Formal Verification of the SCMS proposal for V2X communications by B.Brecht et al.

Firrera Manuel s329226

Prof. Sisto Riccardo
Bussa Simone

June 1, 2025

Abstract

This report presents the work carried out for the Security Verification and Testing course, part of the MSc in Cybersecurity at Politecnico di Torino. The focus of the project was on the formal verification of the Secure Credential Management System (SCMS) for V2X communications using the ProVerif protocol verifier[2]. Building upon previous efforts, we implemented additional sanity checks to validate the correctness of the protocol model. Furthermore, we analyzed the specification to identify potential vulnerabilities and explored various attack scenarios involving entity compromission. The goal was to assess the system's resilience and interpret the attack traces produced by the verifier.

1 Introduction

The objective of this project was to build upon the work carried out by previous students on the formal verification of the Secure Credential Management System (SCMS) for V2X communications[3] using the ProVerif verifier. Our initial focus was on implementing additional sanity checks to ensure the correctness of the protocol implementation. These checks helped confirm that the system met basic security properties under normal operating conditions.

We also explored various attack scenarios, simulating the compromission of specific entities within the system to evaluate its resilience against adversarial behavior. By doing so, we aimed to understand the impact of such compromises and analyze the potential attack traces reported by the ProVerif verifier.

This iterative approach allowed us to deepen our understanding of the protocol's robustness and the effectiveness of the ProVerif tool in modeling complex security protocols for vehicular communications.

2 The Secure Credential Management System

The automotive industry is rapidly advancing toward the deployment of Vehicle-to-Everything (V2X) communication technologies, which enable vehicles to exchange information with other vehicles (V2V),

infrastructure (V2I), pedestrians (V2P), and networks (V2N). These capabilities support a wide range of applications, from infotainment and navigation services to safety-critical functions such as collision avoidance, cooperative adaptive cruise control, and emergency vehicle warnings.

Given the critical nature of many V2X applications, ensuring the security and privacy of the underlying communications is essential. Without strong protections, malicious actors could inject false information, disrupt traffic systems, or compromise the privacy of road users. To address these challenges, the U.S. Department of Transportation and the Crash Avoidance Metrics Partners (CAMP) developed the Security Credential Management System (SCMS), a comprehensive PKI-based solution tailored specifically for V2X environments.

2.1 Overview of the SCMS

The SCMS is a scalable, privacy-preserving Public Key Infrastructure (PKI) designed to secure V2X communications. It ensures that messages are authenticated and tamper-proof, while simultaneously protecting the privacy of the message senders. Unlike traditional PKIs, which are typically designed for smaller scales and static identities, the SCMS is built to issue and manage an enormous volume of short-lived, pseudonymous certificates—on the order of 300 billion certificates annually in the U.S. alone.

2.2 SCMS Architecture and Components

The SCMS is composed of multiple interconnected components and authorities 1, each with a clearly defined role:

- **Root Certificate Authority (RCA)**: The trust anchor of the system, responsible for signing the certificates of subordinate CAs.
- **Intermediate Certificate Authorities (ICA)**: Provide a level of indirection and support scalability by signing end-entity CAs such as the Enrollment CA and the Authorization CA.
- **Enrollment Certificate Authority (ECA)**: Issues long-term enrollment certificates to devices after verifying their legitimacy.
- **Pseudonym Certificate Authority (PCA)**: Issues short-lived pseudonym certificates that are used by devices to sign V2X messages.
- **Location Obscurer Proxy (LOP)** and **Registration Authority (RA)**: Act as intermediaries between devices and CAs to prevent linking of pseudonym certificates to enrollment certificates.
- **Misbehavior Authority (MA)**: Collects evidence of misbehavior and coordinates revocation of malicious or faulty devices.
- **Linkage Authorities (LA1 and LA2)**: Operate independently to support efficient certificate revocation while maintaining user privacy. They generate linkage values that are embedded in pseudonym certificates to enable revocation without exposing identities.

2.3 SCMS Use Cases

The SCMS is designed to support four key operational use cases that are essential for managing trust, security, and privacy in V2X communications. These use cases define how devices interact with the SCMS throughout their lifecycle—from initialization to potential revocation:

1. **Bootstrapping**: This is the initial step where a device (e.g., a vehicle's on-board unit) is enrolled into the SCMS. During bootstrapping, the device securely communicates with the Enrollment Certificate Authority (ECA) to obtain a long-term enrollment certificate. This certificate confirms the device's legitimacy and allows it to request further credentials from the system.

