Using a Dual-Layer Specification to Offer Selective Interoperability for Uptane

Marina Moore[1], Ira McDonald[2], Andre Weimerskirch[3], Sebastien Awwad[1], Lois Anne DeLong[1], and Justin Cappos[1]

[1] New York University
{marinamoore,gsa215,lad278,justincappos}@nyu.edu
[2] High North Inc.
blueroofmusic@gmail.com
[3] Lear Corporation
AWeimerskirch@lear.com

Abstract. This work introduces the concept of a dual-layer specification structure for standards that separates interoperability functions, such as backwards compatibility, localization, and deployment, from those essential to reliability, security, and functionality. The latter group of features, which constitute the actual standard, make up the baseline layer for instructions, while all the elements required for interoperability are specified in a second layer, known as a Protocols, Operations, Usage, and Formats (POUF) document. We applied this technique in the development of a standard for Uptane [37], a security framework for over-the-air software updates used in many automobiles. This standard is a good candidate for a dual-layer specification because it requires communication between entities, but does not require a specific format for this communication. By deferring wire protocols and other implementation details to POUFs, the creators of the Uptane Standard were able to focus on the basic procedures and operations needed to secure automotive updates. We demonstrate the effectiveness of this format by specifying a POUF for the Uptane Reference Implementation [36].

Keywords: standardization · interoperability · security · ota · updates · ota updates .

1 Introduction

Standardization represents an important step in the growth of a product or technology. It implies that a sufficient level of adoption has occurred to warrant issuing sanctioned guidelines for the product's safe implementation and use. In turn, having accepted procedures for consistent implementation promotes trust and may encourage adoption [38]. But, for all of its positive aspects, standardization requires the individuals or organizations behind the project to make some difficult choices. A standard, by definition [24] requires adherence to specified steps, procedures, and actions. Yet, in real world applications, it is not always possible to mandate how actions are to be implemented due to factors such as the physical nature of existing systems, proprietary management policies, or differing international requirements [21].

The flip side of this problem is that without mandating specific procedures, it may be impossible to guarantee interoperability of programs, either within a system or across a given group of products. In addition, manufacturers may find their ability to upgrade technologies limited, as new components may not work on legacy systems with narrowly specified data formats. In the automotive industry, where manufacturers often source parts from many suppliers, a lack of interoperability can also mean either reduced selection or costly re-tooling to accommodate differences in system specifications. When it comes to manufacturing vehicles, interoperability is not a trivial goal. Yet, the ability to affirm vehicles were built using guidelines ensuring safe and secure implementations is equally pressing, given the scrutiny auto manufacturing is under as the third most regulated industry in the U.S. [22].

In the past, these conflicting needs have been addressed simply by writing auto industry standards with different levels of detail. While some standards, such as ISO 26262 and ISO/SAE DIS 34 [15, 16] specify framework requirements without describing the implementation details, others such as IEEE 802.11 [6] specify interoperability by defining communication protocols and other details about how communication should function.

The limitations of these "either/or" solutions came to light in our efforts to standardize the Uptane framework. Uptane, officially known as Joint Development Foundation Projects, LLC, Uptane Series, gives automotive original equipment manufacturers (OEMs) and third-party suppliers the tools to secure over-the-air (OTA) software updates on components in connected vehicles [20]. It released its first standard document under IEEE-ISTO in 2019 [35], and is now hosted by the Linux Foundation. During this process, the standardization team identified numerous examples in the existing guidelines where stipulating how a process must be carried out would be problematic for wide adoption of the standard. However, in practice each implementation needs to use specific protocols in order to communicate with other implementations.

This paper introduces a new approach to standardization that eliminates the need to choose security and interoperability over flexibility. It proposes a dual-layer specification in which the first layer is a traditional standard that defines a framework and the optional second layer specifies fine-grained implementation details needed for the tool to work in a given context. This second layer can include the wire protocols, or how data is represented and encoded to be transmitted between devices. Our proposed specification labels this second layer a POUF, an acronym for Protocols, Operations, Usage, and Formats, or the specific additional requirements needed to support interoperability.

As an example of how such a specification could work, consider the way the order of day, month, and year can vary in how dates are written in different countries. All these representations have the same goal: to specify the current date. So the first layer of a dual-layer specification could stipulate *when* a procedure must be carried out without *specifying the order of these numbers.* The

POUF layer would then describe how that information would be communicated and displayed in the given implementation. In the United States, it would specify month, day, year, while a POUF used by most European countries would specify day, month, year. Each POUF contains the information required by the standard, but tailors the time expression to the location where it is used.

To see how our dual-layer specification model might work in a complete system, we used it to write a POUF for an implementation of Uptane based on the current Uptane Standard (V 1.0.0). This POUF layer includes a description of the custom features of the implementation, including how cryptographic keys are managed, the definitions of all transmitted data using Abstract Syntax Notation One with Distinguished Encoding Rules (ASN.1/DER), and details of the networking infrastructure. Together, the documents allow an implementer to interoperate with any implementation that shares both layers of the specification.

Based on our experience with Uptane, we attest that the dual-layer specification offers numerous benefits beyond accommodating conflicting needs. By delegating all the fine-grained formatting details for particular systems to POUFs, those preparing the Uptane Standard could focus just on essential design elements that are broadly applicable. In addition, POUFs provide a common language to discuss interoperability, even when implementers are working on closed-source projects. With a POUF, adopters can provide the relevant formatting and networking information without releasing proprietary designs.

The main contributions of this work are as follows:

- We point out the limitations of traditional standards, particularly as they apply to the automotive industry.
- We introduce the idea of a dual-layer specification as a means to provide customization options for a standard that contains relatively minimal implementation details.
- We develop a sample dual-layer specification to demonstrate the applicability of the approach for the Uptane framework.

2 Background

Functional safety is a primary concern in the automotive industry, and both it and quality are measured against not only regulatory requirements but also international standards [15]. As such, the contents of standards can be very important to decision making by OEMs and suppliers. Below we share a brief look at how standards are written, and then describe the Uptane framework, which inspired the dual-layer specification. We also outline the key elements of interoperability that would be included in a POUF layer.

