Direct Anonymous Attestation on the Road: Efficient and Privacy-Preserving Revocation in C-ITS

Benjamin Larsen[*], Thanassis Giannetsos[†], Ioannis Krontiris[∓], Kenneth Goldman[‡]
[*]Technical University of Denmark (DTU), Cyber Security Section, Denmark
[†]Ubitech Ltd., Digital Security & Trusted Computing Group, Greece
[∓] Huawei Technologies Duesseldorf GmbH, Munich, Germany
[‡] IBM T. J. Watson Research Center, Hawthorne, New York
{benlar@dtu.dk, agiannetsos@ubitech.eu, ioannis.krontiris@huawei.com, kgoldman@us.ibm.com}

ABSTRACT

Vehicular networks rely on Public Key Infrastructure (PKIs) to generate long-term and short-term pseudonyms that protect vehicle's privacy. Instead of relying on a complex and centralized ecosystem of PKI entities, a more scalable solution is to rely on Direct Anonymous Attestation (DAA) and the use of Trusted Computing elements. In particular, revocation based on DAA is very attractive in terms of efficiency and privacy: it does not require the use of Certificate Revocation Lists (CRLs) and revocation authorities can exclude misbehaving participants from a V2X system without resolving (i.e. learning) their long-term identity. In this paper, we present a novel revocation protocol based on the use of DAA and showcase a detailed design and modeling of the implementation on a real TPM platform in order to demonstrate its significant performance improvements compared to existing solutions.

KEYWORDS

Security, Privacy, Direct Anonymous Attestation, Trusted Platform Module (TPM), Revocation, C-ITS, V2X

ACM Reference Format:
Benjamin Larsen et. al.. 2021. Direct Anonymous Attestation on the Road: Efficient and Privacy-Preserving Revocation in C-ITS. In *Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '21), June 28–July 2, 2021, Abu Dhabi, United Arab Emirates.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3448300.3467832

1 INTRODUCTION

Connected vehicles, as part of the emerging Cooperative Intelligent Transportation Systems (C-ITS) are positioned to transform the future of mobility. This change is enabled by the exchange of messages between vehicles (V2V) and between vehicles and transport infrastructure (V2I), comprising together the overall vehicular communication (V2X). V2X communication systems are expected to greatly improve road safety and traffic efficiency while better supporting autonomous driving. V2X can also save lives by providing road hazard warnings to the driver and reducing collisions [3].

WiSec '21, June 28–July 2, 2021, Abu Dhabi, United Arab Emirates
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8349-3/21/06…$15.00
https://doi.org/10.1145/3448300.3467832

However, despite their benefits, privacy is a key concern in this facet of C-ITS, since the involved vehicle transmissions can be used to infringe the users' location privacy [28]. Many V2X applications rely on broadcasting continuous and detailed location information, as for example, through the Cooperative Awareness Messages (CAM), which are broadcasted unencrypted by vehicles at the frequency of 10 Hz. If this information is misused (all exchanged messages can be eavesdropped within radio range) can lead to the extraction of detailed location profiles of vehicles and path tracking [10]. Since there is usually a strong correlation between a vehicle and its owner [12], location traces of vehicles have the potential to reveal the movement and activities of their drivers.

Addressing this challenge, current approaches are based on PKI-based solutions [11] with privacy-friendly authentication services through the use of short-term anonymous credentials, i.e., *pseudonyms* [20]. The common denominator in such architectures is the existence of trusted (centralized) infrastructure entities for the support of services such as authenticated vehicle registration, pseudonym provision, revocation, etc. The location privacy is protected by requiring that each vehicle uses multiple pseudonyms, frequently changing from one pseudonym to another [11].

The use of changing pseudonyms can be considered the state-of-the-art in VANET privacy-enhancing technologies like the one that was recently proposed in [27]. Prominent solutions include the Security Credential Management System (SCMS) [27], which is a product of vehicle OEM consortia and the US Department of Transport (USDOT), and the Cooperative-ITS Certificate Management System (CCMS) developed by the European Committee for Standardisation (CEN) and European Telecommunications Standards Institute (ETSI), with support from the European Commission [6].

These architectures have several inherent drawbacks stemming from the fact that they are based on a complex and centralized ecosystem of PKI entities, which we need to trust for issuing and distributing pseudonym certificates. First, a technical and organizational separation of capabilities between the PKI entities is required to cope with internal attackers, resulting in a very costly solution to implement in practice. The bottleneck of having to connect to the back-end infrastructure to acquire pseudonym certificates is resolved by downloading a larger pseudonym pool size, which then provides less protection against Sybil attacks. In the context of revocation policies for removing misbehaving nodes from the network, they are based on the dissemination of CRLs, which is an inefficient solution in terms of computational and communication overhead.

To address the aforementioned challenges of centralised PKI solutions, several researchers have suggested moving towards a decentralised approach, where trust is shifted from the back-end infrastructure to the edge [9, 24]. In its recent white paper on privacy-by-design aspects of C-V2X, 5GAA is also pointing to the need of more scalable and decentralised solutions in the future, eliminating the need for trust built around "federated infrastructures" [1].

As it has been shown by recent work, one way to do this is by leveraging the use of Direct Anonymous Attestation (DAA) and the incorporation of trusted computing technologies [9, 14, 24]. DAA, originally introduced by Brickell, Camenisch, and Chen [2], is a cryptographic protocol designed primarily to enhance user privacy within the remote attestation process of computing platforms, which has been adopted by the Trusted Computing Group (TCG) [23], in its latest specification.

Applying the DAA protocols for securing V2X communication results in the redundancy (and removal) of most of the PKI infrastructure entities, including the pseudonym certificate authority: vehicles can now create their own pseudonym certificates using an in-vehicle trusted computing component (TC), and DAA signatures are used to self-certify each such credential that is verifiable by all recipients. Furthermore, a DAA-based model supports a more efficient revocation of misbehaving vehicles that don't require the use of CRLs, removing, therefore all the computational and communication overhead that comes with it. Instead, when the Revocation Authority issues a revocation request, this triggers the TC of the misbehaving vehicle to delete all of its pseudonymous certificates and cryptographic key pairs, thus, rendering the TC unable to generate new pseudonyms in the future. However, the details of this process have not been shown and demonstrated so far, and its feasibility remains an open question.

In this paper, we provide a novel revocation protocol for C-ITS based on DAA that leverages the benefits of trusted computing to offer an efficient and privacy-respecting solution. Our protocol provides both revocations of the vehicle from the system and revocation of specific pseudonyms that cannot be reused again (referred to as hard and soft revocation, respectively). We provide an implementation of the protocol in a real TPM in order to demonstrate that the proposed solution is applicable to the real world and can meet the strict performance requirements, as documented in ETSI standards, and also verify that near-constant revocation time is achievable even when multiple pseudonyms are entailed.

2 MOTIVATION AND CONTRIBUTION

Revocation is a standard consideration for any C-ITS system. In case of a misbehaving vehicle, the wrongdoer can be evicted and be prevented from further participation. In the case of PKI-based solutions, the revocation can be done in standardized ways by adding the revoked certificates to a CRL, which is then published by the CA responsible for that trust domain. However, for vehicles using short-lived pseudonym certificates, things are more complicated. If a vehicle possesses multiple certificates that are unlinkable, every single certificate needs to be put on the CRL, which would increase the bandwidth requirement to unfeasible levels. Nowatkowski et al. [19] have shown that the CRL list may grow as much as 2.2 GB, depending on the policy for the number of pseudonyms on vehicles.

There are several approaches in the bibliography that try to address the problem with the size of CRLs. First, CCMS takes the approach not to revoke pseudonym certificates but instead revoke only the long-term identity of the vehicle [6] to prevent it from acquiring new ones. SCMS includes a linkage value to pseudonym certificates derived from cryptographic seed material [27]. Publication of the seed is sufficient to revoke all certificates belonging to the revoked vehicle. However, this requires the addition of two new entities in the architecture, called Linkage Authorities (LAs), with corresponding technical and organizational guarantees to operate separately. Alternative solutions leverage encrypted pseudonyms during the provisioning process [18, 21] so that a vehicle can only decrypt pseudonyms after receiving the encryption keys. In addition to the efficiency problems of the above revocation mechanisms, they all require the resolution of the vehicle's long-term identity from their pseudonyms, thus, posing a significant privacy threat.

2.1 Related Work

In order to address these shortcomings of PKI-based solutions, there is an increasing effort by researchers to apply anonymous credentials as a solution for privacy-respecting V2X communication. For example, Foster et al. presented PUCA [8] a pseudonym scheme that allows vehicles to use anonymous credentials for authentication with the PCA when obtaining new pseudonyms in the existing European ITS system, changing only the pseudonym issuance phase. PUCA foresees no way of credential revocation. The REWIRE V2X revocation protocol [7] uses trusted computing to enable revocation without pseudonym resolution. An enhanced variant was presented in O-TOKEN [25] where an additional key pair is embedded into pseudonym certificates. However, in both schemes, there are inherent trust assumptions been made on the *correctness* of each vehicle Electronic Unit (ECU) that limits their feasibility and applicability in real-world environments. More specifically, they have not considered an enhanced threat model where malicious and/or compromised ECUs can monitor and modify all interactions between the host and the attached Trusted Component (TC); i.e., using the TC as an "*oracle*" that can interact with for executing sensitive crypto operations in order to bypass the revocation.