2. **Certificate Provisioning**: Once enrolled, the device periodically obtains batches of short-lived *pseudonym certificates* from the Pseudonym Certificate Authority (PCA). These certificates are used to sign V2X messages during daily operation. Provisioning is handled via the Registration Authority (RA) and the Location Obscurer Proxy (LOP), which anonymize the request to prevent linking between the device's identity and its pseudonym certificates.

3. **Misbehavior Reporting**: Devices within the network monitor for signs of misbehavior—such as false alerts or malicious activity. When misbehavior is detected, evidence is sent to the Misbehavior Authority (MA). The MA analyzes this evidence to determine whether a device is acting maliciously and whether it should be revoked.

4. **Revocation**: If the MA confirms that a device is misbehaving, it initiates the revocation process. This involves working with both Linkage Authorities (LA1 and LA2) to retrieve the linkage seeds associated with the misbehaving device's pseudonym certificates. These linkage values are used to identify and revoke all certificates from that device without revealing its long-term identity. Revoked certificates are published in Certificate Revocation Lists (CRLs), which are distributed to all participants in the system.

2.4 Privacy and Pseudonym Certificates

To protect the privacy of drivers and prevent long-term tracking, the SCMS issues *pseudonym certificates*, which are short-lived certificates that allow vehicles to sign messages without revealing their long-term identity. Vehicles are provisioned with batches of these certificates, which they rotate frequently (e.g., every 5 minutes or based on location change) to make it difficult for observers to correlate messages and track users over time.

Several architectural and procedural safeguards are implemented to preserve unlinkability:

- **Organizational Separation**: Different SCMS components are operated by separate organizations to prevent collusion. For example, the ECA and PCA do not share data directly.
- **Proxy-Based Request Routing**: Vehicles send certificate requests through the RA and LOP, which strip identifying information before forwarding them to the CAs.
- **Dual Linkage Values**: Pseudonym certificates contain linkage values generated by two independent Linkage Authorities (LA1 and LA2). Only when these values are combined—under MA control—can certificates be linked for revocation purposes.

2.5 Revocation and Misbehavior Handling

Efficient and privacy-respecting revocation is a key requirement of SCMS. When a vehicle is determined to be misbehaving (e.g., sending false alerts), the MA initiates a revocation process. Using evidence submitted by other vehicles and infrastructure, the MA identifies the misbehaving device and triggers a coordinated response:

1. The MA requests the linkage seeds from both LA1 and LA2.
2. These seeds allow the computation of all future linkage values for that device.
3. Pseudonym certificates with matching linkage values can then be added to Certificate Revocation Lists (CRLs).

This approach ensures that only the misbehaving vehicle's certificates are revoked, without revealing its past or long-term identity.

3 Modeling in ProVerif & Simplifications

The source codes used for this project are publicly available on GitHub [1]. Given the complexity of the SCMS architecture and its numerous interacting components, it is necessary to introduce certain simplifications in order to effectively model, isolate, and test individual parts of the infrastructure.