2.1 Standards for the Automotive Industry

Standardization subjects a technology to a careful and critical examination before it is codified in a set of required steps. Once written, standards allow organizations within an industry to use the same basic procedures for quality, safety, and/or security [24], ensuring consistency across models and brands. While standards and regulations often have the same goals, the former are generally voluntarily implemented, while the latter are mandated by a governmental entity [39]. In addition, standards are generally the result of consensus among a designated group of experts, following a process to get community feedback on one or more drafts [30] [10] [11] [29]. This process provides assurance that a standard codifies the current best practices for a product, and can demonstrate quality to customers and regulators [38].

The wording of a standard must be explicit in order to distinguish between elements that must be implemented to achieve compliance, those that are recommended (i.e., best practices) but not essential, and those that are completely optional. Most standards will follow practices established within its industry or field, including the use of clearly defined keywords. The Uptane Standard uses keywords established in IETF BCP 14, a Best Current Practice document published by the Internet Engineering Task Force. Under BCP 14, any design element designated as MUST or SHALL is required for compliance, SHOULDs are recommendations, and MAYs are optional [4]. The number of SHOULD and MAY elements greatly affect how implementations of a standard interoperate with others. The Uptane Standard has 94 instances of SHOULDs and MAYs, an intentional word choice that allows for wide variance in implementations [34].

2.2 Elements of Interoperability

The SHOULDs and MAYs described above can include detailed specifications and formats essential to interoperability. Elements such as what data is transmitted, how it is sent, and what data binding formats are used to encode it need to be compatible for implementations to successfully send data to each other.

A data binding format specifies how the data elements are to be encoded (e.g., character set and repertoire restrictions for string data) for transmission and the order in which they must appear (i.e., serialization and marshalling). A number of data binding formats, including JavaScript Object Notation (JSON), Extensible Markup Language (XML), Hypertext Markup Language (HTML), Comma-Separated Values (CSV), and Concise Binary Object Representation (CBOR) [31, 40, 13, 26, 2], have been introduced for use in software, and in standards specifications. All of these formats are programming language neutral and are designed to allow data to be sent reliably from one application or system to another. Data elements are transformed into and retrieved from these data binding formats in a consistent way.

2.3 Uptane Software Update Framework

The Uptane software update framework, which motivated our work on the dual-layer specification, began development early in 2016. Supported by the US Department of Homeland Security, it was formally introduced to the wider community the following year [20]. Since then, Uptane has achieved very rapid adoption in the automotive industry, in part because input was solicited at every stage of development from industry insiders, including representatives of OEMs that manufacture 78% of all cars on US roads [33][18].

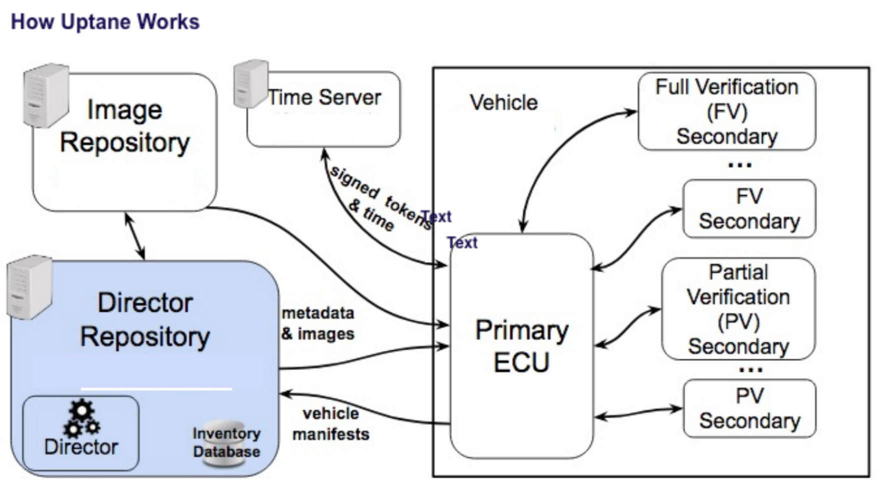


Fig. 1. A graphic depiction of the components and operations used by Uptane [17].

Uptane uses multiple servers, known as repositories, to download images and verify their authenticity by checking accompanying metadata before and after downloading. This significantly reduces the chance that malware could be introduced during the software update process. The Image Repository, shown at the top of the left-hand side of Figure 1, is the keeper of every image currently deployed, along with the metadata files that prove their authenticity. The Director Repository, shown at the bottom left-hand side, controls what software is distributed to each Electronic Control Unit (ECU) in the vehicle. The Time Server at the top of the diagram is one way for vehicles to access the current time in a cryptographically secure way, since many ECUs do not have an internal clock for verifying the freshness of messages, images, metadata, and certificates.

ECUs in a vehicle implementing Uptane are divided into two categories: primary and secondary. One ECU in each vehicle that has access to additional storage space, computing power, and an internet connection is designated as a Primary ECU and will be responsible for transmitting information from the repositories to the other ECUs in the vehicle. The ECUs receiving this information are called Secondary ECUs.

During the update process, the Primary ECU and the repositories communicate certain data, including image metadata, evidence of the current time, and the status of currently installed images. The process starts when the ECU sends its vehicle version manifest, a compilation of all the signed information about existing images, to the Director Repository. The Director uses this input to determine which images should be installed next. The metadata for these selected images is then sent to the ECU from both the Director Repository and Image Repository. The ECU will verify the image by checking cryptographic signatures and comparing metadata from the Director Repository and Image Repository. If the verification finds no issues, the images can be downloaded for the target ECUs and the vehicle version manifest will be updated.

3 Developing and Managing a Dual-Layer Specification

A dual-layer specification is composed of two separate documents that together provide a complete set of instructions for implementation. In order to separate the basic operating principles from the fine-grained implementation details, we developed a set of questions, the answers to which can determine how essential a particular step or design detail within a specification might be.

This section first describes the design criteria that evolved from creating the Uptane Standard. The three criteria listed in Section 3.1 not only guided design choices, but also helped to determine points where greater flexibility might be required in the existing specification. Section 3.2 then presents a workable way to determine what aspects of the specification are critical. From these two sets of statements and questions, a structure can be determined for a dual-layer specification.

