Establishing End-to-End Secure Channel for IoT Devices through an Untrusted C-ITS Network

Simon Bouget[1], Shahid Raza[1] and Martin Furuhed[2]
[1]*RISE Research Institute of Sweden, Isafordsgatan 22, Kista 16440, Sweden*
[2]*Technology Nexus Secured Business Solutions, Telefonvägen 26, Hägersten 12626, Stockholm, Sweden*

Keywords: IoT, ITS, OSCORE, Network Security, Vehicular Network, Tamarin, Formal Verification.

Abstract: Critical infrastructure is becoming increasingly connected, with tighter integration to the Internet of Things (IoT). Transportation systems in particular are getting smarter with increased cooperation between vehicles and the supporting infrastructure (V2X communications), and with intelligent devices introduced in the ecosystem, either tightly integrated to the vehicle (e.g. ECUs, cameras, ...) or external sensors (e.g. temperature sensor in an attached container, smart traffic light, ...). A number of communication and security protocols are being standardized for this Cooperative Intelligent Transport Systems (C-ITS). However, using the current C-ITS standards, the security of individual devices may terminate at the gateway of a vehicle, and consequently in most existing vehicles, individual systems leak sensitive data across vendors.
In this paper, we propose an end-to-end security architecture between C-ITS devices and back-end servers, in which sensitive data from individual devices can be transmitted without trusting third-parties providing the communication infrastructure (e.g. proxies, vehicle gateways, routers). The proposed solution is a standard-based integrated system that exploits recent IoT security standards and ensures inter-operability between C-ITS protocols and conventional Internet protocols. We perform a formal analysis of our architecture using the Tamarin Prover and show that it guarantees the secrecy and authenticity of the communications under adversarial settings.

1 INTRODUCTION

Today's environment in the automotive sector is very heterogeneous, with various software stacks coexisting and with various medium of communication involved. Further, the current major standard on smart vehicles —the Cooperative Intelligent Transport System (C-ITS) standard (Festag, 2014)— offers no security guarantee by default for communication *between* vehicles (or *ITS Units*), in order to give each Unit more flexibility on how to handle incoming communication and relay messages. Picture for instance the following situation: a smart vehicle has external interfaces to multiple mutually un-trusted services/applications such as the infotainment platform, the OEM servers, the vehicle sensors, etc., and serves as a single gateway to put them in relation with each others. To this purpose, it may need to translate messages between several incompatible protocol stacks, repackage messages before forwarding them to their intended destination, etc., thus breaking security sessions in the process. A recent addition to the C-ITS standard, ISO 21177, proposes an optional mechanism to establish and authenticate a secure session between ITS Units, but it relies on two crucial assumptions: a) it requires (D)TLS 1.3 support; and b) there is a pre-existing trust relationship between devices (either directly or through brokering). Both of these assumptions are problematic. First off, some of the physical devices that interact in a C-ITS network are extremely resource-constrained, with limited battery life and hardware capabilities, resulting in very hard constraints on: processing power, memory, and communications. This makes (D)TLS not properly suited for the context. Second, requiring a pre-existing trust relationship in order to establish a secure session only pushes the issue one step further. Thus, even with ISO 21177, it is still challenging to successfully secure a connection going through a C-ITS vehicular network end-to-end: from the point of view of an application trying to reach a distant back-end server, even if its connection to the truck is secure, it has no clear picture of how its messages will be processed and routed beyond that point. They may be re-

Bouget, S., Raza, S. and Furuhed, M.
Establishing End-to-End Secure Channel for IoT Devices through an Untrusted C-ITS Network.
DOI: 10.5220/0010462305620570
In *Proceedings of the 7th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2021)*, pages 562-570
ISBN: 978-989-758-513-5

layed through other neighboring vehicles, or through a nearby roadside unit, all third-party entities that the application has no reason to trust. The OSCORE protocol (Object Security for Constrained RESTFul Environment) (Selander et al., 2019), recently standardized by the IETF, solves the first point by replacing (D)TLS with an application-layer encryption scheme that is very resource-efficient and compatible with a large number of transport protocols. However, similarly to ISO 21177, the OSCORE standard assumes that the two parties trying to communicate have established a common *security context* beforehand.

The traditional solution to establish this trust relationship/security context would be to use a PKI (Public Key Infrastructure), but this is in itself a very resource-intensive process, relying on asymmetric cryptography and large payload which are expensive to send via radio communication. Consequently, establishing an OSCORE security context between a constrained IoT device and a distant back-end server while going through a C-ITS vehicular network requires to adapt the process to the limitations of this specific ecosystem.