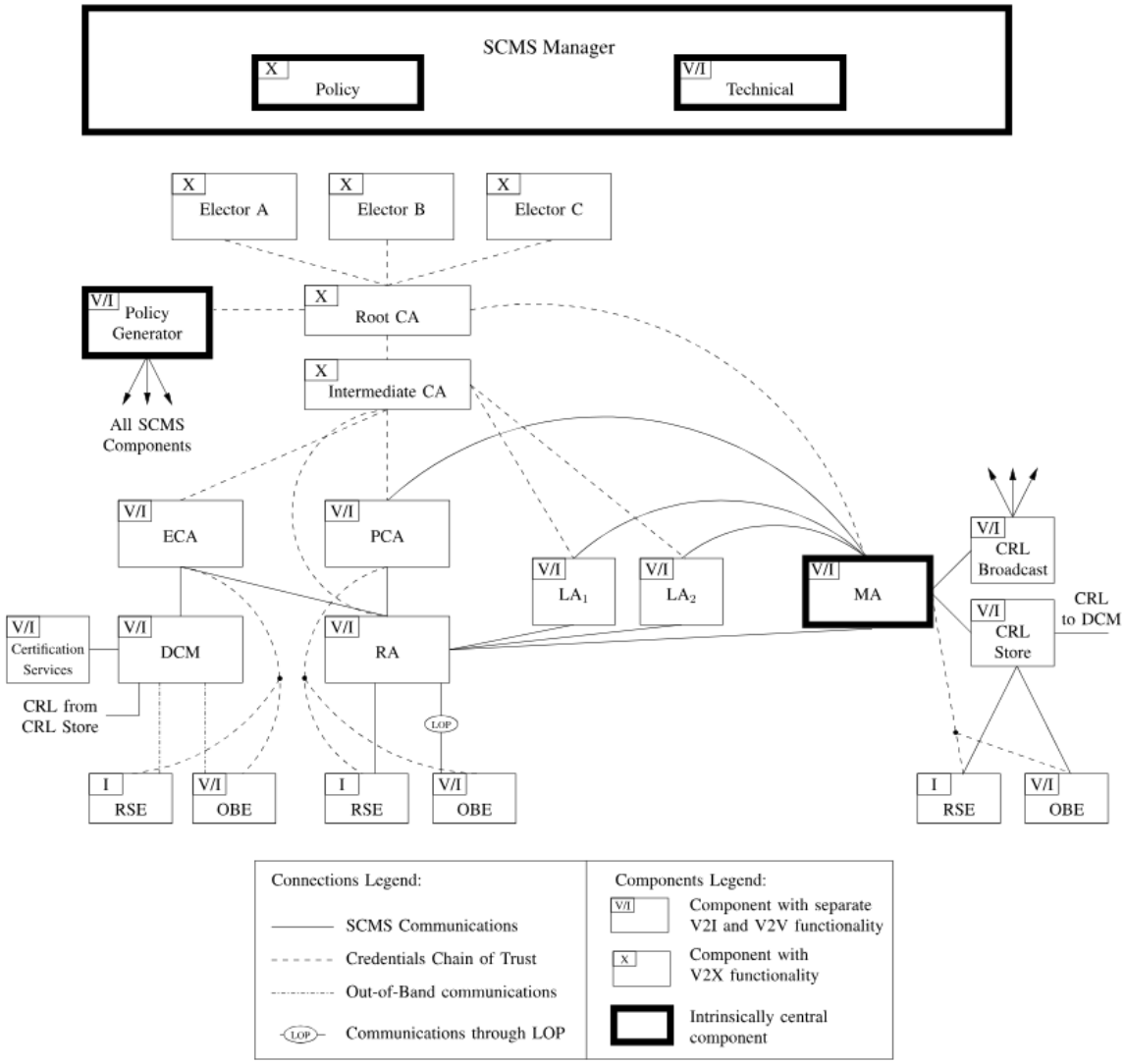

Fig. 1. SCMS architecture overview.

Figure 1: SCMS Architecture overview

Formal verification tools such as ProVerif operate under strict computational and logical constraints, making it impractical to model the entire SCMS system in its full detail.

Additionally, hardware and performance limitations play a significant role in the design choices made for actual implementations, which further motivates the need to focus on specific subsystems when analyzing security properties.

For the purpose of our analysis, we divided the SCMS functionality into two primary domains:

- **Enrollment and Provisioning**: This part covers the initial bootstrapping of devices, the issuance of long-term enrollment certificates, and the periodic provisioning of short-lived pseudonym certificates. It involves components such as the ECA, PCA, RA, and LAs.
- **Misbehavior Reporting and Revocation**: This domain includes the mechanisms for detecting, reporting, and revoking misbehaving entities in the network. It involves interactions between the device, the Misbehavior Authority (MA), the RA, the PCA and the LAs.

By decoupling these two parts, we are able to more precisely verify their individual security properties—such as authentication, confidentiality, unlinkability, and robustness against compromise—without the full complexity of the entire SCMS system overwhelming the analysis process.

3.1 Device Enrollment & Pseudonym Certificate Provisioning

This section focuses on the first two operational use cases of the SCMS: device enrollment and pseudonym certificate provisioning. These steps are fundamental to enabling secure and privacy-preserving V2X communication.

During the enrollment phase, a device—such as an On-Board Equipment (OBE) unit—registers with the SCMS by interacting with the Enrollment Certificate Authority (ECA). The ECA issues an enrollment certificate that uniquely identifies the OBE by embedding its device identifier. This long-term certificate establishes the device's legitimacy and authorizes it to participate in subsequent interactions with the SCMS.

Once enrolled, the OBE can begin the process of pseudonym certificate provisioning. This process is significantly more complex and involves multiple SCMS components, including the Registration Authority (RA), Location Obscurer Proxy (LOP), and Pseudonym Certificate Authority (PCA). These components work together to ensure that:

- The device receives a batch of short-lived pseudonym certificates that can be used to sign Basic Safety Messages (BSMs) and other V2X communications.
- The identity of the device is protected through organizational separation and request anonymization.
- The pseudonym certificates are unlinkable to each other and to the enrollment certificate.

The collaboration between these entities ensures that vehicles can authenticate their messages while preserving user privacy and resisting tracking by external or internal observers.