3.1 Criteria for a Flexible Design

When creating the Uptane Standard, it became clear that a traditional protocol standard, which ensures compliance through adherence to explicit specifications, would not provide either the flexibility or the simplicity needed for widespread adoption of the framework within the automotive marketplace. Yet, we were aware that issues with the limitations of traditional standards were not confined to our immediate work. Therefore, in setting the basic design criteria for a new approach to standardization, we looked to address issues that transcend the automotive space. These criteria include:

1. **The design must work within the context of existing systems.** Potential adopters have existing systems that cannot be entirely rewritten to follow the wire protocol required by any new technology they adopt. Therefore, the new standard format must allow for some flexibility in how the requirements are applied to any given system.
2. **The design should allow implementers to find common ground without giving up trade secrets.** While legacy systems have particular requirements, there are some unifying ideas across these systems. If the same design would work on multiple systems, it would be beneficial for these systems to be able to interoperate and share best practices. This requires a format that allows such sharing without the need to reveal complete system designs. This is especially true in industries like the automotive market where proprietary practices dominate.

3. **The design should simplify specification and purchasing choices for implementing entities.** OEMs in the auto industry work with many suppliers in the design of a vehicle, as do manufacturers of any complex product or service. If multiple suppliers were able to create interoperable implementations for use on the same vehicle, criteria other than interoperability could be considered in choosing suppliers.

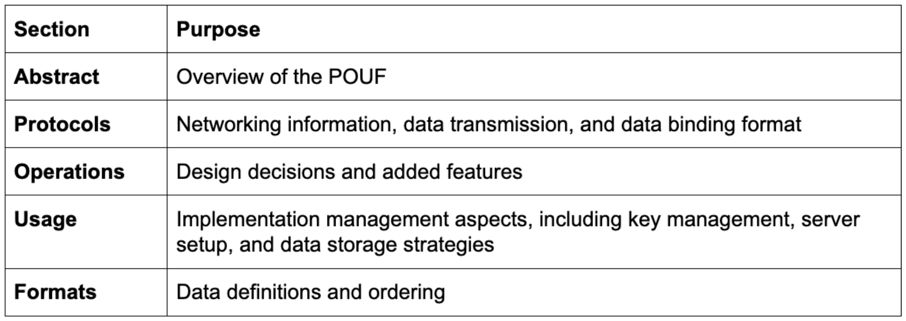
These criteria suggest a new approach to developing standards that looks to offer some options for how critical operations are conducted. To ensure these options do not compromise the goals of the standard, we need to determine what elements of an implementation are most vulnerable in terms of security, efficiency, or functionality.

3.2 Determining Critical Operations

As discussed in Section 2, the standard layer of a dual-layer specification uses the keywords MUST and SHALL to indicate required steps and procedures, while MAY and SHOULD indicate elements that are optional or recommended. When creating a dual-layer specification, authors must revisit the features required by MUST or SHALL to determine whether they could instead be specified as MAYs or SHOULDs and further described in the POUF layer. To make this distinction between required and flexible features, standards authors can answer a series of questions.

– What is the overall purpose of the specification? Is security the ultimate goal of its implementation?
– What vulnerabilities are associated with the feature in question?
– Is there one format/approach, etc that is the only solution for protecting against these vulnerabilities?
– Are there any known formats which are clearly less effective against these vulnerabilities?
– What is the goal of the feature? Is there a different, valid way to achieve this goal?
– Does the stipulated approach allow for backwards compatibility as systems evolve?
– Is the stipulated approach the most efficient choice?
– Are there any specific risks to security or functionality in using alternative approaches?

Such questions will provide a clear dividing line between what must be stipulated and what could instead be labeled as a MAY or SHOULD in the standard. Descriptions of which of these “good, but not essential” practices are used by an implementation are included in the POUF and enable us to achieve the design criteria set in Section 3.1.



Section	Purpose
Abstract	Overview of the POUF
Protocols	Networking information, data transmission, and data binding format
Operations	Design decisions and added features
Usage	Implementation management aspects, including key management, server setup, and data storage strategies
Formats	Data definitions and ordering

Fig. 2. The sections of a POUF

3.3 POUF Contents

A POUF can be thought of as an inventory of all the operations and procedures used by an implementation, as well as specific details for applying the standard on a network. As POUFs are shared with other organizations, these decisions can propagate the establishment of best practices. The core data of a POUF lies in the implementation details of the four categories that make up its name: Protocols, Operations, Usages, and Formats as illustrated in Figure 2.

The Protocols section contains explicit networking information, including the encoding and networking protocols used, the manner in which files are hosted on repositories, and the types of requests, responses, and notifications the system supports. In addition, it should contain a Message Handler Table listing all supported messages sent or received during the update process.

The Operations section documents any features that are required by the POUF in addition to the standard. The POUF may upgrade the requirement level of features described as SHOULD or MAY in the standard or may exclude these features. As a POUF must be based on the requirements of the standard, any decision beyond those mandated in the standard–even ones referenced in the Standard as options–must be explained. By documenting these decisions, a potential adopter would better understand the system requirements it was written to address. In addition, the POUF also preserves “institutional memory” of why specific decisions about the implementation were made.

The Usage section describes how data is managed by the implementation, including server setup procedures and key creation, distribution, revocation, and rotation. It should also contain a Data Table delineating what keys and other data each entity is able to access.

The Formats section is composed of all data definitions used during data transmission, including all messages described in the Protocols section.

3.4 Using POUFs

A standard implementer following an existing POUF should do the following:

– Use the abstract and header to determine whether the POUF is appropriate for their circumstances.
– Read the Protocols section for the details of networking infrastructure to be set up.
– Add all the relevant features described in the Standard and the Operations section of the POUF, following guidance in the Usage section for all the data that must be available.
– Set up servers and key management infrastructure as described in the Usage section.
– Read the Formats section for the exact layout of data in each message to be sent over the network.