Motivating Example 1: Container with Sensitive Material. Some sensitive material needs to be shipped over a long distance. It is transported in a container with built-in sensors to monitor some critical parameters, e.g. temperature for frozen food, no tampering or excessive humidity for a piece of art, level of radiation for nuclear wastes, etc. The manifest of the container also needs to be securely stored and transmitted, e.g. to customs. This container may cross multiple countries using various transportation methods (boat, rail, road, ...) and may not have access to its own long-range communications and rely on the transporting vehicle to serve as a relay. Being designed with transportation in mind from the start, it is reasonable to assume such a container would implement the latest standards used in the transport industry, such as the ISO C-ITS family.

Motivating Example 2: Passenger Health Sensor. A passenger with a serious health condition is using a wearable device to monitor health status. This device handles private and sensitive data, and may need access to the Internet for instance to store backup data on a distant server with more memory or to alert healthcare professionals if the patient gets critical. This health sensor will have to cooperate with the vehicle to establish a secure communication, but still does not want the intermediaries (vehicle, roadside infrastructure, etc.) to get access to the data, only the final end-point of the communication. In this case, since the health sensor is not directly related to the automotive sector, there is no reason to assume it implements any of the relevant standards.

Contribution: In this paper, we present an architecture and workflow that complements C-ITS security standards (ISO 21177) and enables full end-to-end security between a back-end server and an IoT device embedded in a smart vehicle, whether this device is compliant with the current C-ITS standard or not, such as in the two use-cases above. Our solution is fully standard-based, notably leveraging the recent IETF standard OSCORE. We also performed a formal security analysis of our solution and proved that it achieves OSCORE prerequisites.

In the rest of this paper, we give relevant background on the state of the art for automotive networks security and the various technologies we use (Section 2), we present the overall architecture of the proposal (Section 3) and discuss the security guarantees it provides (Section 4), before concluding (Section 5).

2 STATE-OF-THE-ART AND BACKGROUND

Security in automotive networks and transport infrastructures is a very active area. The stakes are immense, with the value of the global intelligent transport system market estimated at more than $20 billions in 2018 and expected to keep growing in future years[1], with national and international institutions pushing for more concerted efforts and standardization (Shan et al., 2019). Moreover, automotive networks require novel solutions to solve their specific challenges, such as: very heterogeneous networks, with a lot of stakeholders and competing vendors that do not trust each others; difficulties to coordinate due to communication nodes (vehicles) quickly moving geographically; need to update over the air (Vasenev et al., 2019); very large attack surface (Stabili et al., 2018) and difficulties to use asymmetric cryptography due to real-time constraints (Laštinec, 2017). On top of that, integrating constrained devices in an automotive network adds another layer of compatibility issues, since both domains developed separate technologies to solve their specific issues, without regards for the other one.

2.1 Automotive Networks

Automotive networks are complex entities that gather very different devices with varying capabilities and

[1]Intelligent Transport System Market – Forecast (2020 - 2025), IndustryArc, 2018, https://www.industryarc.com/Report/15024/intelligent-transport-system-market.html

constraints. Vehicles are the basic units, and can communicate either P2P in a mesh network composed of other vehicles, or with a road-side infrastructure which has a wired connection and can relay the communication to its final destination. The choice of which type to use generally depends on the application. For instance, safety information needs to be exchanged with close vehicles in real time, while traffic statistics can be sent to a central for aggregation and analysis at a more relaxed rhythm.

To ensure that all these units can cooperate, various standardization organizations (ISO, ETSI, etc.) have developed the C-ITS standards (Festag, 2014), for Collaborative Intelligent Transport Systems, itself based on the ITS standard (Nowacki et al., 2012). While ITS focuses on the intelligence at the level of a single unit (a vehicle or a roadside unit), C-ITS extends it to the communications between these systems to provide additional benefits, such as advanced warning in case of accident or reduced traffic congestion.