Whitefield et al. [24] first applied DAA to the V2X case and showed how to enable vehicles to manage their own pseudonym certificates, however revocation was left open. Hicks et al. [14] proposed a scheme that leverages the decentralized properties of DAA towards enabling a secure and privacy-preserving revocation coupled with strong vehicle authentication. However, even though they use DAA, they don't provide a fully decentralised architecture. Their solution still depends heavily on newly introduced centralized entities (on top of the existing RA), such as an Enrollment Authority and an Authorization Authority. This effectively makes their scheme susceptible to malicious insiders who can get enough information and break the anonymity behind the pseudonyms and also link multiple pseudonyms together back to the same entity. Finally, Kumar et al. [17] proposed the use of DAA for verifier-local revocation, but their scheme is rather inefficient, because they still use revocation lists and the computational and communication overhead is linear to the size of the list.

2.2 Overview of Revocation using DAA

Figure 1 introduces the typical DAA [2] pseudonym life-cycle architecture [24]. As we can see, only two trusted third parties are needed; (i) the Issuer who is responsible for authenticating vehicles

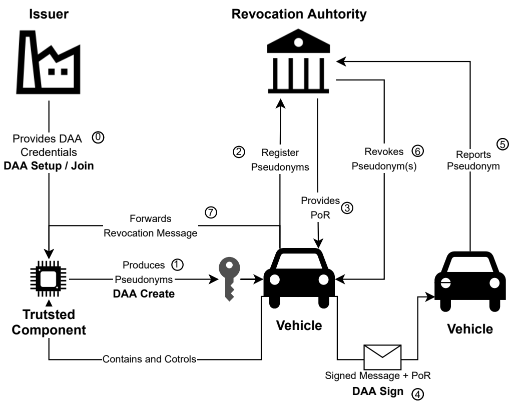


Figure 1: Conceptual Overview

through the JOIN protocol (Step 0) and (ii) the Revocation Authority (RA) that shuns out misbehaving vehicles from the ITS within the revocation domain that is managing. In our context, vehicles are the combination of a *host*, that is a vehicular onboard computer, and a *trusted computing component* (TC) that executes in the "secure world"; together, they form the platform which we refer to from this point onwards as the vehicle.

Using DAA, the trusted computing component (TC) in the vehicle is responsible for creating the pseudonym certificates without involving any infrastructure component from the back-end (Step 1 - DAA CREATE). Only the Issuer knows the identity of a vehicle. During the DAA SETUP and JOIN phases, the Issuer verifies that the TPM is valid and provides credentials that can be later used for creating either linked or unlinked pseudonyms; this is decided by the vehicle itself (DAA CREATE phase). Unlinkable pseudonyms enable the provision of *unconditional anonymity*, a property that is not provided by other proposed Vehicular DAA protocols [14]. The credentials do not contain any personal identifying information. The signing key of the TPM is not linked to the vehicle, and it is certified blindly by the Issuer. Revealing to whom the pseudonym corresponds to is infeasible because the identity of the TC (and hence that of the vehicle) is not linked to its signing key.

The vehicle cannot use pseudonyms unless it has been registered with the RA (Step 2 - DAA JOIN). The registration consists of providing the RA with unique values that can be used later to revoke the key (DAA SETUP). We call these values *revocation hashes*. Upon such a registration, the vehicle will receive Proof of Registration (PoR) (Step 3), which is sent along with a signed message (Step 4). Without PoR, any vehicle should disregard the message, as the RA would not be able to revoke such a key. In case a vehicle suspects malicious behavior of a vehicle, it reports the corresponding pseudonym to the RA (Step 5). Let's assume a number of reports containing a misbehaving vehicle's pseudonym have been already issued to the RA, and the decision to revoke the vehicle has been made based on strong evidence.

The RA does not perform any pseudonym resolution to discover the identity of the misbehaving vehicle. Instead, it initiates the revocation protocol by creating a signed revocation message using its secret key and broadcasts this to all vehicles containing the public pseudonym key that needs to be revoked (Step 6). All vehicles receive the revocation message, and their hosts are required to forward them to their corresponding TCs (Step 7). It is the TC of the revoked vehicle which is responsible for deleting the pseudonyms.

To be more precise, we differentiate between two kinds of revocations: soft- and hard-revocation. Soft revocation means the RA revokes a specific pseudonym that was used for signing the message based on which the misbehavior policy violation was detected, while hard revocation means revoking all pseudonyms associated with a specific vehicle, thus, not allowing it to further participate in the overall system as an authenticated participant. These two revocation variants essentially reflect the current need for *message-based* and *identity-based* revocation [14]: The first scenario might be triggered when revocation needs to occur due to a technical defect of a vehicle; i.e., malfunctioning sensor (thus, we want to temporarily revoke his ability to participate in the system by not allowing it to re-use this specific pseudonym). The second case mainly deals with malicious attackers who need to be banned from any subsequent communication as soon as possible.

Table 1: Notation used

Symbol	Description
$EK^{\dagger}$	Endorsement Key
$AK^{\dagger}$	Authorization Key
$AK_{old}{}^{\dagger}$	Authorization Key from previous iteration
$RAK^{\dagger}$	Revocation Authority Key
$EpK\dagger$	Ephemeral Key
KH_x	Key Handle identifying a loaded key (x) in the Tc.
$E_{\mathtt{TMP}}, A_{\mathtt{TMP}}$	Key Creation Template EpK and AK respectfully
P_d	A digest representing a governing policy
r_{index}	A non-volatile index in Tc containing 64 bits.
ACI	A non-volatile index in Tc containing 8 bits.
r_{bit}	Revocation bit(s).
$A_{\mathtt{LIMIT}}$	Max number of correct authorizations for ACI
$A_{\mathtt{COUNT}}$	Current number of correct authorizations for ACI
ACI_{old}	An instance if ACI from previous iteration
σ_x	Cryptographic signature
H	A hash output (digest)
λ	A set of hard revocation and soft revocation policy (leaf digests, l)
β	A compound policy (of branch digests, b) to satisfy to allow revocation.
$\mathcal{P}$	A set of β plus initial write policy.
$P_{\mathtt{Auth}}$	Final policy digest (P_d) of all data in $\mathcal{P}$ signed by AK
t	A ticket generated by Tc proving verification of a signature.

† An asymmetric keypair, containing both public and a private key, denoted xK_{pub} and xK_{priv}.

3 CONCEPTUAL PROTOCOL OVERVIEW

In this section, we make a high-level overview of our protocol, showcasing how we can implement policy regulations for governing the pseudonyms using the functionality of the Trusted Platform Module (TPM) been used as the trusted component [22]. More specifically, we present how we leverage an internal, tamper-proof register of the TPM, where each bit represents the state of a pseudonym. Creating such a register, or index, in a way that remains tamper-proof for even the host requires deep analysis of the TPM and its internal functions. In the remainder of this paper, the symbols and abbreviations depicted in Table 1 are adopted.

3.1 Soft- and Hard-Revocation

We refer to the tamper-proof index of the TPM as *Revocation Index*. It has 64 bits, and each bit represents the state of a pseudonym, i.e., a set bit means revoked; otherwise, the pseudonym can be used for anonymous message signing (DAA SIGN). We consider two cases: Revocation of a single pseudonym, namely soft revocation, is rather

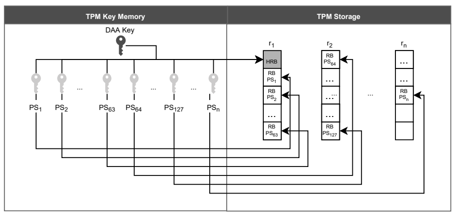


Figure 2: Pseudonyms in TPM, linked to different indexes that shares the same hard revocation bit

straightforward. As we trust the TC managing the keys, it will be asked to set the key's respective revocation bit to a "revoked state". On the other hand, hard revocation refers to the possibility of being able to use one of the 64 bits to revoke all pseudonyms in a single round. This means that the DAA key should be linked to the first bit of the Revocation Index, while the pseudonyms are linked to one of the other 63 bits, as well as the first (hard revocation) bit.

We must accommodate the possibility that different authorities govern different areas of the vehicular network, i.e., an RA in one domain should not be allowed to revoke pseudonyms linked to another domain. Therefore, we must protect each bit in the revocation index, allowing only predetermined RAs to execute a revocation process.This can be achieved by building policies for each pseudonym, representing the command being executed (set bit) with particular parameters (which bit). A particular RA must sign these to authorize a revocation. We refer to them as *revocation hashes* and each pseudonym is linked to the following: one for soft revocation and one for hard revocation. These revocation hashes are registered with the RA, who must sign them before using them in the revocation process. It should now be clear that the hard revocation hash must include both the hard-revocation bit and the pseudonyms' unique soft-revocation bit. If this is not the case, the hard-revocation hash for all linked pseudonyms would be equal, as they are to be signed by the same RA and setting the same bit. By including the soft-revocation bit, we "blind" the revocation hash.