Because POUFs are written to support the deployment of a specific version of a standard, the POUF header must include the standard version number. This version number must be updated whenever a new version of the standard is released. When minor changes are made to the standard, they can just be described in the abstract. However, if larger changes are made, particularly ones that are not backwards compatible, a new POUF will need to be prepared.

An updated POUF always replaces its predecessor, though the old version may continue to be available to implementers for a transitional period of time (e.g., one year). This flexibility can allow implementers to continue to use the older POUF version if they choose not to update immediately.

4 Implementing a Dual-layer Specification

To test the feasibility of using a dual-layer specification, we needed to apply it to a particular product or process. As the Uptane Standard was written to allow for flexible implementations, we decided to write a POUF for the Uptane Reference Specification. This section provides an example of the decision-making that goes into clarifying the boundaries between the two layers.

As the name suggests, the Uptane Reference Implementation is a basic implementation of the Uptane Standard. By design, it includes no major customizations to the Uptane Standard. It does, however, include several features that were excluded from the Standard to make its adoption viable as an independent update system. Primarily, these exclusions relate to the use of a Time Server to provide a signed current time to vehicle ECUs and support for asymmetric encryption between ECUs and repositories.

4.1 What goes in the Standard

To implement a dual-layer specification for Uptane, we first had to determine what pieces of the specification belonged in the Standard, and what could be specified in a POUF. As Uptane is a secure update system, determining whether an aspect of the specification is security critical was our primary criteria. Here are a few examples of elements that were included in the Standard, and others that were left to the POUF layer.

Central to the security promises of Uptane is it's metadata validation procedure that verifies updates to ensure the file delivered is the most up-to-date version of the file requested. We reviewed this process and determined that some elements were security critical to all implementations, including many of the fields in the metadata files, the process of metadata verification, and the delegation of trust to separate roles. These concepts must be mandated in the standard. On the other hand, how metadata is made available to vehicles, how encryption keys should be stored and managed, and any additional metadata fields, such as custom installation instructions, are more flexible. As such, these features were not mandated in the Standard, but instead were left to be specified in individual POUFs. Though POUF elements are vital for the functioning and ease-of-use of the system, the exact method of their implementation is not critical for security.

Another element designated for the POUF was how the current time was obtained. An accurate reading of the current time is essential in preventing such threats as freeze or rollback attacks. If an attacker is trying to send old updates to a vehicle, the vehicle must be able to detect that these updates are no longer valid. However, there are many ways a vehicle might securely obtain the time, and some vehicles may already have one built into their infrastructure. Therefore we split this requirement between the Standard and the POUF. As shown in Figure 3, the Standard requires that a vehicle have access to a secure source of time, and describes what this time will be used for in the update process. Meanwhile the POUF describes how the secure time will be received and verified.

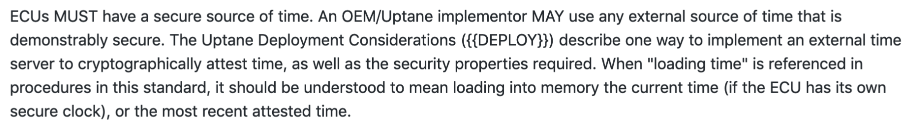

ECUs MUST have a secure source of time. An OEM/Uptane implementor MAY use any external source of time that is demonstrably secure. The Uptane Deployment Considerations ({{DEPLOY}}) describe one way to implement an external time server to cryptographically attest time, as well as the security properties required. When "loading time" is referenced in procedures in this standard, it should be understood to mean loading into memory the current time (if the ECU has its own secure clock), or the most recent attested time.

Fig. 3. The description of the Time Server in the Uptane Standard.

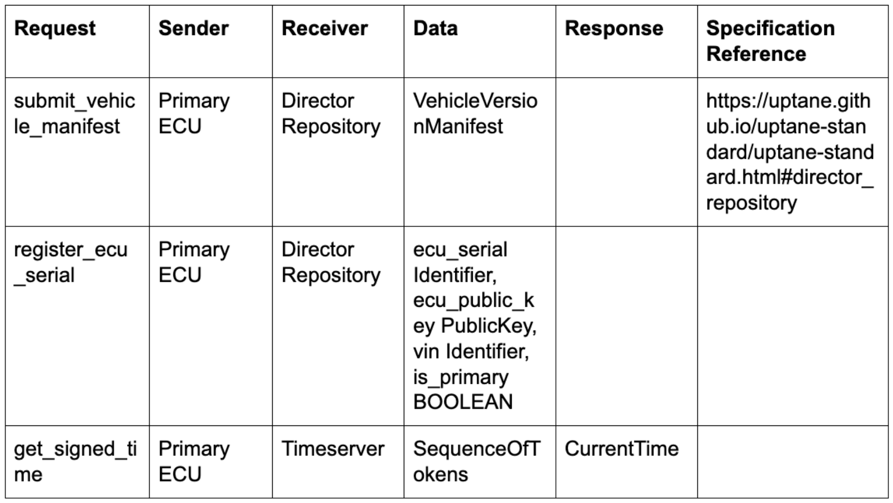
4.2 What goes in a POUF

First, a POUF includes metadata sections presenting a high-level overview of its design goals. In this case, the metadata sections describe the purpose of the Uptane Reference Implementation and tie the POUF to version 1.0.0 of the Uptane Standard. This introductory information is followed by a detailed account of the Protocols, Operations, Usage, and Formats used in the Uptane Reference Implementation.

Protocols The Protocols section must include descriptions of all network traffic sent by the implementation so that a reader could create a compatible API.

For the Uptane Reference Implementation POUF, we first observed all network traffic sent by the implementation, and recorded what data was sent, how it was encoded, and how it was accessed by vehicles and servers. This information was included in the POUF as a Protocol description, a Message Handler Table, and a description of all hosted files.

The Protocol description presents a rationale for using the ASN.1/DER protocols to define data structures, and the XML remote procedure call protocol (XML-RPC) to exchange data over the networks. In addition to its ease of implementation and readability, ASN.1 notation is familiar to many in the automotive industry and thus is an appropriate choice for the Uptane community. Recognizing though that there are a few known vulnerabilities in this wire protocol, this section also contains advice to implementers about using certified secure ASN.1 libraries. This is an example of the type of tradeoffs the dual-layer specification can offer. It allows its target clientele to use familiar systems but makes clear any potential security implications within this choice.