C-ITS is built around the functional concept of ITS Station (ITS-S), physically implemented in a Station Unit (SU), itself composed of one or various Station Communication Unit (SCU), each providing a specific function. According to ISO 21217, a SCU is organized in several modules, separated in 4 layers — which can be loosely tied to the standard Communication layers defined by the IETF (RFC 1122) used on traditional Internet: Access (IETF Link layer), Network & Transport (IETF IP and Transport layers), Facilities (no direct equivalent), and Application (IETF Application layer)— and 2 cross-layers categories, Security and Management, as illustrated by the various Stations in Figure 2. The two cross-layers modules are further defined in ISO 21177 (Security) and ISO 24102 (Management). C-ITS Stations Units and SCUs are further classified based on the type of network connections they possess: internal to a single SU, to other C-ITS SU, or to an external (non C-ITS) network. The last type in particular is designed as a Station Gateway, and is in charge of the translation between the C-ITS stack and other external protocol stacks (such as OSI-based web stacks).

A **notable limitation of the C-ITS model** is that security properties are by default only defined for point-to-point communications between two SCUs, but each SCU is free to repackage incoming communications before relaying them. Furthermore, the optional mechanisms defined in ISO 21177 rely on (D)TLS 1.3 and are tailored for communications between two SCUs inside the same SU, or between the ITS-SU and the proprietary network embedded in the same smart vehicle, but not for transient IoT devices which would connect at the edges of a C-ITS network. This gives much needed flexibility to the complex automotive ecosystem, but makes it hard to provide any End-to-End security guarantees to IoT devices, and further in this paper, **we propose Application-layer encryption as a solution** (Section 3).

2.2 IoT Devices

A lot of the devices composing the Internet-of-Things possess limited capabilities due to physical constraints, be it bandwidth, communication range, battery life, etc. To cater to those specific needs, new protocols have been designed, tailored for machine-to-machine communication and better efficiency. In particular, two networks stacks have been standardized by the IETF (Internet Engineering Task Force), that we describe here.

2.2.1 CoAP

CoAP (Constrained Application Protocol) (Shelby et al., 2014) is a subset of HTTP designed specifically for machine-to-machine communication and the Internet of Things, to be cheaper and more efficient. Security, if enabled at all, is established in the transport layer, generally using DTLS (Rescorla and Modadugu, 2012) over UDP (Postel, 1980), and a secure CoAP session ran on top of it. This is cheaper than the classic HTTPS stack using TLS over TCP, but still too expensive, and more importantly it has issues with heterogeneous networks, and security is usually broken at proxies and relays.

2.2.2 OSCORE

Alternatively, the OSCORE (Selander et al., 2019) stack establishes security in the application layer, using CBOR (Concise Binary Object Representation) (Bormann and Hoffman, 2013) and COSE (CBOR Object Signing and Encryption) (Schaad, 2017) on top of raw CoAP. As this happen in the uppermost layer, data can go through the translation process between proxies and relays using different stacks while still remaining secure. Moreover, OSCORE uses symmetric encryption, which is much less resource intensive for the constrained devices comprising the Internet of Things. In Table 1, we summarize the correspondence between the C-ITS model and the protocol stacks developed for IoT devices.

Table 1: Comparison between standard web and IoT-dedicated protocol stacks — IoT use specific protocols, tailored for machine-to-machine communications and efficient use of resources, but there is a loose correspondence between the various systems.

Basic Web stack:	IPv4/v6 → TCP → HTTP
Basic IoT stack:	6LowPAN → IPv6 → UDP → CoAP
Secure Web stack:	IPv4/v6 → TCP → TLS → HTTPS → EST
Secure IoT stack (CoAP-S):	6LowPAN → IPv6 → UDP → DTLS → CoAP-S → EST-CoAPS
Secure IoT stack (OSCORE):	6LowPAN → IPv6 → UDP → CoAP → OSCORE → EST-OSCORE
ITS-Station design:	Access → Network. & Transp. → Facilities

3 OSCORE USE IN C-ITS NETWORKS

Securing end-to-end communications for a resource-constrained IoT device participating in a vehicular network cannot be done at the Network & Transport layer because each C-ITS Unit relaying a message might break the (D)TLS session. In this Section, we describe how to use OSCORE as an Application-layer encryption mechanism instead to provide End-to-End security in automotive networks while being compatible with C-ITS standards.

3.1 End-device as a C-ITS Station

In our first motivating example in the Introduction, a container with sensitive material would obviously come in contact with a vehicular network while it is being transported, so it would make sense for the manufacturer to build its communication capabilities as a C-ITS Station Unit. More generally, in cases when the end-device is assumed to participate in a vehicular network (container, traffic monitor, etc.), it should be made compliant with the C-ITS standards.