Pseudonym limitations. Since the size of the Revocation Index is 64 bits and given we need one for the hard revocation, we can revoke only 63 pseudonyms with this index, which is not enough to cover the requirements of vehicular applications. A naive approach to support more pseudonyms would be to create more revocation indexes and having a new DAA key for each one, following the same principle. However, this poses two distinct problems: first, the DAA Setup and Join phases are very time-consuming and require communication with the Issuer. Secondly, only 63 pseudonyms can be linked together in the TPM, making hard revocation of a larger set impossible. Therefore, we create multiple revocation indexes using all of their bits for soft revocation and we maintain only one hard revocation bit to be the one defined in the initial revocation index. So all pseudonyms share the same hard revocation bit, meaning hard revocation hashes would be equal for all pseudonyms having soft revocation bits in other indexes. An example of this is shown in Figure 2, where all pseudonyms linked to the DAA key share a common hard revocation bit (first bit of r_1), while their corresponding soft revocation bits span several revocation indexes.

With the current implementation of the TPM, it is not possible to set multiple bits in different indexes, which means they would have to be executed as two commands, removing the "blinding" of the hard revocation hash. To combat this challenge, we propose that the hard-revocation hash represents a command that sets the hard revocation bit and any other unique combination of bits in the index, allowing for 2^{63} pseudonyms with a unique hard-revocation index. This requires that the revocation index's bits are revocable only by a single RA; otherwise, an anonymizing mask could cause the unintentional revocation of keys linked to another revocation domain. To support multiple RA domains in a single index, the number of linked pseudonyms is limited to 2^n where n represents the available bit space for each pseudonym set. For instance, with a single index and considering a single RA the limit is 2^{63}, with two RAs it is 2^{31}, and so forth. However, it is possible to use different indexes if more are needed. As the TPM is limited in its internal storage to a minimum of 1600 bytes (for automotive), out of which 12800 bits can be used for managing pseudonyms, considerations should be made to reduce the number of indexes used.

3.2 Building the Protocol

It should now be evident that we protect the revocation index with a set of policies. This, however, raises an interesting challenge: When we write a policy that determines what parameters must be used, a so-called *command parameter hash* must include the name[1] of the entity it applies to - in this case, the revocation index. As the policy is intended *for* the index, it is included in its public part. The index's name depends on the policy, and the policy depends on the index: *we, therefore, have an infinite hash loop.*

To avoid this hash loop, we use a different approach[2] where a unique key, referred to as *Authorization Key* (AK), authorizes the policy (Figure 3). The process is that *any* policy signed by the AK is considered a valid policy. The revocation index's actual policy digest is the name of the authorizing key, and in order to satisfy that policy, one must have the policy signed by the AK. The index and revocation policies, therefore, are no longer coupled together. However, this approach raises another obstacle: *we must guarantee that the AK can only sign a single policy.* If not, the host can sign any policy to comply with and control the revocation index.

To address this challenge, we protect the AK by yet another policy. This policy must dictate how many times the host can use the key. We do this by creating an additional index, called the *Authorization Counter Index* (ACI), and protecting it by a PIN or password as an authorization value. The ACI contains two parts: Authorization Counter and Authorization Limit. We use this index to create the authorization key policy by leveraging `PolicySecret` functionality. This policy states that we must prove our knowledge of the ACI's secret by authenticating with it. Every time the password is used for this index, the internal counter increments. When the counter reaches the pre-determined limit, it fails. With this index, we can define the AK policy as the correct authentication with the ACI, meaning that we must perform (password) "proof-of-knowledge" for that index, therefore, incrementing the counter and limiting the use of the key to the authorization limit of 3 + 1, where the first three authorization processes are for: i) writing the authorization limit, ii) authorizing the final policy digest, and iii) activating the revocation index. The last authorization is required for the AK to

[1]Name is a hash of the public parameters of an entity, including its policy.
[2]This was identified as a solution in collaboration with the TPM WG of TCG.

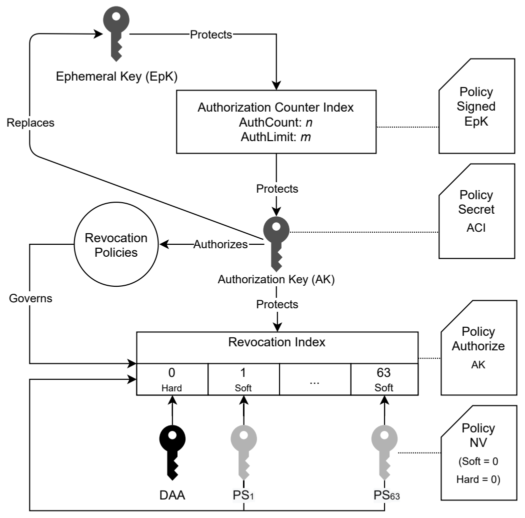


Figure 3: Protocol Functionality and Lifecycle

be used once as the Ephemeral Key in order to make the next ACI immutable. Now the host can still re-create the ACI and, therefore, reset the counter and infer additional authorizations. To overcome this limitation, we protect the update of the ACI by setting a policy that requires a signature from an Ephemeral Key - that is, a key that only exists during the initial setup, therefore, making the ACI immutable. A straightforward way is to create the ephemeral key in TC's NULL hierarchy. In this way, the host cannot recreate the key since the NULL hierarchy seed is reset on reboot.

Looking over this set of required tasks, it becomes cumbersome that the reboot is necessary for each revocation index created. It is neither efficient nor safe to reboot the Electronic Control Unit (ECU) after 64 pseudonyms have been used. One solution is to give the Authorization Key one additional authorization and use this as the ephemeral key for the next ACI. This has the same effect as a reboot and will render the ACI immutable after the initial write.

Following this approach, we can successfully initialize both the ACI and the revocation index. Before the host can start using the keys, it must initiate the revocation index by updating it; otherwise, it is unusable. In order to guarantee that this initial write-operation can only happen once, we will give the AK an additional authorization and sign a digest allowing the initial write.

4 ARCHITECTURAL DETAILS & PROTOCOLS

4.1 Authorization Counter Index

The initial phase of creating the Primordial Authorization Counter Index can be seen in Figure 9 in Appendix A.3. As can be seen, the host must provide an index identifier, template, and authorization limit. As a potentially untrusted host can process such information, this underlines why a trusted entity should verify the index in order to further weaken the trust assumptions regarding the trustworthiness of the host (Appendix A.2). The first thing to be executed is the creation of the Ephemeral Key in the `NULL` hierarchy. This can be verified by a certification process that documents the key and TC's validity as well as the current boot count.

To create the index, the host creates a policy based on the Ephemeral Key and instructs the TC to define the space with a random secret. The secret is essentially a password; however, the endmost goal is to use it as a usage counter. Hence there is no need to keep it secret.

The policy defined earlier must be complied with to write the authorization limit to the index, so the host uses the Ephemeral Key to sign a nonce. After complying with the policy, writing the authorization count, and inherently incrementing the authorization count, the key should be rendered inoperable by executing a reset. After the reset has been completed, the index can be certified by the endorsement key. Such a certificate can be used to prove that the index has a specific authorization limit, and the current boot count proves the reset has been executed. After the reboot, the ACI is guaranteed immutable and is prepared to act as a guard for limiting the number of times the upcoming authorization key can be used.

4.2 Revocation Index

Before initializing the revocation index, we need to create the AK and link it to the ACI. As depicted in the first line of Figure 7, in Appendix A.3, the host builds a `PolicySecret` policy based on the ACI name: In order for the host to use the key, it needs to provide the secret for the ACI, thus, resulting in the increment of the authorization counter. This policy is embedded in the template for the key, and in the future, only this template with this specific policy will allow the correct AK recreation.

As with other entities, the created key can be certified and signed by the Endorsement Key for later verification. To create the index, the host calculates a new policy digest that links the index's usage to the AK by leveraging the `PolicyAuthorize` functionality - meaning that any policy signed by the authorization key is valid. The index is now built within the trusted component, and its policy depends on what the authorization key signs in the next phase.

4.3 Generate Final Policy Digest

The policy digest to be authorized by the AK is calculated following the steps described in Algorithm 1. Recall that the policy is a compound policy built by logical `AND` and `OR` statements. We can visualize this logical composition, as depicted in Figure 8, Appendix A.3, where we can see that two policies must be true in order to produce l_1 or l_2. The first would be setting the soft revocation bit while the latter for updating the hard revocation bits. Either of these will satisfy the `OR` operation, producing b_1, which in turn is an input to a final `OR` operation. More specifically, if the RA provided an authentic signature with a parameter either a hard- or soft-revocation hash (that is unique to a key), this is a valid branch for a single pseudonym, and its revocation will take place. Each branch digest b represents a valid soft- or hard revocation for a single pseudonym. An `OR` operation may take up to 8 input parameters; hence, it might be necessary to have multiple layers of these operators.

It is possible to calculate these by executing the commands in a trial session of the TC, but we showcase this by calculating it on the host. From an implementation standpoint, we start by initializing our variables and continue to define our very first branch digest: the activation. Recall that we have to update the index before it can be used. To ensure this can only be performed once, we leverage the AK's use-limit property and allow an initial write to the index. We then continue into a loop where we iterate over all revocable pseudonyms: For each of the keys, we create two leaf digests, l_1 and l_2. We also initiate S_{data} and H_{data}, representing the parameters used in the SetBits command: the bits being set. We increment the anonymizer and set the H_{data}, thereby ensuring the uniqueness of the parameter hash. After having anonymized the data, we can set the respective hard- and soft revocation bits and continue calculating the respective revocation hashes. We see $k.sri.name$, the pseudonym k's soft revocation index's name, and hri representing the hard revocation index.