Request	Sender	Receiver	Data	Response	Specification Reference
submit_vehicle_manifest	Primary ECU	Director Repository	VehicleVersionManifest		https://uptane.github.io/uptane-standard/uptane-standard.html#director_repository
register_ecu_serial	Primary ECU	Director Repository	ecu_serial Identifier, ecu_public_key PublicKey, vin Identifier, is_primary BOOLEAN		
get_signed_time	Primary ECU	Timeserver	SequenceOfTokens	CurrentTime	

Fig. 4. The Message Handler Table from the Uptane Reference Implementation POUF.

The Message Handler table includes a description of all messages sent over the network. The table lists all requests supported by the POUF, with columns for the request, sender, receiver, included data, expected response, and a reference link to the Uptane Standard. As shown in Figure 4, these fields each describe part of the request:

- The *request* is the name used by XML-RPC to make the request.
- The *sender* is the Uptane entity that sends the request
- The *receiver* is the Uptane entity that receives the request

– The *data* lists the names of data formats that are described in detail in the Formats section.
– The *response* lists the data, if any, that the receiver will reply with.
– The *link to the Uptane Standard* is optionally used to link the reader to more information about why the request is made.

Lastly, the Protocols section contains a description of the metadata files used in the Uptane Reference Implementation that are hosted on a server to be accessed by vehicles during the update process. This information includes a description of all the hosted files, how the location and filenames for all downloaded files will be determined, and the client-server relationship between networked devices.

Operations The Operations section contains any requirements of the POUF in addition to those in the specification.

For the Uptane Reference Implementation, this section charts all the SHOULDs and MAYs referenced in the Uptane Standard with instructions on how the Reference Implementation will handle these situations. These SHOULDs and MAYs include all the recommended practices and features that are mentioned, but not required in the Standard. This section also documents features like the Time Server, which are used in the Reference Implementation but not described in the Uptane Standard.

In order to make the section more readable, we grouped related items by topic, such as metadata features, encryption algorithms, and delegations. Within these groupings, we documented each customization and included a rationale for each design decision. These rationales provide insight into the goals of an implementation and can be used to preserve the decision making process. See Figure 5 for an example of how the Time Server is documented in the POUF.

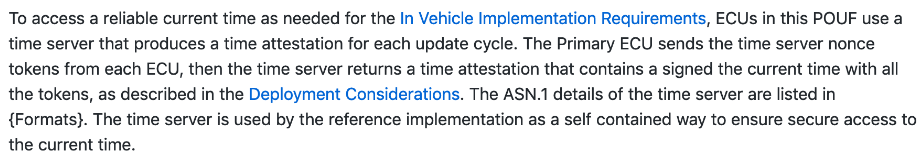
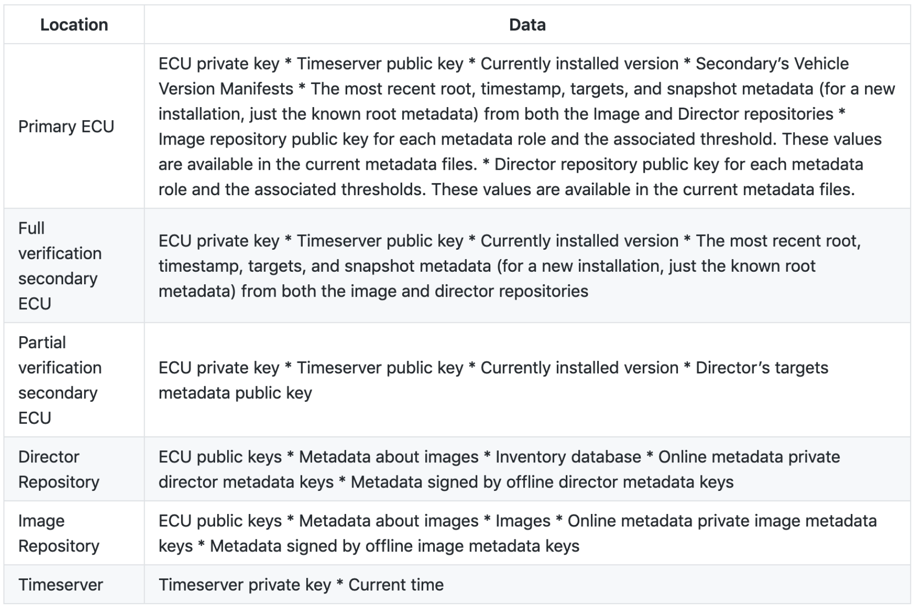

To access a reliable current time as needed for the In Vehicle Implementation Requirements, ECUs in this POUF use a time server that produces a time attestation for each update cycle. The Primary ECU sends the time server nonce tokens from each ECU, then the time server returns a time attestation that contains a signed the current time with all the tokens, as described in the Deployment Considerations. The ASN.1 details of the time server are listed in {Formats}. The time server is used by the reference implementation as a self contained way to ensure secure access to the current time.

Fig. 5. The description of the Time Server from the Operations section of the Uptane Reference Implementation POUF.

Usage The Usage section contains details relevant to the setup and management of the Reference Implementation, including key management, server setup, and data storage. As the implementation was written before the POUF, this section documents actual experiences in implementing these processes. For example, though the Uptane security model encourages the use of offline keys and separation of responsibilities where possible, in this implementation all keys are stored online with a key threshold of one. Despite the overall safety goal of the Standard, this implementation is not designed to be used in practice, and its main goal is to demonstrate the efficacy of Uptane. So this section of Reference Implementation POUF details this setup, including the rationale for the above choices.

The Usage section also details the steps needed to set up the servers before the Reference Implementation can be used. This includes the databases that need to be created, the manner in which servers discover vehicles, and how key initialization is performed. Unlike what is outlined in the Uptane Standard, this section not only describes what servers are needed, but also stipulates their architecture. For example the Uptane Standard mandates that an Image Repository must exist and that it must contain target files and metadata that can be accessed by ECUs. In addition to these requirements, the POUF describes the steps that an implementer must take to set up the Repository, as well as the database schemas used to keep track of ECUs.