To secure its communications, such a device has to include an OSCORE module in its Facilities layer that the Applications can call on to encrypt their content, and a module to manage relevant Certificates and Security Contexts in its Security section, as shown in Figure 2 in the top-right. The encrypted OSCORE payload is then sent out on the C-ITS network just as any other message, and only the final destination of the message, which knows the corresponding *Security Context* is able to decrypt the payload and access its content. We give more details on establishing the Security Context further in Section 3.3.

3.2 Connection through a Gateway

In our second motivating example, a health sensor worn by a patient with a chronic condition that needs to be continuously monitored, the sensor is not necessarily used in a vehicle, and most of the time will connect through the hospital or patient's home WiFi, so its manufacturer will not have strong incentives to make the sensor compliant with the C-ITS standard. However, the sensor may want to rely on a vehicular network advanced connectivity while the patient is being transported. More generally, when a device is not targeted for use in vehicular networks and does not implement the C-ITS standard, but can sometimes come in contact with one, it needs an entry point into the C-ITS network to benefit from its capabilities.

In that case, the device has to rely on the C-ITS *Station Gateway* of the host vehicle. The Gateway has to implement the translation process from the OSCORE stack used by the end-device to whichever protocol stack it uses to communicate with the rest of the C-ITS network, as illustrated in Figure 1. After this, the Gateway forwards the re-packaged encrypted OSCORE payload to the rest of the C-ITS network and once again it is transmitted as any other message to its final destination, which is the only one able to decrypt it with the corresponding Security Context. As in the first case though, initially establishing the Security Context is still the responsibility of the end-device, not of the Gateway.

3.3 Establishing a Security Context for OSCORE

In both cases described previously, we mentioned that encrypting and decrypting the OSCORE payload relies on the two end-points sharing an OSCORE *Security Context*. It is needed since OSCORE uses symmetric encryption, which is less resource intensive than asymmetric encryption and better suited to constrained IoT devices, but requires establishing a shared secret between the two ends of the communication. The OSCORE specification assumes that the Security Context has already been established between the two end-points of a communication. The IETF working group LAKE (Lightweight Authenticated Key Exchange) is currently at work on standardizing this process, and has an active draft for a protocol named EDHOC (Ephemeral Diffie-Hellman Over

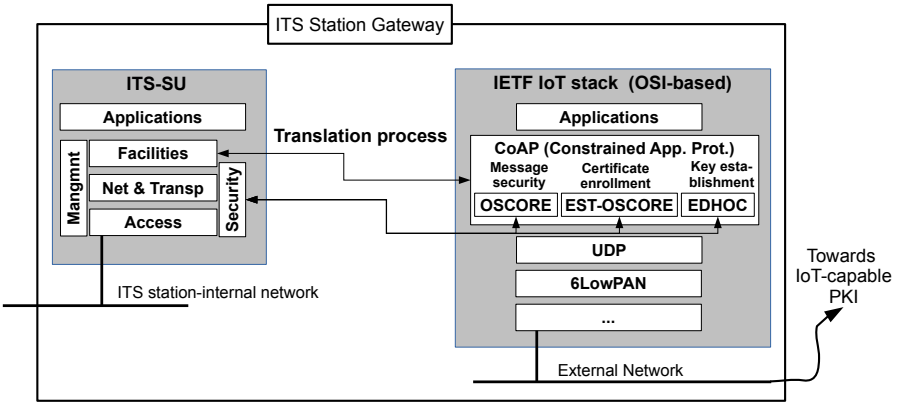

Figure 1: Translation needed to bridge the ITS and OSCORE worlds.