With this data, we can calculate the soft- and hard revocation leaf digests, l_1, l_2, and then calculate the branch digest b. This is inserted into β, the list of all branch digests, and then we loop again. When all b's are inserted into β, we can calculate the final digest to be authorized during the activation of the revocation index.

4.4 Activate Revocation Index

Before we can use the revocation index, we must update it, which in our case is setting it to zero. Recall that the policy is built to allow a write to the index if the AK provides a signature over the command to execute. The policy is not authorized yet, so the first task that the host needs to perform is to prepare the values to be hashed and signed: hashing the final policy and generating the command parameter hash for the initial write. It then continues to gain access to the AK by providing the ACI's secret, inherently incrementing the authorization count.

Now the host can get a signature over the policy and acquire a verification ticket: A ticket guarantees that it is this particular TC that has verified the signature, therefore, guaranteeing the command and its parameters have been correctly authorized.

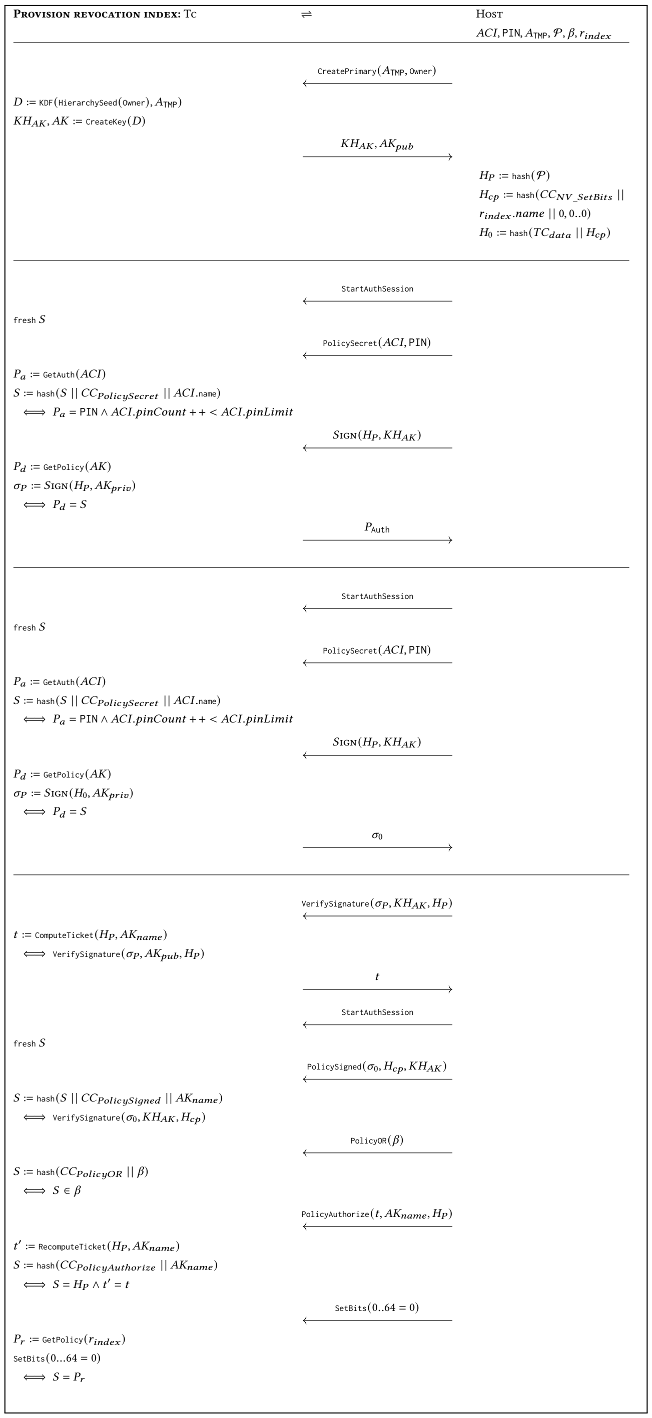
To gain (writing) authorization privileges, the host initiates a new session and executes one of the index's valid policies, namely the PolicySigned with the previously acquired signature over the zero-write command parameter hash. The current session digest should now match the branch digest b_0, and the host executes PolicyOR with β. After a successful verification from the TC, it will replace the session digest with a concatenation of all provided branch digests in β, which should be the index's authorized policy. The host executes PolicyAuthorize with the previously acquired ticket and signature. The TC then verifies the ticket, the signature, and finally, the session digest (whether it matches the authorized digest). If this is the case, the TC replaces the session digest with the AK's name: the policy digest for the index. The host can now execute a write operation, and the index has been activated.

While it is now possible for the host to update the index, a possibly compromised host does not have any incentive to write anything but zeroes, as it would set pseudonyms in an initially revoked state. At the end of this phase, the index has been correctly activated and set. Furthermore, the final policy calculated in the previous step has also been authorized. It is now ready to be used for managing the revocation states of pseudonyms. As with the primordial ACI, the index has been initialized with an authorization limit and counter and is immutable. The index can also act as a protector for regulating a new AK.

4.5 Initialize New Authorization Counter Index

To initiate a new ACI, we must guarantee immutability by using an old AK, as it will only have a single authorization left, based on the previously defined ACI. Thus, we start by creating a policy digest based on PolicySigned and the name of the old AK. We then define

Figure 4: Activate Revocation Index


PROVISION REVOCATION INDEX: TC
$\rightleftharpoons$
HOST
$ACI, \mathsf{PIN}, A_{\mathsf{TMP}}, \mathcal{P}, \beta, r_{index}$
CreatePrimary$(A_{\mathsf{TMP}}, \mathsf{Owner})$
$D :=$ KDF(HierarchySeed(Owner)$, A_{\mathsf{TMP}})$
$KH_{AK}, AK :=$ CreateKey(D)
KH_{AK}, AK_{pub}
$H_P :=$ hash$(\mathcal{P})$
$H_{cp} :=$ hash$(CC_{NV_SetBits} \,||\, r_{index}.name \,||\, 0, 0..0)$
$H_0 :=$ hash$(TC_{data} \,||\, H_{cp})$
StartAuthSession
fresh S
PolicySecret(ACI, PIN)
$P_a :=$ GetAuth(ACI)
$S :=$ hash$(S \,||\, CC_{PolicySecret} \,||\, ACI.$name$)$
$\iff P_a = \mathsf{PIN} \wedge ACI.pinCount{+}{+} < ACI.pinLimit$
$\textsc{Sign}(H_P, KH_{AK})$
$P_d :=$ GetPolicy(AK)
$\sigma_P := \textsc{Sign}(H_P, AK_{priv})$
$\iff P_d = S$
P_{Auth}
StartAuthSession
fresh S
PolicySecret(ACI, PIN)
$P_a :=$ GetAuth(ACI)
$S :=$ hash$(S \,||\, CC_{PolicySecret} \,||\, ACI.$name$)$
$\iff P_a = \mathsf{PIN} \wedge ACI.pinCount{+}{+} < ACI.pinLimit$
$\textsc{Sign}(H_P, KH_{AK})$
$P_d :=$ GetPolicy(AK)
$\sigma_P := \textsc{Sign}(H_0, AK_{priv})$
$\iff P_d = S$
σ_0
VerifySignature(σ_P, KH_{AK}, H_P)
$t :=$ ComputeTicket(H_P, AK_{name})
$\iff$ VerifySignature$(\sigma_P, AK_{pub}, H_P)$
t
StartAuthSession
fresh S
PolicySigned$(\sigma_0, H_{cp}, KH_{AK})$
$S :=$ hash$(S \,||\, CC_{PolicySigned} \,||\, AK_{name})$
$\iff$ VerifySignature$(\sigma_0, KH_{AK}, H_{cp})$
PolicyOR(β)
$S :=$ hash$(CC_{PolicyOR} \,||\, \beta)$
$\iff S \in \beta$
PolicyAuthorize(t, AK_{name}, H_P)
$t' :=$ RecomputeTicket(H_P, AK_{name})
$S :=$ hash$(CC_{PolicyAuthorize} \,||\, AK_{name})$
$\iff S = H_P \wedge t' = t$
SetBits$(0..64 = 0)$
$P_r :=$ GetPolicy(r_{index})
SetBits$(0...64 = 0)$
$\iff S = P_r$

a bit-space with that policy. Since we have a single authorization left in our AK, we can now create this new key. Recall that in order to use this key, we must execute PolicySecret and provide the PIN to the previous ACI. Thus, we need to provide a signature, based on the old key, and execute the PolicySigned command.