Verified Security Properties

Using formal verification with the ProVerif tool, we analyzed the simplified model of enrollment and certificate provisioning to verify the following key security and privacy properties:

- **Authentication:** Only valid devices possessing a legitimate enrollment certificate can obtain pseudonym certificates. This property was verified using *correspondence assertions* in ProVerif, ensuring that a certificate issuance event can only occur if a valid enrollment event has taken place beforehand.
- **Confidentiality:** Sensitive data, such as the device identifier, must not be leaked to unauthorized entities during the provisioning process. This was verified using the `attacker(term)` query in ProVerif, to ensure that the attacker cannot derive or observe specific terms such as device IDs or secret values.

- **Unlinkability:** An adversary must not be able to link multiple pseudonym certificates to the same device or to its enrollment certificate. This property was modeled using *observational equivalence* (via the `choice` construct) in ProVerif, by comparing two indistinguishable scenarios where different devices are issued certificates. This was done both on PCA and LA side.

- **Anonymity:** Even if parts of the provisioning infrastructure (e.g., the PCA) are compromised, they must not be able to identify the requesting device. Anonymity was verified using *choice-based equivalence* in ProVerif, ensuring that the identity of the device making a request cannot be distinguished by any observer, including a compromised internal component.

- **Uniqueness:** Time-based queries were employed to verify that each certificate request, given a specific set of input parameters, results in a unique output. This ensures that pseudonym certificate requests cannot be duplicated or replayed without detection, thereby contributing to the system's robustness against certain classes of impersonation or replay attacks.

The results demonstrate that the SCMS provisioning process, when correctly implemented and with strong organizational separation, can uphold both security and privacy goals.

The simplified architecture used for modeling is shown in Figure 2. In this representation, only the fundamental components—namely the Pseudonym Certificate Authority (PCA), Registration Authority (RA), Linkage Authorities (LA), and Enrollment Certificate Authority (ECA)—are included.

This reduction was necessary due to hardware limitations and to maintain tractability within the verification framework. Additionally, some components were excluded either due to a lack of detailed public specifications from the original authors or because they do not directly influence the security properties under investigation in this work.

The dashed lines originating from the Root Certificate Authority (Root CA) represent the trust relationships established through certificate issuance. These lines indicate the chain of trust that authorizes each entity to participate in the SCMS. Although the Root CA is not actively involved in operational workflows, its role as the trust anchor is implicitly assumed in all verification scenarios.

3.2 Misbehavior Reporting & Revocation

In the SCMS, On-Board Equipment (OBE) units are capable of detecting and reporting misbehavior from other devices in the network. When misbehavior is observed—such as the transmission of falsified messages—OBEs can submit reports to the Misbehavior Authority (MA). The MA is responsible for analyzing the reports using misbehavior detection algorithms and determining whether revocation procedures should be initiated.

The revocation process is complex and involves coordination among multiple SCMS components. If misbehavior is confirmed, the MA requests linkage chain identifiers from the Linkage Authorities (LA1 and LA2), which allow identification of all pseudonym certificates belonging to the misbehaving device, without revealing the device's long-term identity. The Enrollment Certificate Authority (ECA) may also be involved if the device's enrollment certificate needs to be revoked. The final outcome is that both the enrollment certificate and all pseudonym certificates issued after the misbehavior event are invalidated, and corresponding entries are added to the Certificate Revocation List (CRL).

Verified Security Properties

- **Component Authentication:** All entities participating in the revocation process (e.g., MA, LA1, LA2, and ECA) must be authenticated to avoid unauthorized issuance of revocation commands. This was verified using *correspondence assertions* and by modeling authentication handshakes between components to ensure the legitimacy of each actor.

- **Enrollment Certificate Revocation:** When misbehavior is confirmed, the device's long-term enrollment certificate must be revoked and rendered unusable for future interactions. This property was modeled using custom `event` constructs in ProVerif to track the lifecycle of enrollment certificates, and verified through *reachability checks* to ensure revocation events occur as expected.