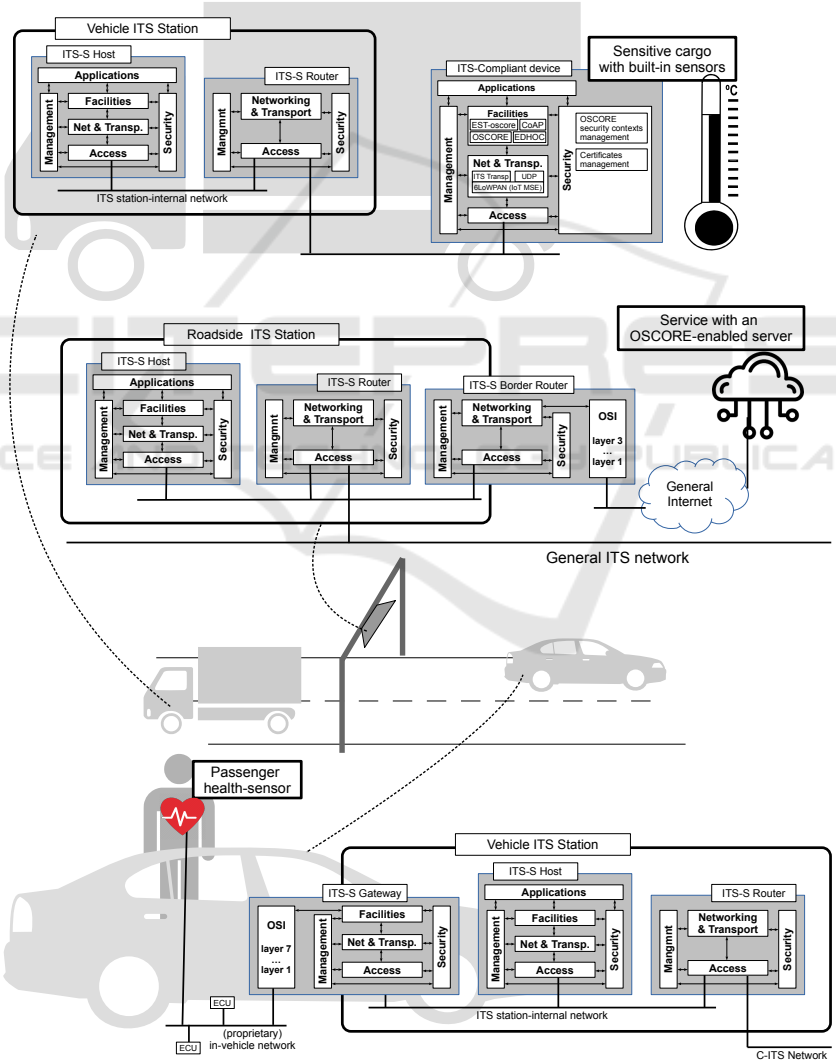

Figure 2: Example of a C-ITS automotive network integrating OSCORE-enabled devices — Two possibilities: a) the device is ITS compliant, and can seamlessly integrate the C-ITS network (truck cargo with built-in sensors); or b) the device is not ITS compliant and has to connect through a Gateway which can handle the translation process for it (health sensor worn by a passenger in the car).

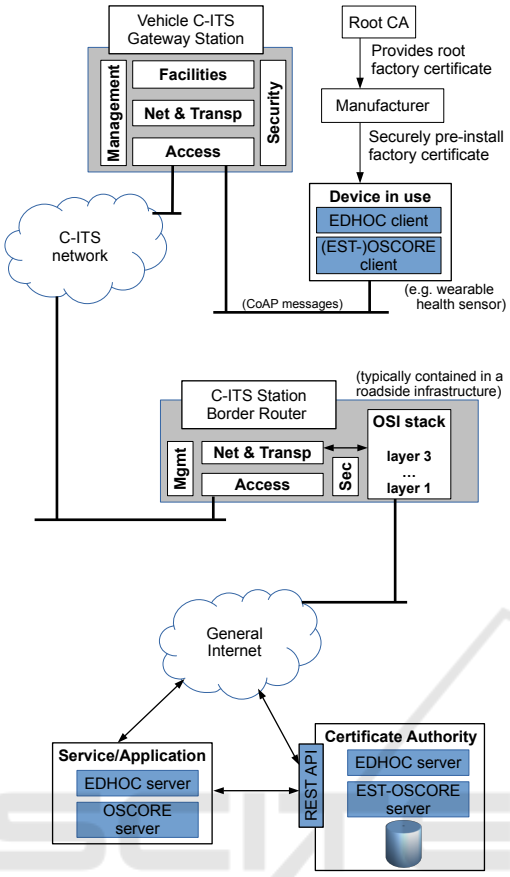

Figure 3: Architecture of an ITS/OSCORE system – A device is provisioned with a factory certificate by its manufacturer. It connects to a C-ITS Network via a Gateway, which relays the CoAP messages containing EDHOC/EST/OSCORE payloads up to a Border Router connected to the general Internet, where the messages can reach their final destination, either a Certificate Authority or a Service.