Lastly, the Usage section will contain a table documenting the location of all the essential data required to build an implementation of the POUF. This includes what data each entity stores, and what public or private keys can be accessed to ensure that public key cryptography is properly initiated. The Data Table from the Reference Implementation POUF is shown in Figure 6.



Location	Data
Primary ECU	ECU private key * Timeserver public key * Currently installed version * Secondary's Vehicle Version Manifests * The most recent root, timestamp, targets, and snapshot metadata (for a new installation, just the known root metadata) from both the Image and Director repositories * Image repository public key for each metadata role and the associated threshold. These values are available in the current metadata files. * Director repository public key for each metadata role and the associated thresholds. These values are available in the current metadata files.
Full verification secondary ECU	ECU private key * Timeserver public key * Currently installed version * The most recent root, timestamp, targets, and snapshot metadata (for a new installation, just the known root metadata) from both the image and director repositories
Partial verification secondary ECU	ECU private key * Timeserver public key * Currently installed version * Director's targets metadata public key
Director Repository	ECU public keys * Metadata about images * Inventory database * Online metadata private director metadata keys * Metadata signed by offline director metadata keys
Image Repository	ECU public keys * Metadata about images * Images * Online metadata private image metadata keys * Metadata signed by offline image metadata keys
Timeserver	Timeserver private key * Current time

Fig. 6. The Data Table from the Uptane Reference Implementation POUF

Formats The Formats section must include the encoding and format of all data transmitted by the implementation. As ASN.1 encoding is used by the Uptane Reference Implementation, its POUF contains the ASN.1 definitions of all data elements. For brevity, this section starts with common definitions that are used in multiple messages, and then describes how these data elements are used in the messages sent between Uptane entities.

5 Discussion

Before proposing adoption of the dual-layer specification design for other standards, we needed to resolve a few additional issues, such as how to store and access POUFs, and identifying possible places where the flexibility offered by such a specification might not be desirable. This section addresses these issues, and also evaluates our success in overcoming the specific limitations of conventional standards that had motivated our new approach for Uptane.

5.1 Storing and Accessing POUFs

A primary issue in POUF management is how to make these instructions accessible to future implementers. This management choice will depend in part on how involved the implementers wish to be in the testing and validation of their own product. While listing POUFs does not automatically imply the host site is responsible for its contents, there could still be concerns about potential liability. Some of the possible storage options are:

- A central team could post links to externally stored POUFs allowing for easy storage without any actual or perceived endorsement.
- Each implementer could choose to store their own POUF without any added validation (although that lack of self-validation would diminish the marketplace value of the POUF).
- A central team could validate POUFs using a given standard test suite.
- Standards authors could decide to trust implementers to write valid POUFs, and then accumulate crowd-sourced information about the quality of the POUFs. (This would not be an acceptable alternative in a regulated industry with functional safety and/or security legal requirements).
- POUFs could be managed by an OEM or other third party that validates the POUF and distributes it to suppliers.

All of these POUF storage methods have benefits and drawbacks that should be balanced for a particular standard. If a standard has an active group of maintainers, then externally validating all POUFs can provide clear choices for adopters. Storing the POUFs externally or simply trusting that they are valid could require less work for the standards development team, but any tampering with these POUFs could reduce the credibility of the standard itself. If POUF management is left to an OEM, it gives that entity the ability to decide when information in the POUF should be shared, for example in an OEM RFC. However, this also makes the OEM responsible for ensuring compliance.

5.2 Tradeoffs

Despite the benefits, careful consideration should be given before creating a dual-layer specification. The team writing the standard should be aware of any possible contradiction between a proposed POUF and the corresponding standard, as it could cause a security vulnerability or corrupt a required feature. The degree of flexibility in a standard may need to be dialed back for technologies in which optional approaches could pose a particular safety or security risk.

Another drawback to dual-layer specifications could be the need to check against two references rather than one while creating an implementation. This added complexity could increase the chance that a required feature might be left out of one or both documents. As such, a dual-layer specification may take a bit longer to create in order to allow time for cross-checks between the two documents. This could be minimized by combining the two documents into a unified design description before starting work on an implementation.

In addition, dual-layer specifications are not suited for every standard development effort. In some cases, interoperability is such a key feature in the operation of a given process that a dual-layer specification should not be used. For example, standards for networking protocols and data binding formats require most of the information that would typically go into a POUF, including the data formats and any metadata transmitted. Interoperability between communicating devices is a key feature of all networking protocols.

The takeaway lesson is that at the start of any standardization effort, the creators should conduct the step recommended in Section 3.2 and ask themselves a number of questions about the function of the technology being standardized, the target audience of users and implementers, and the risks and benefits of flexibility in each procedural step. The answers will determine how flexible the resulting standard should be.

5.3 Does the Dual-Layer Approach Overcome the Limitations of Conventional Standards?

In Section 3.1 we set three flexibility criteria for our dual-layer specification that would make it applicable within and beyond the automotive space. Here, we discuss how our design addresses each of these criteria in the context of the Uptane Standard.

- **The design must work within the context of existing systems.** A dual-layer specification separates the design of a framework from the context of an implementation. The design is laid out in the standard while information about how this design fits into the context of a larger system goes in a POUF. The dual-layer specification for the Uptane framework allows the security goals of Uptane to be specified in the standard while the leaving enough flexibility for the design to use data formats, networking protocols, and other implementation details that work with any existing systems. This separation allowed Uptane to integrate with existing OEM backends.

– **The design should allow implementers to find common ground without giving up trade secrets.** Dual-layer specifications allow implementers to share the same standard for a baseline of security, reliability, or functionality. The POUF layer provides a mechanism to allow for interoperability and could be used to share these relevant details without divulging the code of an implementation. For example, the Protocols and Formats sections of a POUF allow implementers to share how data can be transmitted to an implementation. In addition, the POUF layer is optional. An organization can have a valid implementation of the standard without divulging any information that could be considered proprietary. For Uptane, this separation between standard and POUF allowed suppliers and OEMs to build interoperable implementations without divulging trade secrets.
– **The design should simplify specification and purchasing choices for implementing entities.** The POUF layer allows for interoperable and interchangeable implementations of a standard. OEMs interested in using Uptane are able to specify the POUF they would like implemented and any supplier working with the OEM could implement that POUF to create interoperable implementations. In this way, OEMs can consider factors other than interoperability, such as cost or efficiency when choosing a supplier, and suppliers can make an implementation that will interoperate with an OEM's system by default.