- **Pseudonym Certificate Revocation:** All pseudonym certificates issued to the misbehaving device after the misbehavior event must be identified and revoked. As with enrollment certificate

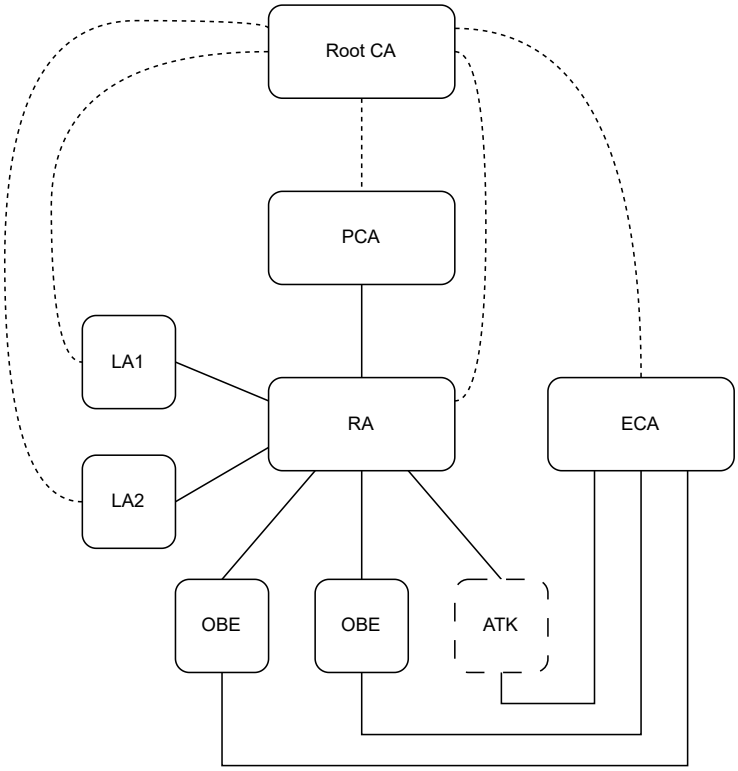

Figure 2: Simplified SCMS Enrollment & Provisioning Architecture overview

revocation, this was verified using ProVerif `events` to trace certificate issuance and validate that only relevant pseudonym certificates are included in the revocation process.

The simplified architecture is shown in Figure 3. Several simplifications were introduced to reduce computational resource usage and to allow a stronger focus on modeling and verifying the revocation process, without excessively abstracting the provisioning phase.

To achieve this, all provisioning-related entities were encapsulated within a single logical macro-component. This approach eliminated the need for public channel communication between internal components, significantly reducing the state space explored by ProVerif. As a result, verification times were drastically improved, allowing the tool to analyze relevant traces much more efficiently.

Despite the abstraction, the core logic of certificate provisioning and revocation remains intact. From the ProVerif queries and the resulting traces, it is possible to observe the full workflow, including the effective revocation of certificates through their insertion into the Certificate Revocation List (CRL). This confirmed the correct functioning of the revocation mechanisms under the defined model and attacker assumptions.

4 Attacking the SCMS

In this section, we analyze the impact of various entity compromises within the SCMS architecture. By simulating attacker capabilities—such as leaking an entity's private key—we assess how the system's security guarantees are affected. Compromise is modeled in ProVerif by explicitly leaking the private key on the public channel:

```
out(pub_ch, entity_pri_key)
```

We compare compromised scenarios against the uncompromised baseline to evaluate the system's resilience.

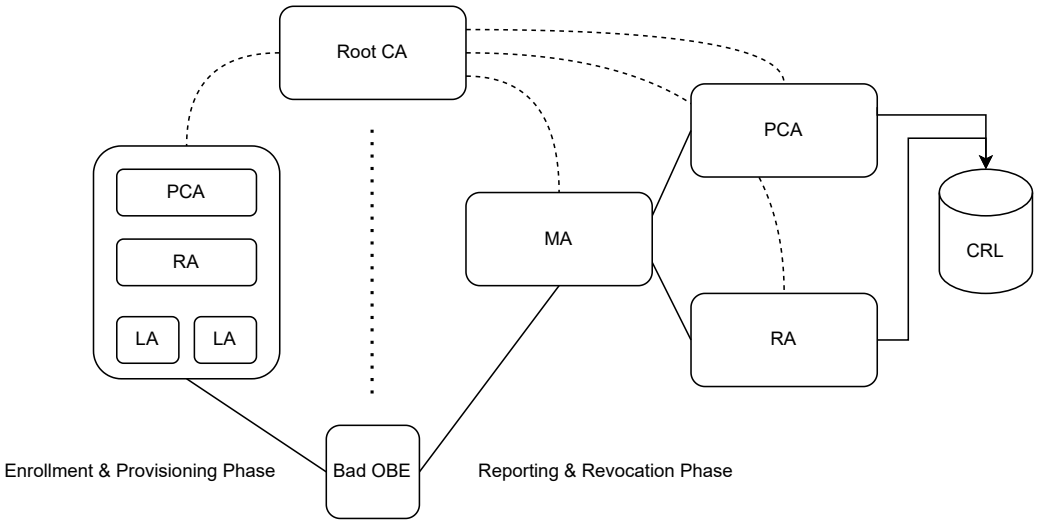

Figure 3: Simplified SCMS Reporting & Revocation Architecture overview

4.1 Provisioning Attacks

Running the model **without any compromise** yields the following results:

1. `not event(Sent_Valid_Enrollment_Cert_ECA(ecert))` is **false**.
2. `not event(Sent_Valid_Attacker_Cert_ECA(aecert))` is **true**.
3. - `not event(Sent_Valid_Message(msg))` is **false**.
   - `not event(Received_Valid_Message(msg))` is **false**.
4. `not event(Message_Discarded_Revoked_Certificate(certid))` is **true**.
5. - `event(received_plv_from_la1_RA(plv))` ⇒ `event(sent_plv_from_LA(plv))` is **true**.
   - `event(received_plv_from_la2_RA(plv))` ⇒ `event(sent_plv_from_LA(plv))` is **true**.
6. `not attacker(scrt[])` is **true**.
7. `not attacker(a_seed[])` is **true**.
8. `event(Received_Valid_Pseudonym_Request_RA(request))` ⇒ `event(Sent_Valid_Pseudonym_Request_OBE(request))` is **true**.
9. `event(Received_Valid_Pseudonym_Request_PCA(request))` ⇒ `event(Sent_Valid_Pseudonym_Request_RA(request))` is **true**.
10. - `event(Received_Valid_Pseudonym_Certificate_OBE(certificate))` ⇒ `event(Sent_Valid_Pseudonym_Certificate_PCA(certificate))` is **true**.
    - `inj-event(Received_Valid_Pseudonym_Certificate_OBE(certificate))` ⇒ `inj-event(Sent_Valid_Pseudonym_Certificate_PCA(certificate))` is **true**.
11. `event(Received_Valid_Message(message_1))` ⇒ `event(Sent_Valid_Message(message_1))` is **true**.
12. `not attacker(pre_linkage_value[])` is **true**.
13. `event(Sent_Valid_Pseudonym_Request_OBE(signed_request_for_ra_1))@t0` ∧ `event(Sent_Valid_Pseudonym_Request_OBE(signed_request_for_ra_1))@t1` ⇒ $t_0 = t_1$ is **true**.

14. `event(Created_Request_OBE(ecert,but_A,but_H,fk_2,fe_2))@t0` ∧ `event(Created_Request_OBE(ecert,but_A,but_H,fk_2,fe_2))@t1` ⇒ $t_0 = t_1$ is **true**.
15. `event(Sent_Valid_Pseudonym_Request_RA(signed_request_for_pca_1))@t0` ∧ `event(Sent_Valid_Pseudonym_Request_RA(signed_request_for_pca_1))@t1` ⇒ $t_0 = t_1$ is **true**.
16. `event(Sent_Valid_Pseudonym_Response_PCA(signed_encrypted_response_for_obe_1))@t0` ∧ `event(Sent_Valid_Pseudonym_Response_PCA(signed_encrypted_response_for_obe_1))@t1` ⇒ $t_0 = t_1$ is **true**.
17. `Choice[obe_enrollment_keymat0, obe_enrollment_keymat1]`: Observational equivalence is **true**.
18. `Choice[la_plv0, la_plv1]`: Observational equivalence is **true**.

Detected Attacks and Explanation

- **Attacker obtains an enrollment certificate:**
  - `not event(Sent_Valid_Attacker_Cert_ECA(aecert))` is **false**: attacker obtains a valid enrollment certificate.
  - `not attacker(scrt[])` is **false**: attacker successfully receives a pseudonym certificate.
  - `event(Received_Valid_Pseudonym_Request_RA(request))` ⇒ `event(Sent_Valid_Pseudonym_Request_OBE(request))` is **false**: attacker sends requests directly to the RA.
  - `event(Received_Valid_Message(message_1))` ⇒ `event(Sent_Valid_Message(message_1))` is **false**: attacker can broadcast seemingly valid messages.

  Explanation: If an attacker manages to acquire a valid enrollment certificate (e.g., by bypassing checks or exploiting weak validation), they can initiate certificate requests and receive valid pseudonym certificates. This allows them to impersonate legitimate ECUs in the system.