COSE (Selander et al., 2020a)). EDHOC is a Key Exchange protocol, i.e. a method to securely establish a shared secret known only by the two participants, that is targeted towards IoT devices and optimized for small message sizes, in order to save radio communication and power consumption. It also has the advantage to be based on COSE, just like CoAP/OSCORE, thus being able to share libraries with the other protocols and to reduce the memory footprint of the stack. Other Key Exchange protocols may be used if better suited to a specific context, such as cTLS (compact TLS, with an active draft in the TLS working group) (Rescorla et al., 2020) in protocol stacks where TLS is already used for another reason. But in general, using EDHOC is the simplest way to establish an OSCORE Security Context, and we will focus on this case in the rest of this paper. The typical life-cycle of a device, as illustrated in Figure 4, would be the following: (i) Initially, a manufacturer pre-installs a "factory" or "bootstrap" certificate that will uniquely identify a device for the entirety of its existence. (ii) When used for the first time, a device will use its factory certificate to authenticate at a Certificate Authority (CA) of its choice and enrolls for a new "working" certificate. This CA should run an EDHOC server, which the device will contact to derive symmetric keys and establish an OSCORE context. This context is then used to run a certificate enrollment procedure thanks to EST-OSCORE (Selander et al., 2020b), a method using the payloads defined in the EST standard (Enrollment over Secure Transport (Pritikin et al., 2013)) and protecting them with OSCORE. (iii) With the newly enrolled certificates, a device can then contact and authenticate to any service they want and run EDHOC to generate an OSCORE context, and then communicate securely with the service using OSCORE. Different CAs can also be used to issue further certificates, either re-using the factory certificate again, or using one of the certificates enrolled through the first CA and executing the same EDHOC + EST-OSCORE procedure.

In Figure 3, we summarize the relationships between the various actors involved in that life-cycle: the Manufacturer is responsible for securely installing on each Device a factory certificate provided by the Root Certificate Authority. The Device ultimately wants to establish a secure channel to a given Service or Application, and can do so through the connectivity provided by the C-ITS Network. The Device typically connects through a Gateway Station and sends CoAP messages (carrying EDHOC/OSCORE/EST payloads) that are relayed by the C-ITS Networks until they reach a Border Router, which can forward the messages to the general Internet. The C-ITS network serves as a bridge between the Device, the Service, and the Certificate Authority that allows them to authenticate each-others.

4 SECURITY ANALYSIS

We used the Tamarin prover (Meier et al., 2013) to analyze our proposed architecture. Tamarin has been widely used to analyze protocols' security features, such as TLS (Cremers et al., 2017), 5G (Basin et al., 2018a), or voting protocols (Basin et al., 2018b).

4.1 Introduction to Tamarin

Tamarin is a symbolic analysis tool using multiset *rewriting rules* —to encode a protocol specification and the adversary's capabilities— and *first-order*

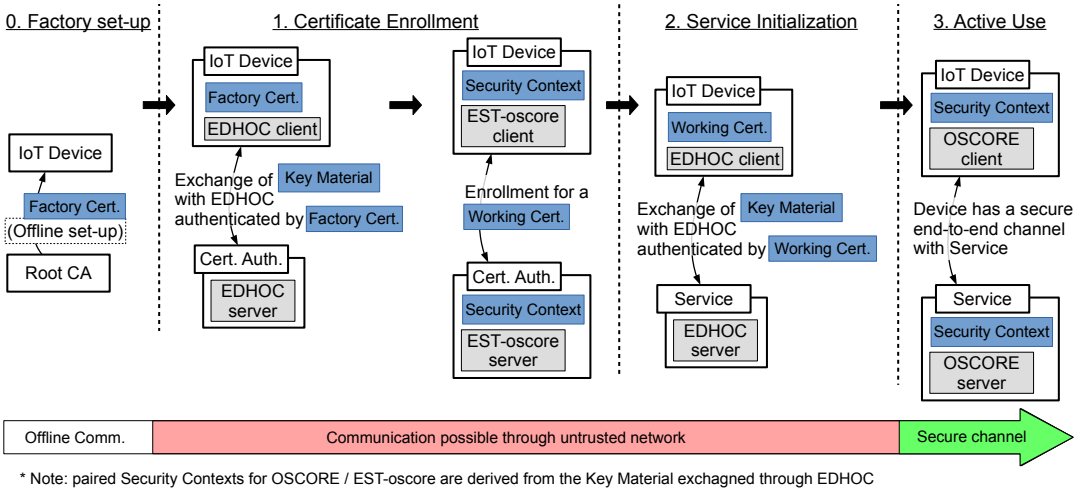

Figure 4: Typical life-cycle of an IoT device in our use-case — 1) A device contact a CA, using its pre-installed factory certificate as authentication, and exchange key material with EDHOC; it derives an OSCORE security context and uses it to enroll for a new working certificate with EST-oscore; 2) the previous procedure is repeated, now using the working certificate to authenticate to a desired service instead; 3) in the end, the device and the service are connected with a secure and authenticated channel protected by OSCORE.