6 Related Work

The dual-layer specification presented in this paper is an attempt to inject flexibility into documents that, by nature, require exact specifications. Before proposing such a fundamental change, we examined other works that present the current approach to the standardization process, as well as previous attempts to add flexibility, and other new approaches to preparing these documents.

6.1 The Standardization Process

A quick survey of standards organizations—including the Internet Engineering Task Force (IETF)[30], the Institute of Electrical and Electronics Engineers[10], the International Standards Organization[11], and the Society of Automotive Engineers[29]—shows most follow a common process for preparing these documents. A need for formal guidelines in the design, development, or implementation of a product or technology is identified. A group of relevant stakeholders is then assembled to write, review, and revise a draft standard which is released to a broader audience. The audience provides feedback which guides additional revision before an edited version is released. Each standards organization designs their process to balance the need for completeness of a standard and timeliness of its release.

When new technologies are standardized, one of the key considerations is often how to make these technologies interoperable with existing devices and standards. In an analysis of standardization efforts, Cargill includes “The standard is finished and implementations are incompatible” as one of the six reasons that standards often fail [5]. It follows that addressing interoperability during the design of a standard will help the standard to succeed. LaMaire et al describe how new wireless technologies require changes to existing standards to ensure future interoperability [19]. The standards described, including IEEE 802.11 [6] address interoperability concerns by mandating all networking operations. Any changes to the wire protocol would then conflict with the existing standard and so result in a new release of the standard with these changes incorporated [7][8]. Alternately, interoperability can be addressed after a standard is written through interoperability testing [28]. Such a test is often used in conjunction with standards that are intended to specify interoperability.

Dual layer specifications address interoperability during the standardization process while still allowing for flexible implementations. The POUF describes compatibility so that implementations can be interoperable when needed, but not at the cost of an overly rigid standard.

6.2 Flexibility in Standards

Braa et. al. discuss the need for flexibility in information infrastructure in the creation of technology standards for health data in developing countries [3]. They note that flexibility is important in technology because of the likelihood of future improvement, user requirements that change over time, and integration with other technologies. To address these issues, the authors propose using standards that are vertically and horizontally modular so that each standard only describes a small portion of the system and can be integrated with other standards. A dual-layer specification can increase flexibility by allowing each standard to have multiple POUFs for integration with different technologies. These POUFs can be adjusted over time without affecting the underlying standard.

Beck introduces a formal model for layered systems built around the hourglass model [1]. This model, which is used in practice in internet protocols and operating systems, allows multiple applications to interoperate through a common layer specification. The author discusses the benefit of making the connecting layer as small as possible to allow for a wider variety of applications, and introduces the Deployment Scalability Tradeoff to formalize this relationship between interface size and scalability. A dual-layer specification is similar to the top half of the hourglass model in that multiple POUFs interact with the same standard, while greater flexibility in the standard allows for a wider variety of valid POUFs. However, interoperability in dual-layer specifications is done through multiple POUFs instead of a single interface because the goal of a dual-layer specification is to have a single standard operate with multiple systems, while the goal of the hourglass model is to create systems that can interoperate through a shared interface.

Some standards allow implementers to choose from a fixed number of options, based on their individual priorities. For example, TPM 2.0 [32] allows for five possible security levels and an implementer must choose one as described in the standard. Adopters can write profiles to describe the subset of TPM that they are using. For example, the Windows TPM 2.0 profile lists the security level chosen and all features that must be included for interoperability [23]. Unlike POUFs, these profiles do not describe anything outside the scope of the standard. They simply list which options from the standard are implemented. This process allows for some limited flexibility to manage security and functionality tradeoffs.

6.3 New Approaches to Standardization

To address limitations in existing standardization procedures, some groups are working on new ways to create standards. IEEE-ISTO was created to standardize software with a faster development process that uses industry experts organized into alliances or consortia [27]. The Internet Research Task Force (IRTF) Crypto Forum Research Group combines work on cryptography standards with the creation of informational materials to offer guidance on the use of cryptographic mechanisms [9]. The IEEE P2413 working group [14] is working to create general security, privacy, and safety guidelines for IoT devices. The group is considering solutions that address incompatible implementations, including creating a standard that allows for flexibility in implementations to allow devices with various primary functions to interoperate in an IoT system. TLS 1.3 was developed using a new process of proactive development designed to find security flaws before the release of a standard [25]. A study of the Kaleidoscope conference found that transparency and academic involvement can improve standards quality [12]. All of these strategies improve the standardization process, but none of them allow the combination of interoperability and flexibility offered by a dual-layer specification.

7 Future Work

A dual-layer specification can be applied to any standard development effort that requires both rigid security or functionality requirements and a need for broad stakeholder buy-in. This includes standards for intra-vehicle communication, safety requirements, and security practices. In all of these cases the standard layer provides a description of the security and functionality requirements while the POUF layer provides enough flexibility to get broad support for these requirements. Beyond the automotive space, smart homes, IoT devices, and medical devices could benefit from standards that provide guarantees across a variety of implementations.

Dual-layer specifications and the implementations that use them should be subject to rigorous testing to ensure adherence to both layers of the specification. If two implementations use the same POUF, there could be interoperability testing between these implementations. In addition, every POUF should be tested to ensure it does not contradict the standard. These and other testing efforts would help ensure that all implementations of a dual-layer specification achieve the functionality and interoperability requirements of the system.