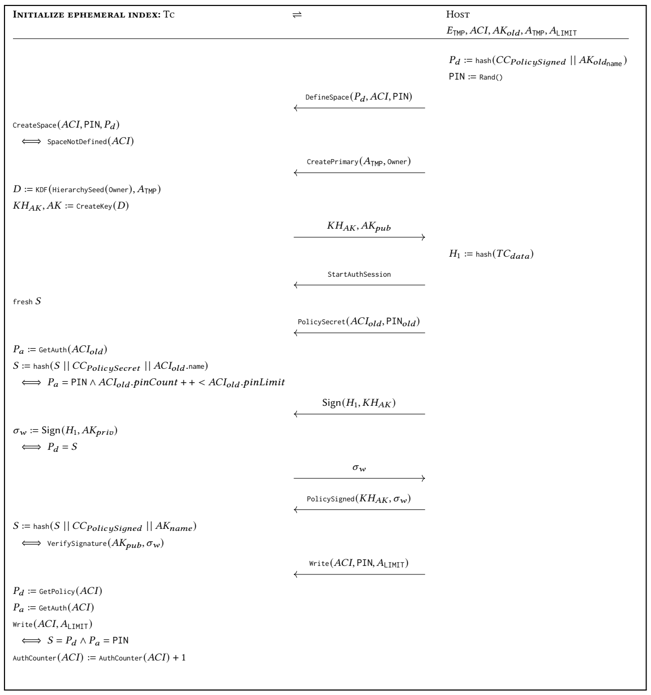

INITIALIZE EPHEMERAL INDEX: TC
⇌
HOST
$E_{\mathsf{TMP}}, ACI, AK_{old}, A_{\mathsf{TMP}}, A_{\mathsf{LIMIT}}$
$P_d :=$ hash$(CC_{PolicySigned} \,||\, AK_{old_{\mathsf{name}}})$
PIN := Rand()
DefineSpace(P_d, ACI, PIN)
CreateSpace(ACI, PIN, P_d)
$\iff$ SpaceNotDefined(ACI)
CreatePrimary$(A_{\mathsf{TMP}}, \mathsf{Owner})$
$D :=$ KDF(HierarchySeed(Owner)$, A_{\mathsf{TMP}})$
$KH_{AK}, AK :=$ CreateKey(D)
KH_{AK}, AK_{pub}
$H_1 :=$ hash(TC_{data})
StartAuthSession
fresh S
PolicySecret$(ACI_{old}, \mathsf{PIN}_{old})$
$P_a :=$ GetAuth(ACI_{old})
$S :=$ hash$(S \,||\, CC_{PolicySecret} \,||\, ACI_{old}.\mathsf{name})$
$\iff P_a = \mathsf{PIN} \wedge ACI_{old}.pinCount{+}{+} < ACI_{old}.pinLimit$
Sign(H_1, KH_{AK})
$\sigma_w :=$ Sign(H_1, AK_{priv})
$\iff P_d = S$
σ_w
PolicySigned(KH_{AK}, σ_w)
$S :=$ hash$(S \,||\, CC_{PolicySigned} \,||\, AK_{name})$
$\iff$ VerifySignature(AK_{pub}, σ_w)
Write$(ACI, \mathsf{PIN}, A_{\mathsf{LIMIT}})$
$P_d :=$ GetPolicy(ACI)
$P_a :=$ GetAuth(ACI)
Write$(ACI, A_{\mathsf{LIMIT}})$
$\iff S = P_d \wedge P_a = \mathsf{PIN}$
AuthCounter$(ACI) :=$ AuthCounter$(ACI) + 1$

Figure 5: Initialize New Authorization Counter Index

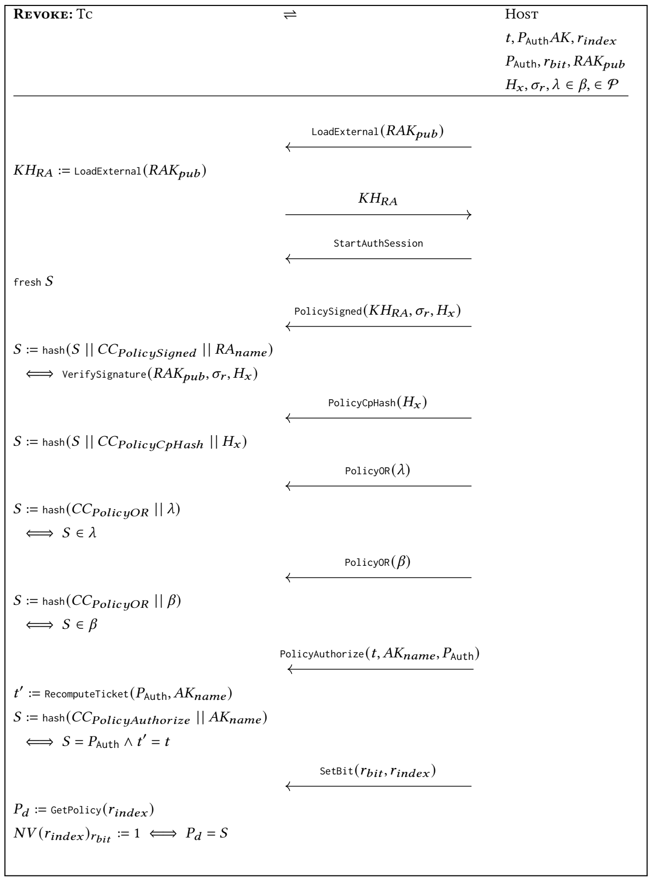

REVOKE: TC
⇌
HOST
$t, P_{\mathsf{Auth}} AK, r_{index}$
$P_{\mathsf{Auth}}, r_{bit}, RAK_{pub}$
$H_x, \sigma_r, \lambda \in \beta, \in \mathcal{P}$
LoadExternal(RAK_{pub})
$KH_{RA} :=$ LoadExternal(RAK_{pub})
KH_{RA}
StartAuthSession
fresh S
PolicySigned(KH_{RA}, σ_r, H_x)
$S :=$ hash$(S \,||\, CC_{PolicySigned} \,||\, RA_{name})$
$\iff$ VerifySignature$(RAK_{pub}, \sigma_r, H_x)$
PolicyCpHash(H_x)
$S :=$ hash$(S \,||\, CC_{PolicyCpHash} \,||\, H_x)$
PolicyOR(λ)
$S :=$ hash$(CC_{PolicyOR} \,||\, \lambda)$
$\iff S \in \lambda$
PolicyOR(β)
$S :=$ hash$(CC_{PolicyOR} \,||\, \beta)$
$\iff S \in \beta$
PolicyAuthorize$(t, AK_{name}, P_{\mathsf{Auth}})$
$t' :=$ RecomputeTicket$(P_{\mathsf{Auth}}, AK_{name})$
$S :=$ hash$(CC_{PolicyAuthorize} \,||\, AK_{name})$
$\iff S = P_{\mathsf{Auth}} \wedge t' = t$
SetBit(r_{bit}, r_{index})
$P_d :=$ GetPolicy(r_{index})
$NV(r_{index})_{r_{bit}} := 1 \iff P_d = S$

Figure 6: Revoking Vehicle's Pseudonyms & DAA Key Pairs

4.6 Revocation

Upon receiving a (potential) revocation message, the host loads the corresponding RA's public key into the TC. The received message contains the public pseudonym key, pk_{ps}, which needs to be revoked, and a revocation hash and signature; i.e., of the respective revocation index, r_{index} and (hard or soft) revocation bit, r_{bit}. Recall that the policy for gaining write-access to the revocation index includes both `PolicySigned` (verifying that the message originates from the correct RA responsible for the trust domain where the vehicle also belongs to) and `PolicyCpHash` (correct revocation bit(s)): both these policies must be correctly satisfied before the TC allows the successful revocation. Executing these processes should produce a valid leaf digest (the result of the hashing done in the TC after `PolicyCpHash` is executed, also noted as l_x), validated by executing `PolicyOR` with a reference list of the possible leaf digest (λ) for that specific branch. Suppose the leaf digest matches a digest in the reference list. In that case, the session digest is replaced by a hash of the reference list: the branch digest, verified by an additional `PolicyOR`. We depict the execution of these policy commands in Figure 6 where it is highlighted that `PolicyOR` will replace the current session digest with a hashed concatenation of all provided reference branch digests (β) *if and only if* the current session digest is in the provided reference. If the reference list is unaltered, the hashed concatenation should be the authorized policy, which is verified by executing `PolicyAuthorize` with an authorization ticket, the name of the authorizing key, and, of course, the authorized policy. If the session digest matches the approved policy, then it is replaced by the hash of the command code of `PolicyAuthorize` and the authorizing key's name, which is the policy for the revocation index. The host can now execute a write to the index, but only with the parameters used in the `PolicyCpHash`, ensuring the correct bits are set; hence, *the right pseudonym(s) are revoked.*

Once all required pseudonyms, and their DAA key pairs, are deleted, the TC responds to the vehicle with a signed revocation confirmation σ_{rvk} which is then sent to the RA. Upon reception of the revocation confirmation, the RA verifies that this is signed by the same TC that issued the pseudonym certificate that was revoked, thus, implying that the correct vehicle has revoked itself. The entire signature can be verified using the DAA VERIFY. By the end of this protocol, there are strong guarantees that the vehicle in question has been revoked without the need for any pseudonym resolution. The RA has verifiable evidence from the vehicle that it has performed the revocation enforced by the TC.