logic formulas —to define security properties. These rewriting rules induce a transition system describing the potential executions of (unbounded numbers of) protocol instances in parallel, allowing us to cover the potential interactions of enrolling at many CAs and using various services concurrently. Tamarin's default adversary model corresponds to a Dolev-Yao(Dolev and Yao, 1983) adversary who has complete control over the network. Users can also extend this default model, either to give more capabilities to the adversary or limit what it can do on the network.

Tamarin also includes an automatic solver that can analyze this transition system and generate formal proofs of the properties This automatic solver relies on a formal semantics and is guaranteed to be correct but may not always terminate. In that case, the protocol specifications and/or the properties to prove need to be modified until the tool is able to reach termination. Alternatively, the user can manually guide the tool instead of relying on the automatic heuristics. When the solver reaches termination, it generates either a proof that a property is valid or a counter-example that exhibits why it is invalid. Counter-examples are presented in a graphical interface and can be used to guide corrections or further refinements of the model. Proofs, however, are not supposed to be human-readable and consist of hundreds of lines of formal logic statements and exhaustive enumeration of all possible cases. Consequently, with correctness being guaranteed by the tool itself added to the lack of readability, discussing the proofs has little value, and the difficulty in such formal analyses is to ensure that the model is a reasonable approximation of the reality and that the security properties defined actually correspond to the real world attacks a protocol aims to protect against. This is what we will do in the rest of this section.

4.2 Model and Specification

Our full specification is available at https://github.com/Simon-Bouget/tamarin-end2end-security/blob/main/general.spthy and can be checked with Tamarin auto-prover using the default heuristic in a few seconds on a common laptop. We modeled three different **kind** of actors: (a) Devices, with a pre-installed factory certificate, modeled as a private/public key pair. (b) Certificate authorities (CA), which can deliver new working certificates, modeled as a private/public key pair based on a fresh nonce and signed by the CA. (c) Services that a device may want to use, modeled as entities with an easily verifiable public key and controlling the corresponding private key. Additionally, we do not assume any trust in the C-ITS network infrastructure, which has been modeled as an attacker with full control over network communications, which is equivalent to the attacker being in control of any relaying node in the C-ITS infrastructure. However, we assume that the adversary cannot physically tamper with the end-devices and access the long-term keys stored there, nor can it corrupt the Certificate Authorities. We modeled the workflow to establish a secure OSCORE session starting with only a factory

certificate, described in the previous Section 3.3. We do not show the code here, but we remind the readers that it is publicly available online.

4.3 Security Properties

In Tamarin, security properties are defined as 1st-order logic formulas, which we use to define the properties that must be true in order to guarantee "end-to-end security" for the communications protected by OSCORE: (i) The OSCORE master secret actually remains secret (from the point of view of the adversary) during the certificate enrollment process and the establishment of the Security Context. (ii) The messages between services and end-devices are secret, i.e. no one except the destination of a message can decrypt its payload. (iii) The messages between services and end-devices are authenticated, i.e. both the service and the device are certain they talk to each other, and replay-protected, i.e. an adversary cannot record an encrypted message and send it a second time to trick its recipient. Altogether, it ensures that a device can establish a secure end-to-end channel with a service and that they can communicate safely even through an untrusted C-ITS network. No man-in-the-middle can spy on their communication or impersonate them, and no replay attack is possible.

5 CONCLUSION

In this paper, we present a solution, compatible with the C-ITS automotive standards and entirely based on standardized (or soon-to-be standardized) open-source protocols, to get end-to-end security between a resource-constrained IoT device and a distant back-end server, including through an un-trusted vehicular network. It leverages the newly standardized OSCORE protocol as an application-layer encryption mechanism to secure a payload even through relays and proxy forwarding, and it covers the whole lifecycle of a device, from the original Certificate Enrollment to the communication between service and end-device through the initial OSCORE Security Context establishment. Further, we analyzed our solution using the Tamarin Prover, a well-known and reputable security analysis tool, and guarantee that our specifications provide authenticity and secrecy even against an adversary in full control of the network.