- **Attacker obtains enrollment certificate and RootCA is compromised:**
  - `not attacker(scrt[])` is **false**
  - `event(Received_Valid_Pseudonym_Request_RA(request))` ⇒ `event(Sent_Valid_Pseudonym_Request_OBE(request))` is **false**
  - every other authentication check fails

  Explanation: A compromised RootCA means that all subordinate CAs (ECA, PCA, RA) are also implicitly compromised. The attacker can issue their own root-trusted certificates, enabling complete forgery of system credentials and undetectable impersonation.

- **ECA private key compromised:** *Explanation:* With the Enrollment Certificate Authority's key, an attacker can issue arbitrary enrollment certificates. This has the same effect as attacker enrollment: it enables them to participate fully in the provisioning protocol as legitimate entities.

- **RA private key compromised:**
  - `not attacker(scrt[])` is **false**: attacker can obtain a secret response from the PCA.
  - `not attacker(a_seed[])` is **true**: butterfly seeds remain protected.
  - `event(Received_Valid_Pseudonym_Request_PCA(request))` ⇒ `event(Sent_Valid_Pseudonym_Request_RA(request))` is **false**: attacker can spoof RA.
  - `event(Received_Valid_Message(message_1))` ⇒ `event(Sent_Valid_Message(message_1))` is **false**: attacker can inject messages into the network.

  Explanation: If the RA's private key is leaked, the attacker can forge pseudonym requests towards the PCA. Although some secrets (e.g., butterfly seeds) remain safe, the attacker can impersonate the RA to request certificates and relay falsified information.

- **PCA private key compromised:**
  - `inj-event(Received_Valid_Pseudonym_Certificate_OBE(certificate))` ⇒ `inj-event(Sent_Valid_Pseudonym_Certificate_PCA(certificate))` is **false**: attacker can forge certificates.
  - `not attacker(scrt[])` is **true**: legitimate secret OBE-PCA messages are not leaked.

  Explanation: The attacker can now forge valid pseudonym certificates directly. This breaks the integrity of the system, as they can send authenticated messages without having gone through any legitimate provisioning steps.

- **LA private key compromised:**
  - `event(received_plv_from_la1_RA(plv))` ⇒ `event(sent_plv_from_LA(plv))` is **false**: attacker can forge pre-linkage values.

  Explanation: The attacker can generate forged Pre-Linkage Values (PLVs), potentially enabling linkability across pseudonym certificates, or tricking the RA into processing fake PLVs, undermining privacy guarantees.

- **Attacker clones enrolled ECU (`a_seed`, `h_seed`, `fk`, `fe`)):**
  - `not attacker(scrt[])` is **false**
  - `not attacker(a_seed[])` is **false**
  - `event(Received_Valid_Message(message_1))` ⇒ `event(Sent_Valid_Message(message_1))` is **false**

  Explanation: By extracting and cloning the key material of an enrolled ECU, the attacker can duplicate it. The clone is indistinguishable from the legitimate ECU and can independently request valid certificates, send signed messages, and bypass misbehavior detection.

4.2 Misbehavior Handling Attacks

Running the code without any compromission gives the following results:

1. `not event(Sent_Valid_Message(message_1))` is **false**.
2. `not event(Received_Valid_Message(message_1))` is **false**.
3. `not event(Message_Discarded_Revoked_Certificate(cert_id_2))` is **false**.
4. `not event(Enrollment_Discarded_Revoked_Certificate(cert_1))` is **false**.
5. `not event(Add_to_CRL(obe_id_1))` is **false**.
6. `inj-event(Received_Linkage_Value_PCA(request_1))` ⇒ `inj-event(Sent_Linkage_Value_MA(request_1))` is **true**.
7. `inj-event(Received_Hash_Value_MA(response_1))` ⇒ `inj-event(Sent_Hash_Value_PCA(response_1))` is **true**.
8. `inj-event(Received_Hash_Value_RA(request_1))` ⇒ `inj-event(Sent_Hash_Value_MA(request_1))` is **true**.
9. `event(Received_Valid_Message(message_1))` ⇒ `event(Sent_Valid_Message(message_1))` is **true**.

Detected Attacks and Their Explanation

- **MA (Misbehavior Authority) private key compromised:**
  - `inj-event(Received_Linkage_Value_PCA(request_1))` ⇒ `inj-event(Sent_Linkage_Value_MA(request_1))` is **false**: attacker can spoof MA.
  - `inj-event(Received_Hash_Value_RA(request_1))` ⇒ `inj-event(Sent_Hash_Value_MA(request_1))` is **false**: attacker can spoof MA, revoking OBEs.

  Explanation: The MA is responsible for validating misbehavior reports and initiating revocation. If its private key is compromised, the attacker can forge or tamper with linkage values and hash commitments sent to the PCA and RA. This allows arbitrary ECUs to be falsely revoked or real malicious ECUs to remain active. Such attacks directly threaten the system's accountability and can result in denial-of-service for legitimate users.