We have to note here that in case an attacker intercepts this revocation message, or a malicious vehicle host blocks the revocation message intended for the TC (Section 5.1, then the revocation process will not be triggered, and the vehicle's TC will not respond back with a revocation confirmation (note that this is also an issue for REWIRE [7] and O-TOKEN [26]). In order for the revocation to take effect in this case, the TC needs to detect that this has occurred. This can be achieved by a *heartbeat* mechanism, such that the TC periodically expects either a revocation message or a heartbeat (which may be a revocation intended for some other TC, or else a timed message). Revocation messages and heartbeats include information about the period they are intended for; thus, a heartbeat for one period cannot be used at a different time. They are signed by the RA, so they cannot be tampered with or spoofed, and only one message is generated by the RA for each time period. Failure to receive a heartbeat message (or a series of messages so as to allow possible limited connectivity) can act as an indication for potential misbehavior that can also trigger revocation by the TC. In order to improve the safety level provided, this mechanism can make use of the types of heartbeat messages already provided for monitoring the status of one-hop vehicular topologies so as to produce indistinguishable communications and diminish the revocation vulnerability window existing in conventional CRLs [13].

5 SECURITY MODEL

In this section, we discuss the proposed DAA-based soft- and hard-revocation solution with respect to the achieved security and privacy properties. We consider the following roles within the scope of our analysis to be Vehicles, TCs, and RA.

5.1 Threat and Adversarial Model

Vehicular Communication systems are susceptible to both *outsider* and *insider* adversaries [25]. The former are unauthorized entities (i.e., no credentials or trust relationships with other system entities) that seek to compromise the system and disrupt its operation. In contrast, the goal of an insider attacker would be to intercept, block or modify network communications. More specifically, an internal adversary can have an already calculated attack strategy, prior to joining the system, aiming to disrupt the overall set of the aforementioned trusted computing and revocation services. Assuming that it is impractical to break the cryptographic protocols, the main attack vector would be to try and obtain a vehicle's DAA credentials in order to perform a malicious action and then ignore the revocation process; by trying to manipulate any of the setup phases of our protocol (Section 3.2) so that it can continue using any revoked pseudonyms. Adversaries are primarily vehicles or witnesses in the system. However, this does not exclude "*Honest-but-Curious*" (HBC) infrastructure entities who represent legitimate participants with an endmost goal of breaching a vehicle's privacy. The HBC does not deviate from the defined protocol rules but possibly learns information from legitimate message exchange and information monitoring. For instance, a malicious entity inside the RA could try to link the (revoked) pseudonyms back to the originating TPM and, thus, track the host vehicles.

5.2 Security Analysis

The security assurances rest both on the TCs within the vehicles and the correct calculation of the ACIs and revocation indexes (Appendix A.2 highlights how to weaken this later trust assumption by offloading the index verification process to the Issuer) to provide the security guarantees of the correct execution of the revocation process. We consider the following key properties.

Pseudonym Revocation Unlinkability: This property requires that no entity should able to link multiple pseudonyms, either when used for anonymous message signing or when deemed malicious and need to be revoked, back to the originating vehicle. In the setup phase of our protocol (Section 3.2), we can generate pseudonyms achieving this notion of *adaptable unlinkability*, meaning that it should be the vehicle that can adaptably control whether or not any two pseudonyms can be linked by a particular entity (i.e., RA). Our protocol guarantees that these properties hold throughout a pseudonym's lifecycle and that unlinkable pseudonyms retain that property during the revocation phase. Revocation in our protocol is solely based on hashes (Section 4.6). Since these contain at least one unique bit and an optional anonymization mask in the revocation index (Section 4.4), no other entity (be it a Vehicle or even the RA) can compare the two hashes to gain any advantage on the identity of the origin vehicle. Despite the additional (soft-) revocation variant that is not provided by any other existing DAA-based revocation scheme, our protocol supports adaptable unlinkability by allowing the establishment of multiple DAA key pairs, thus, enabling the provision of both absolute- and conditionally-unlinkable set of pseudonyms depending on the requirements of the envisioned use case (collision avoidance vs. infotainment). Existing schemes [14] either use signature-based DAA revocation, which always links pseudonym values to signatures in order to retain centralised revocation capabilities, or still use Certificate Revocation Lists [17] where for each revoked pseudonym, a (trusted) verifying entity needs to generate a proof of non-revocation, thus, making strong assumptions on the Verifier's correctness.

Revocation Assurance: A key requirement of the revocation mechanism is to provide strong guarantees that when an RA has initiated and run the protocol to completion, the revoked vehicle will not be able to provide valid signatures to any other entity. When the revocation process is executed, as defined in Section 4, pseudonyms deemed as revoked are guaranteed to be inoperable. As the pseudonyms can only be decrypted and used inside the TPM, the signing key's usage policy needs to be satisfied and verified by the TPM - which by definition is trusted. The revocation process ensures that these policies can be satisfied *if and only if* all policy digests have been calculated correctly (Section 4.3), and under the minimal trust assumptions (Section 3.2), it is impossible to reverse the process. By including the Issuer in the pseudonym creation phase, these trust assumptions could be further weakened, as described in Appendix A.2. In existing DAA-based revocation schemes, the DAA key is revoked, rendering all pseudonyms inoperable. The host must therefore contact the Issuer and reobtain a new DAA key and create new pseudonyms. They, therefore, suffer from significant computational and communications costs that increase proportionally to the number of revoked/created pseudonyms. This drawback renders such schemes impractical when the number of pseudonyms grows beyond a relatively modest size. In contrast, with our protocol, we can revoke individual or subsets of pseudonyms without rendering the DAA key revoked, allowing for a more flexible and efficient revocation, thus, reducing the dependencies on the Issuer.

Coerced Revocation An insider attacker can also exhibit a more random attack behavior aiming to *evade* the revocation process by either an intentional misbehavior, so as to hide the existence of another "stealthy" compromised vehicle, or posing as multiple vehicles (acting as a Sybil entity [5]). In practice, this requires an adversary using another vehicle's pseudonyms by either trying to re-create them or migrate them to its own TPM. Recall that all pseudonyms are children keys of (encrypted by) the DAA key which in turn is a child key of the vehicle's TPM Endorsement Key. Even if two adversaries were to share pseudonyms, they cannot load (and decrypt) such key hierarchies that have been created by another TPM. This assurance is provided by the TPM, as the trusted platform, and its internal key storage as well as the need to establish an authenticated session before accessing such key handles (only the original creating process has such privileges). This property when considered together with the fact that pseudonym signing keys cannot be used until they are activated, thus, controlling the number of stored pseudonyms that are simultaneously valid at any point in time, renders our protocol (in contrast to the other DAA-based schemes [14, 17]) Sybil-secure.

6 PERFORMANCE EVALUATION

In this section, we analytically evaluate the computational complexity and overhead posed by our protocol. Our solution avoids several of the shortcomings of the revocation solutions in PKI systems [7, 11, 20] and namely: a) it does not require pseudonym resolution in order to trace a pseudonym back to the misbehaving vehicle's long-term identifier, b) it does not require the use and distribution of CRLs, and c) the vehicles do not need to periodically connect to the RA for "pulling" the latest version of the revocation blacklist. So, our solution minimizes bandwidth and connectivity requirements since it is based on the broadcast of a short revocation message containing the pseudonym of the misbehaving vehicle.

We focus the experimental evaluation on the computational complexity and analyze the timing of the core phases, as described in Section 4. We divide the operations into two classes - (1) offline and (2) online. All the operations which can be either pre-computed or not need to be executed in real-time are classified as offline operations (Section 4.1 - 4.5). The operations which need to be performed in real-time are classified as online operations. These include the computations at the TPM and the vehicle host for performing the actual hard- and/or soft-revocation task (Section 4.6). The endmost goal is to determine how computationally expensive each of these sets of offline and online operations is and analyze their impact on the overall feasibility of the protocol. Of course, we are particularly interested in the critical online operations that need to be executed more frequently and with strict time execution constraints.

6.1 Implementation Results

Evaluation Environment Setup & Testing Methods: Implementation was straightforward once the protocols were designed and written in terms of the appropriate TPM calls (see Tables 2 to 5), which also constitutes one of the novelties of this work since, to the best of our knowledge, it is one of the first complete instantiations of such a strong and provable revocation mechanism. The protocols were implemented in C/C++ (Appendix A.1) and the IBM implementation of a TPM software stack (IBM TSS v. 1.6.0) [16].

The experiments' goal is to verify that the proposed solution is functional and outline the phases needed to support it, and most importantly, verify that near-constant revocation time is achievable. It is, therefore, implemented as a single binary with multiple entities since we opted out from considering the *network latency* induced by vehicular mobility, which usually implies volatile network connectivity. In all cases, our goal is to provide strong evidence on efficient revocation service provision and demonstrate its efficiency in comparison to other DAA-based approaches where the process execution time is linear to the size of the revocation list [17].