ACKNOWLEDGMENT

This research has partly been funded by the H2020 ECSEL SECREDAS (Grant ID: 783119) and partly by the H2020 CONCORDIA (Grant ID: 830927).

REFERENCES

Basin, D., Dreier, J., Hirschi, L., Radomirovic, S., Sasse, R., and Stettler, V. (2018a). A formal analysis of 5g authentication. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1383–1396. ACM.

Basin, D., Radomirovic, S., and Schmid, L. (2018b). Alethea: A provably secure random sample voting protocol. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 283–297. IEEE.

Bormann, C. and Hoffman, P. (October 2013). *Concise Binary Object Representation (CBOR)*. RFC 7049, Internet Engineering Task Force (IETF).

Cremers, C., Horvat, M., Hoyland, J., Scott, S., and van der Merwe, T. (2017). A comprehensive symbolic analysis of tls 1.3. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1773–1788. ACM.

Dolev, D. and Yao, A. (1983). On the security of public key protocols. *IEEE Transactions on information theory*, 29(2):198–208.

Festag, A. (2014). Cooperative intelligent transport systems standards in europe. *IEEE Communications Magazine*, 52(12):166–172.

Laštinec, J. (2017). Security extension of automotive communication protocols using ethernet/ip. *Information Sciences and Technologies Bulletin of the ACM Slovakia*, page 49.

Meier, S., Schmidt, B., Cremers, C., and Basin, D. (2013). The tamarin prover for the symbolic analysis of security protocols. In *International Conference on Computer Aided Verification (CAV'13)*, pages 696–701. Springer.

Nowacki, G., Krysiuk, C., Kopczewski, R., and Paszukow, B. (2012). Development and standardization of intelligent transport systems. *International Journal on Marine Navigation and Safety of Sea Transportation*, 6(3).

Postel, J. (August 1980). *User Datagram Protocol*. RFC 768, Internet Engineering Task Force (IETF).

Pritikin, M., Yee, P., and Harkins, D. (October 2013). *Enrollment over Secure Transport*. RFC 7030, Internet Engineering Task Force (IETF).

Rescorla, E., Barnes, R., and Tschofenig, H. (October 2020). *Compact TLS 1.3*. Active Internet-Draft, Internet Engineering Task Force (IETF).

Rescorla, E. and Modadugu, N. (January 2012). *Datagram Transport Layer Security Version 1.2*. RFC 6347, Internet Engineering Task Force (IETF).

Schaad, J. (July 2017). *CBOR Object Signing and Encryption (COSE)*. RFC 8152, Internet Engineering Task Force (IETF).

Selander, G., Mattsson, J., and Palombini, F. (August 2020a). *Ephemeral Diffie-Hellman Over COSE (EDHOC)*. Active Internet-Draft, Internet Engineering Task Force (IETF).

Selander, G., Mattsson, J., Palombini, F., and Seitz, L. (July 2019). *Object Security for Constrained Restful Environments (OSCORE)*. RFC 8613, Internet Engineering Task Force (IETF).

Selander, G., Raza, S., Furuhed, M., Vucinic, M., and Claeys, T. (March 2020b). *Protecting EST Payloads with OSCORE*. Active Internet-Draft, Internet Engineering Task Force (IETF).

Shan, L., Sangchoolie, B., Folkesson, P., Vinter, J., Schoitsch, E., and Loiseaux, C. (2019). A survey on the applicability of safety, security and privacy standards in developing dependable systems. In *International Conference on Computer Safety, Reliability, and Security*, pages 74–86. Springer.

Shelby, Z., Hartke, K., and Bormann, C. (June 2014). *The Constrained Application Protocol (CoAP)*. RFC 7252, Internet Engineering Task Force (IETF).

Stabili, D., Ferretti, L., and Marchetti, M. (2018). Analyses of secure automotive communication protocols and their impact on vehicles life-cycle. In *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 452–457.

Vasenev, A., Stahl, F., Hamazaryan, H., Ma, Z., Shan, L., Kemmerich, J., and Loiseaux, C. (2019). Practical security and privacy threat analysis in the automotive domain: Long term support scenario for over-the-air updates. In *VEHITS*, pages 550–555.