- **PCA (Pseudonym Certificate Authority) private key compromised:**
  - `inj-event(Received_Hash_Value_MA(response_1))` ⇒ `inj-event(Sent_Hash_Value_PCA(response_1))` is **false**: attacker can spoof the PCA.
  - `inj-event(Received_Hash_Value_RA(request_1))` ⇒ `inj-event(Sent_Hash_Value_MA(request_1))` is **false**: attacker can then get the signed request from MA and send to RA, passing the check on the signature and inserting arbitrary messages into the CRL.

  Explanation: The PCA plays a central role in issuing pseudonym certificates and confirming hash commitments during revocation. If compromised, it can send falsified responses to the MA. This undermines both revocation integrity and pseudonym issuance, allowing an attacker to cover the tracks of malicious ECUs or introduce new ones with fake credentials. It becomes possible to forge misbehavior evidence chains, weakening trust in revocation decisions.

5 Hardware Setup

The ProVerif experiments were conducted using two different machines:

- A commodity laptop equipped with an **AMD Ryzen 5 3550H** processor and **16 GB of RAM**.
- A workstation equipped with an **Intel Core i7-3770** processor and **32 GB of RAM**.

Both machines were running a copy of Ubuntu 22.04 LTS and used the same version of ProVerif to ensure consistent results across platforms.

6 Future Work

There are several avenues for expanding this work.

First, using more powerful hardware would allow the full SCMS model to be executed in a single ProVerif file, potentially uncovering complex multi-stage attacks that cannot be modeled under current resource constraints.

Second, the pseudonym certificate provisioning process can be refined to more closely simulate the original SCMS specification, particularly the issuance of batches of short-lived certificates. This would enable better analysis of pseudonym linking and timing-based attacks.

Third, the current implementation of pre-linkage values can be extended by modeling the sequential generation of linkage seeds and the derivation of pre-linkage values based on the time of certificate creation. This would bring the model closer to the full design proposed in the SCMS specification and allow more realistic revocation scenarios to be tested.

Overall, these improvements would provide deeper insights into the security and privacy guarantees of SCMS under realistic conditions.

7 Conclusions

This work aimed to formally analyze the Secure Credential Management System (SCMS) for V2X communications using the ProVerif tool, building upon prior efforts to validate the correctness and robustness of its core mechanisms. The SCMS represents a critical infrastructure component for securing vehicular networks, and ensuring its resilience to both internal faults and external attacks is essential for the safe deployment of intelligent transportation systems.

Our contributions focused on the formal verification of the enrollment and pseudonym certificate provisioning process, along with the misbehavior detection and revocation components. Through a combination of correspondence assertions, secrecy queries, and observational equivalence tests, we verified important security and privacy properties, including authentication, confidentiality, anonymity, and unlinkability. Each of these was evaluated under both normal and adversarial conditions, including simulations of entity compromise.

Given the complexity of the SCMS architecture, we introduced simplifications to reduce computational overhead without compromising the integrity of the security model. This included abstracting less critical components and encapsulating subsystems within macros to enable more tractable verification. The resulting model allowed us to identify the conditions under which critical assumptions break, particularly when key management authorities such as the RA, PCA, or MA are compromised.

Our findings confirm that the SCMS offers strong guarantees in its nominal operational state. However, several scenarios demonstrated how adversaries with access to compromised private keys can circumvent authentication mechanisms, forge pseudonym certificates, and even inject false reports into the revocation process. Nevertheless, certain privacy features, such as enrollment anonymity and partial unlinkability, appear to be preserved even under non-trivial compromise scenarios.

Overall, this work highlights the importance of formal verification tools in assessing large-scale, real-world security architectures. The SCMS remains a promising framework for securing V2X communications, but its real-world deployment must be accompanied by rigorous implementation and key management practices. Future work should aim to refine the model further, include additional components, and leverage more powerful verification tools to fully capture the dynamic behavior of the system over time.

References

[1] https://github.com/firreram/V2X_SCMS_ProVerif.

[2] Blanchet, B. ProVerif: Cryptographic protocol verifier in the formal model. https://bblanche.gitlabpages.inria.fr/proverif/ (Accessed: 2025-04-10).

[3] Brecht, B., Therriault, D., Weimerskirch, A., Whyte, W., Kumar, V., Hehn, T., and Goudy, R. A security credential management system for v2x communications. *IEEE Transactions on Intelligent Transportation Systems 19*, 12 (2018), 3850–3871.