Table 2: Initialize Primordial Authorization Counter Index

Activity	Mean	± (95% CI)
Total Application Stack	578.20 ms	0.87 ms
Total TPM Stack	583.92 ms	0.67 ms
TPM2_CreatePrimary	259.69 ms	0.27 ms
TPM2_DefineSpace	30.04 ms	0.22 ms
TPM2_Sign	86.86 ms	0.20 ms
TPM2_StartAuthSession	18.22 ms	0.20 ms
TPM2_PolicySigned	132.57 ms	0.20 ms
TPM2_NV_Write	38.05 ms	0.30 ms
TPM2_FlushContext	9.14 ms	0.19 ms
TPM2_FlushContext	9.35 ms	0.21 ms

Table 3: Initialize Revocation Index

Activity	Mean	± (95% CI)
Total Application Stack	317.81 ms	0.53 ms
Total TPM Stack	312.88 ms	0.42 ms
TPM2_NV_ReadPublic	12.68 ms	0.19 ms
TPM2_CreatePrimary	260.62 ms	0.42 ms
TPM2_FlushContext	9.63 ms	0.19 ms
TPM2_DefineSpace	29.94 ms	0.23 ms

The results are generated using a laptop with Intel(R) Core(TM) i7-8665U CPU @ 1.90-2.11GHz. The protocols were then tested on two hardware-based trusted component platforms: an Infineon SLB9670 TPM [15] and a Nuvoton TPM so as to conduct a detailed investigation of the parameters that may affect the execution time of our revocation protocols. As can be seen in Section A.1, there is a strong interdependence of the optimal correctness of the results to the execution environmental setup and more specifically on the type of TC leveraged.

Performance Analysis: Although not the focus of the paper, we also opted to measure the actual timings for each one of the DAA Phases in order to provide a more comprehensive overview of the entire pseudonym lifecycle; from the creation to its secure and privacy-preserving revocation (when needed). The DAA JOIN and ISSUE protocols take up around 820 ms, while the creation and certification of a pseudonym key (DAA CREATE) take up 420ms. The DAA SIGN operation is relatively fast and requires 80ms. In contrast, the DAA VERIFY is split into two operations: the verification of the pseudonym key takes up 200ms, and the verification of the ECDSA signature takes up 10ms.

As aforementioned, the most performance-heavy operations of our protocol are the offline operations for initializing both the ACI and RI as well as registering the revocation hashes to the RA. Initializing the ACI, it's evident that the TPM is responsible for most of the incurred time overhead (Table 2). Indeed, the host's only operation is a simple hashing operation for the policy digest. Interestingly enough, the accumulated time for TPM execution is larger than the whole operation. This might be due to the multiple

Table 4: Activate Revocation Index

Activity	Mean	± (95% CI)
Total Application Stack	920.28 ms	1.02 ms
Total TPM Stack	931.99 ms	0.15 ms
TPM2_CreatePrimary	260.26 ms	0.27 ms
TPM2_StartAuthSession	17.78 ms	0.24 ms
TPM2_PolicySecret	25.01 ms	0.19 ms
TPM2_Sign	96.85 ms	0.30 ms
TPM2_FlushContext	9.23 ms	0.19 ms
TPM2_StartAuthSession	17.37 ms	0.25 ms
TPM2_PolicySecret	24.91 ms	0.19 ms
TPM2_Sign	97.28 ms	0.32 ms
TPM2_FlushContext	9.18 ms	0.19 ms
TPM2_VerifySignature	132.71 ms	0.21 ms
TPM2_StartAuthSession	17.94.67 ms	0.29 ms
TPM2_PolicySigned	132.82 ms	0.20 ms
TPM2_PolicyOR	9.89 ms	0.19 ms
TPM2_PolicyAuthorize	25.72 ms	0.21 ms
TPM2_FlushContext	9.32 ms	0.19 ms
TPM2_NV_SetBits	36.80 ms	0.26 ms
TPM2_FlushContext	8.90 ms	0.18 ms

Table 5: Revocation - Soft (S) and Hard (H)

Activity	Mean (S)	Mean (H)	± (95% CI)
Total App. Stack	327.71 ms	323.91 ms	0.93/0.98 ms
Total TPM Stack	349.66 ms	348.28 ms	0.71/0.65 ms
TPM2_LoadExternal	92.29 ms	92.31 ms	0.22/0.22 ms
TPM2_StartAuthSession	17.73 ms	17.56 ms	0.25/0.22 ms
TPM2_PolicySigned	133.62 ms	132.90 ms	0.22/0.20 ms
TPM2_PolicyCpHash	9.18 ms	8.90 ms	0.19/0.18 ms
TPM2_PolicyOR	9.51 ms	9.55 ms	0.19/0.19 ms
TPM2_PolicyOR	9.87 ms	9.75 ms	0.19/0.19 ms
TPM2_PolicyAuthorize	25.57 ms	25.66 ms	0.19/0.19 ms
TPM2_NV_SetBits	34.06 ms	33.31 ms	0.35/0.25 ms
TPM2_FlushContext	9.35 ms	9.20 ms	0.19/0.18 ms
TPM2_FlushContext	9.04 ms	9.14 ms	0.19/0.19 ms

initialization and deinitialization of the internal timing interfaces used. However, the combined time of creating the primordial ACI is rather efficient and applicable to environments where this is needed to be executed multiple times. If new ACIs are needed, this time will increase slightly, as the previous AK's policies must be satisfied. These include starting a new session and executing `PolicySigned` and can be estimated to add a timing overhead of 45ms.

Moving towards the initialization of the revocation index (Table 3), this requires even fewer resources. This is mainly due to the limited operations taking place: the only actual operation being executed is the creation of the revocation index itself. This is shown in the context of a "worst-case" scenario where the ACI has to be read, and the Authorization Key has to be recreated. The time required for the host operations is slightly higher since more hashing operations are needed for creating the policy digest.

Activating the revocation index is the heaviest phase of the protocol, as seen in Table 4. Obviously, the TPM is again consuming most of the required resources. As described in Section 4.4, the AK's policy has to be satisfied twice: first for signing (authorizing) the final digest and secondly for providing the signature for the initial write in the revocation bits. However, recall that this is an offline operation which means that there is no need for a vehicle to wait till the previously created bunch of pseudonyms runs out before creating and activating new pseudonyms and their RBs.

Finally, and more interestingly, we can see that both soft- and hard-revocation take an equal amount of time, as shown in Table 5. This is because the timing required is independent of the number of bits to be set; 1 for hard revocation or multiple in the case of a soft revocation, as it is essentially an OR operation of 64 bits.

Evaluation Analysis: From the above results, we observe that all online operations incur significantly lower overhead (in the order of $323, 91ms$ for the pseudonym revocation and $640ms$ for the pseudonym creation) - in terms of computation - compared to the existing DAA-based solutions that can take up seconds to conclude. For instance, LASER [17] adds an overhead of 22% to this process while VDAA [14] adds more than 50%. However, this comes at the cost of higher offline overhead ($2.5sec$). Nonetheless, this trade-off between offline and online computational costs is very advantageous because the online computations occur significantly more often than the offline ones. Also, as the online procedure requires significantly lower latency than the offline procedure, this renders our protocol more practical than existing schemes. Furthermore, as a triggering point for the commencement of the revocation process is the identification of a vehicles' malicious behavior based on a received CAM message which also entails the signature verification of this broadcast message. the two major V2X standards both use the ECDSA signature scheme [4]. Since our revocation protocol also assumes the use of regular ECDSA signatures (signature used in DAA) on broadcast messages, no additional overhead is incurred.

7 CONCLUSIONS

In this paper, we proposed a novel revocation scheme based on the use of trusted computing technologies and, more specifically, the Direct Anonymous Attestation (DAA) protocol. This secure and privacy-preserving scheme supports trustworthy vehicle-local verification, thus, overcoming the challenges of current solutions that have been proposed in the standards based on the use of traditional PKIs. We have shown that our protocol achieves a significant performance improvement over the prior state of the art by verifying that near-constant revocation time is achievable even when multiple pseudonyms are entailed. We have evaluated all of the internal protocol phases through a qualitative analysis as well as through an actual implementation on two TPM variants.

8 ACKNOWLEDGMENT

This work was partially supported by the European Commission, under the ASSURED project; Grant Agreement No. 952697.

REFERENCES

[1] 5GAA Automotive Association. Oct. 2020. Privacy-by-Design Aspects of C-V2X. White Paper. Available online at https://5gaa.org/wp-content/uploads/2020/11/5GAA_White-Paper_Privacy_by_Design_V2X.pdf.

[2] Ernie Brickell, Jan Camenisch, and Liqun Chen. 2004. Direct Anonymous Attestation. In *Proceedings of the 11th ACM CCS*. 132–145.

[3] Liu Chunli and Tang Li Fang. 2012. The Application Mode in Urban Transportation Management Based on Internet of Things. In *Proceedings of the 2nd International Conference on Electric Technology and Civil Engineering (ICETCE)*.

[4] The PRESERVE Consortium. 2011. Security Requirements of Vehicle Security Architecture. *Technical Report* (June 2011).

[5] John R. Douceur. 2002. The Sybil Attack. In *Peer-to-Peer Systems, First International Workshop, IPTPS*.

[6] European Commission. June 2018. Certificate Policy for Deployment and Operation of European Cooperative Intelligent Transport Systems (C-ITS).

[7] David Förster, Hans Löhr, Jan Zibuschka, and Frank Kargl. 2015. REWIRE – Revocation Without Resolution: A Privacy-Friendly Revocation Mechanism for Vehicular Ad-Hoc Networks. In *Trust and Trustworthy Computing*.

