (i.e. wallet information), such that an issuer or verifier

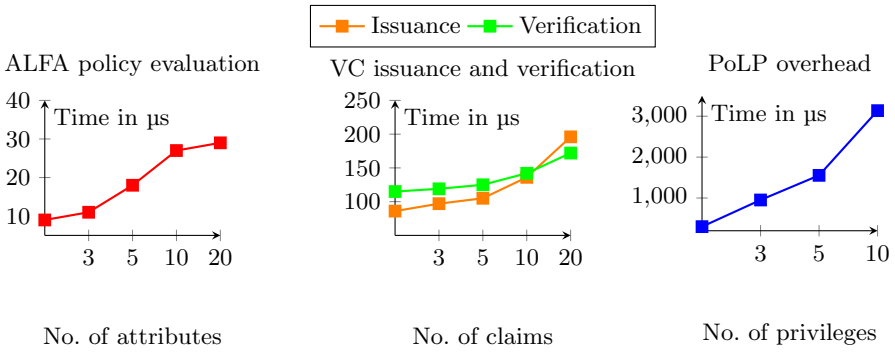

Fig. 4. Performance results of issuing and verifying VCs, and enforcing the PoLP

cannot identify the subject properties, in a way that there is always a collision resistant unique instance set. For future work using zero knowledge schemes, we will look at minimizing data sharing and still be able to create a fully functional ecosystem of multi-party credential exchange and identity verification.

We demonstrated the usage of verifiable credentials through prototype implementation, and how UCON serves as the basis for continuous monitoring through policies. We also provided UCS+, a highly efficient implementation of the UCON model and showed that UCS+ can be used as a dynamic and context aware STS. In addition, we presented a vehicular IAM system that supports context-aware and continuous, and centralised and localised access control based on UCON. We sincerely hope that our work will encourage the extension of privacy-preserving security systems in smart cars and also in other domains where dynamic authorization, continuous monitoring and usage of digital credentials acts as a way of providing safety, security and identity verification while maintaining subject integrity.

References

1. Ammar, M., Janjua, H., Thangarajan, A., Crispo, B., Hughes, D.: Securing the On-board Diagnostics Port (OBD-II) in Vehicles. 8th Embedded Security in Cars (ESCAR USA) (2020)
2. AUTOSAR: Explanation of Adaptive Platform Design (March 2019), https://www.autosar.org/fileadmin/user_upload/standards/adaptive/19-11/AUTOSAR_EXP_PlatformDesign.pdf
3. Bernardini, C., Asghar, M.R., Crispo, B.: Security and privacy in vehicular communications: Challenges and opportunities. Vehicular Communications **10**, 13–28 (2017)
4. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.Y.: High-speed high-security signatures. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 124–142. Springer (2011)
5. Brendel, J., Cremers, C., Jackson, D., Zhao, M.: The provable security of ed25519: theory and practice. IEEE Security & Privacy (2020)

6. Burkacky, O., Deichmann, J., Klein, B., Pototzky, K., Scherf, G.: Cybersecurity in automotive: Mastering the challenge (June 2020), https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/cybersecurity-in-automotive-mastering-the-challenge
7. Chen, L., Moody, D., Regenscheid, A., Randall, K.: Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters. Tech. rep., National Institute of Standards and Technology (2019)
8. Deichmann, J., Klein, B., Scherf, G., Rupert, S.: The race for cybersecurity: Protecting the connected car in the era of new regulation (October 2019), https://mckinsey.com/industries/automotive-and-assembly/our-insights/the-race-for-cybersecurity-protecting-the-connected-car-in-the-era-of-new-regulation
9. Denis, F.: libsodium: A modern and easy-to-use crypto library (2017), https://libsodium.gitbook.io/doc/
10. Dürrwang, J., Braun, J., Rumez, M., Kriesten, R.: Security evaluation of an airbag-ECU by reusing threat modeling artefacts. In: 2017 International Conference on Computational Science and Computational Intelligence (CSCI). pp. 37–43. IEEE (2017)
11. Gupta, M., Sandhu, R.: Authorization framework for secure cloud assisted connected cars and vehicular internet of things. In: Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies. pp. 193–204 (2018)
12. Hamad, M., Prevelakis, V.: Secure APIs for Applications in Microkernel-based Systems. In: ICISSP. pp. 553–558 (2017)
13. Hu, V.C., Ferraiolo, D., Kuhn, R., Friedman, A.R., Lang, A.J., Cogdell, M.M., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K., et al.: Guide to attribute based access control (abac) definition and considerations (draft). NIST special publication **800**(162) (2013)
14. Hugot, V., Jousse, A., Toinard, C., Venelle, B.: omac: Open model for automotive cybersecurity. In: 17th escar Europe: embedded security in cars (Konferenzveröffentlichung) (2019)
15. Josefsson, S., Liusvaara, I.: Rfc8032: Edwards-curve digital signature algorithm (eddsa). Request for Comments, IETF (2017)
16. Kim, D.K., Song, E., Yu, H.: Introducing Attribute-Based Access Control to AUTOSAR. Tech. rep., SAE Technical Paper (2016)
17. Miller, C., Valasek, C.: Remote exploitation of an unaltered passenger vehicle. Black Hat USA **2015**, 91 (2015)
18. OASIS: Abbreviated language for authorization Version 1.0 (2015), https://bit.ly/2UP6Jza
19. OASIS: eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01 (2017), http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.html
20. Park, J., Sandhu, R.: The UCONABC usage control model. ACM Transactions on Information and System Security (TISSEC) **7**(1), 128–174 (2004)
21. Rumez, M., Grimm, D., Kriesten, R., Sax, E.: An Overview of Automotive Service-Oriented Architectures and Implications for Security Countermeasures. IEEE Access **8**, 221852–221870 (2020)
22. Sporny, M., Longley, D., Chadwick, D.: Verifiable credentials data model 1.0. Tech. rep., W3C (November 2019), https://www.w3.org/TR/vc-data-model/
23. Wouters, L., Marin, E., Ashur, T., Gierlichs, B., Preneel, B.: Fast, furious and insecure: Passive keyless entry and start systems in modern supercars. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 66–85 (2019)