8 Conclusion

The dual-layer specification model introduced in this paper gives standards designers a method to minimize the details that must be explicit in a standard, potentially with no loss of security properties. The compromise that resolves the conflict between requiring interoperability and working with legacy systems is addressed by the POUF, a second layer that complements the corresponding standard and allows groups of implementers to create interoperable implementations. By using POUFs to address interoperability, the standard can focus on functionality or security goals and specify these in a way that can be adopted by legacy systems. We verified the feasibility of the dual-layer specification model by applying a POUF to the Reference Implementation of the Uptane Standard framework. This application of the dual-layer specification model demonstrates its effectiveness in separating critical functionality from interoperability concerns. This new model of standardization has the potential to revolutionize the standards process by allowing for more flexibility while ensuring key security and functionality features are consistent across an industry.

References

1. Beck, M.: On the hourglass model. Communications of the ACM **62**(7), 48–57 (july 2019)
2. Bormann, C., Hoffman, P.: Concise binary object representation (cbor). Standard RFC 7049, Internet Engineering Task Force (2013)
3. Braa, J., Haneth, O., Heywood, A., Mohammed, W.: Flexible standards. In: IEEE SIIT (2005)
4. Bradner, S.: Key words for use in rfcs to indicate requirement levels. Standard, Network Working Group (1997)
5. Cargill, C.F.: Why standardization efforts fail. The Journal of Electronic Publishing **14** (2011)
6. C/LM - LAN/MAN Standards Committee: Ieee standard for wireless lan medium access control (mac) and physical layer (phy) specifications. Standard IEEE 802.11-1997, Institute of Electrical and Electronics Engineers (1997), https://standards.ieee.org/standard/802_11-1997.html
7. C/LM - LAN/MAN Standards Committee: Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. Standard IEEE 802.11-2012, Institute of Electrical and Electronics Engineers (2012), https://standards.ieee.org/standard/802_11-2012.html
8. C/LM - LAN/MAN Standards Committee: Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. Standard IEEE 802.11-2016, Institute of Electrical and Electronics Engineers (2016), https://standards.ieee.org/standard/802_11-2016.html
9. Crypto forum research group. https://irtf.org/cfrg, last accessed 8 June 2020

10. Develop standards. https://standards.ieee.org/develop/develop-standards/process.html, last accessed 5 November 2019
11. Developing standards. https://www.iso.org/developing-standards.html, last accessed 5 November 2019
12. Griffin, P.H.: Standardization transparency. Chen L., McGrew D., Mitchell C. (eds) Security Standardisation Research **8893**, 57–68 (2014)
13. Html 4.0 specification. Standard, W3C Recommendation (1997)
14. IEEE Standards Association: Standard for an Architectural Framework for the Internet of Things (IoT) (2019)
15. ISO/TC 22/SC 32 Electrical and electronic components and general system aspects: Road vehicles functional safety. Standard ISO 26262, International Standards Organization (2018), https://www.iso.org/standard/68383.html
16. ISO/TC 22/SC 32 Electrical and electronic components and general system aspects: Road vehicles cybersecurity engineering. Standard ISO 21434, International Standards Organization (2020), https://www.iso.org/standard/70918.html
17. Kuppusamy, T.: Securing over-the-air updates against nation state actor. Uptane 2018 SAE meeting (2018)
18. Kuppusamy, T., DeLong, L., Cappos, J.: Uptane security and customizability of software updates for vehicles. IEEE Vehicular Technology Magazine pp. 66–73 (2018)
19. LaMaire, R.., Krishna, A., Bhagwat, P., Panian, J.: Wireless 1ans and mobile networking: Standards and future directions. IEEE Communications Magazine pp. 86–94 (1996)
20. Mathews, L.: Uptane will protect your connected car from hackers. Forbes (2017)
21. McDonald, C.: How the development of standards will affect the internet of things. Computer Weekly (2014)
22. McLaughlin, P., Sherouse, O.: The mclaughlin-sherouse list: The 10 most-regulated industries of 2014. Mercatus Center (2016)
23. Microsoft: TPM V2.0 Command and Signal Profile (October 2018)
24. Nist glossary. https://csrc.nist.gov/glossary/term/Standard, last accessed 29 June 2019
25. Paterson, K.G., van der Merwe, T.: Reactive and proactive standardisation of tls. Chen L., McGrew D., Mitchell C. (eds) Security Standardisation Research **10074**, 160–186 (2016)
26. Shafranovich, Y.: Common format and mime type for comma-separated values (csv) files. Standard RFC 4180, Internet Engineering Task Force (2005)
27. Shaw, N.E.: Emerging technologies and the new face of standards. IEEE Industry Standards and Technology Organization (2017)
28. Shaw, N.E.: Certification programs: Different models that work. IEEE Industry Standards and Technology Organization (2018)
29. Standards development process. https://www.sae.org/standardsdev/devprocess.htm, last accessed 5 November 2019
30. Standards process. https://www.ietf.org/standards/process/, last accessed 5 November 2019
31. T Bray, E.: The javascript object notation (json) data interchange format. Standard RFC 8259, Internet Engineering Task Force (2017), https://tools.ietf.org/html/rfc8259
32. Trusted Computing Group: Trusted Platform Module Library Family 2.0 Level 00 Revision 1.59 (March 2020)
33. Uptane adoptions. https://uptane.github.io/adoptions.html, last accessed 29 June 2019

34. Uptane Alliance: Ieee-isto 6100.1.0.0 uptane standard for design and implementation. Standard IEEE-ISTO 6100, IEEE Industry Standards and Technology Organization (2019), https://www.iso.org/standard/43464.html
35. Uptane alliance homepage. https://ieee-isto.org/member_programs/uptane-alliance, last accessed 29 June 2019
36. Uptane reference implemenation. https://github.com/uptane/uptane, last accessed 29 June 2019
37. Uptane website. https://uptane.github.io/overview.html, last accessed 29 June 2019
38. Viardot, E.: Trust and standardization in the adoption of innovation. IEEE Communications Standards Magazine pp. 31–35 (2017)
39. What is the difference between a code, standard, regulation and specification in the electrical industry? https://blog.nvent.com/erico/erico-what-is-the-difference-between-a-code-standard-regulation-and-specification-in-the-electrical-industry/, last accessed 21 February 2020
40. Extensible markup language (xml) 1.0 (fifth edition). Standard, W3C Recommendation (2008), https://www.w3.org/TR/xml/