[8] D. Förster, F. Kargl, and H. Löhr. 2014. PUCA: A pseudonym scheme with user-controlled anonymity for vehicular ad-hoc networks (VANET). In *IEEE Vehicular Networking Conference (VNC)*. 25–32.

[9] Thanassis Giannetsos and Ioannis Krontiris. 2019. Securing V2X Communications for the Future: Can PKI Systems Offer the Answer?. In *Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES '19)*.

[10] Stylianos Gisdakis, Thanassis Giannetsos, and Panos Papadimitratos. 2014. SPPEAR: Security & Privacy-preserving Architecture for Participatory-sensing Applications. In *Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless &#38; Mobile Networks*. 39–50.

[11] Stylianos Gisdakis, Marcello Lagana, Thanassis Giannetsos, and Panos Papadimitratos. 2013. SEROSA: SERvice oriented security architecture for Vehicular Communications. In *VNC*. IEEE, 111–118.

[12] Philippe Golle and Kurt Partridge. 2009. On the Anonymity of Home/Work Location Pairs. In *Proceedings of the 7th International Conference on Pervasive Computing* (Nara, Japan) *(Pervasive '09)*. 390–397.

[13] J. J. Haas, Yih-Chun Hu, and K. P. Laberteaux. [n.d.]. Efficient Certificate Revocation List Organization and Distribution. *IEEE J.Sel. A. Commun.* 29, 3 ([n. d.]).

[14] C. Hicks and F. D. Garcia. 2020. A Vehicular DAA Scheme for Unlinkable ECDSA Pseudonyms in V2X. In *2020 IEEE European Symposium on Security and Privacy (EuroS P)*. 460–473. https://doi.org/10.1109/EuroSP48549.2020.00036

[15] Infineon Technologies AG. [n.d.]. Iridium SLB 9670 TPM2.0 Linux. https://www.infineon.com/cms/en/product/evaluation-boards/iridium9670-tpm2.0-linux// [Online; accessed 03-May-2019].

[16] International Business Machines. [n.d.]. IBM's TPM 2.0 TSS Version 1119. https://sourceforge.net/projects/ibmtpm20tss/ [Online; accessed 03-May-2019].

[17] Vireshwar Kumar, He Li, Noah Luther, Pranav Asokan, Jung-Min (Jerry) Park, Kaigui Bian, Martin B. H. Weiss, and Taieb Znati. [n.d.]. Direct Anonymous Attestation with Efficient Verifier-Local Revocation for Subscription System. In *Proceedings of the 2018 on AsiaCCS*. 567–574.

[18] Virendra Kumar, Jonathan Petit, and William Whyte. 2017. Binary Hash Tree Based Certificate Access Management for Connected Vehicles. In *Proceedings of WiSec '17*. 145–155.

[19] M. E. Nowatkowski, J. E. Wolfgang, C. McManus, and H. L. Owen. 2010. The effects of limited lifetime pseudonyms on certificate revocation list size in VANETS. In *Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon)*. 380–383.

[20] J. Petit, F. Schaub, M. Feiri, and F. Kargl. 2015. Pseudonym Schemes in Vehicular Networks: A Survey. *IEEE Communications Surveys Tutorials* 17, 1 (2015).

[21] Marcos A. Simplicio, Eduardo Lopes Cominetti, Harsh Kupwade Patil, Jefferson E. Ricardini, and Marcos Vinicius M. Silva. 2019. ACPC: Efficient revocation of pseudonym certificates using activation codes. *Ad Hoc Networks* 90 (2019).

[22] TCG. [n.d.]. *TPM 2.0 Library - Trusted Computing Group.* trustedcomputinggroup.org/resource/tpm-library-specification/

[23] Trusted Computing Group. [n.d.]. Trusted Computing Platform Alliance (TCPA) main specification. http://www.trustedcomputinggroup.org.

[24] J. Whitefield, L. Chen, T. Giannetsos, S. Schneider, and H. Treharne. 2017. Privacy-enhanced capabilities for VANETs using direct anonymous attestation. In *2017 IEEE Vehicular Networking Conference (VNC)*. 123–130.

[25] Jorden Whitefield, Liqun Chen, Frank Kargl, Andrew Paverd, Steve Schneider, Helen Treharne, and Stephan Wesemeyer. 2017. Formal Analysis of V2X Revocation Protocols. In *International Workshop on S&T*.

[26] J. Whitefield, L. Chen, F. Kargl, A. Paverd, S. Schneider, H. Treharne, and S. Wesemeyer. 2017. Formal Analysis of V2X Revocation Protocols. In *Security and Trust Management - 13th International Workshop, STM* (Oslo, Norway), Vol. 10547.

[27] W. Whyte, A. Weimerskirch, V. Kumar, and T. Hehn. [n.d.]. A security credential management system for V2V communications. In *(VNC'13*.

[28] Zhang Xiong, Hao Sheng, WenGe Rong, and Dave E. Cooper. 2012. Intelligent transportation systems for smart cities: a progress review. *Science China Information Sciences* (2012).

Table 6: Experiment with two Nuvoton TPMs

Command	NPCT650	NPCT750
TPM2_CreatePrimary	65.68 ms	45.50 ms
TPM2_StartAuthSession	5.04 ms	10.70 ms
TPM2_PolicySecret	4.29 ms	11.60 ms
TPM2_Sign	204.38 ms	27.30 ms
TPM2_VerifySignature	263.68 ms	53.90 ms
TPM2_FlushContext (Session)	3.16 ms	9.80 ms
TPM2_FlushContext (Key)	2.71 ms	11.10 ms

A APPENDIX

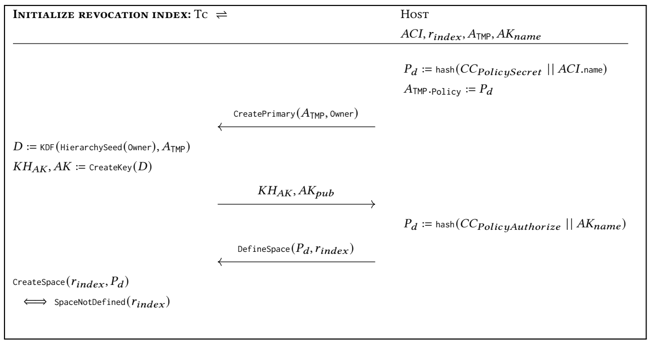
A.1 Experimentation Summary

In general, we see fast execution, and most of the work is done by the underlying trusted component. This implies that these timings are close to the most optimal, as optimizing the code will only have a non-essential impact. Furthermore, it is evident that re-creating the AK is a relatively heavy operation and should be kept in TPM volatile memory for as long as needed. Regarding a large number of pseudonyms, more `PolicyOR`'s are needed during revocation. This timing represents the first 6 pseudonyms, though just by adding two levels more (20ms), we can support $6 \cdot 8^2 = 384$ pseudonyms.

Interestingly, we noticed relatively large timings when it came to creating primary keys and verifying signatures. Therefore, an experiment was concluded in a different environment with two different TPMs. The following timings are acquired from a Dell desktop, x86, Ubuntu16.04 Xenial.

The results in Table 6 is an obvious example that both the environment and TPM will have an impact on the timings. In the latter example, we saw faster timings on creating keys but slower timings on signing and verifying signatures using an older Nuvoton TPM. The more modern Nuvoton TPM reduced the key operation's timings by a great deal, though more time is taken on non-key operations. This can be suspected due to Nuvoton potentially adding hardware support for ECC operations in their newer silicon. It is a prime example of timings being dependent on the physical implementation. These timings are extracted from the application layer, environmental factors such as operating system, workloads, and communication busses (I^2C, SPI, LPC) can impact timings. Despite the additional uncertainty revolving around timings in desktop environments, the revocation timings retain a small operation time. With only minimal added time when including an immense number of pseudonyms, it significantly improves traditional public key infrastructures, see section 8. Creating new pseudonyms dynamically is an operation that one must assume to happen often. This will include the three preparation phases: initializing a new ACI, initializing and activating a new revocation index. The new ACI is estimated to have only a slight increase in time, why the preparation to create new pseudonyms only takes in the order of around two seconds. As the host can execute these operations at any time, the host should do it before running out of pseudonyms, for example, when half of the existing pseudonyms have been used.

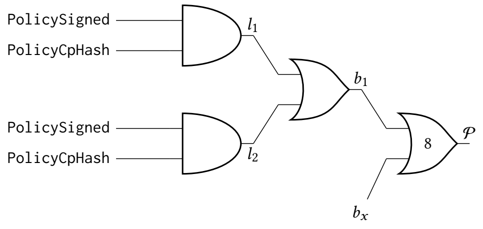
Figure 7: Initializing Revocation Index


Initialize revocation index: Tc ⇌
Host
$ACI, r_{index}, A_{TMP}, AK_{name}$
P_d := hash($CC_{PolicySecret}$ || ACI.name)
A_{TMP}.Policy := P_d
CreatePrimary(A_{TMP}, Owner)
D := KDF(HierarchySeed(Owner), A_{TMP})
KH_{AK}, AK := CreateKey(D)
KH_{AK}, AK_{pub}
P_d := hash($CC_{PolicyAuthorize}$ || AK_{name})
DefineSpace(P_d, r_{index})
CreateSpace(r_{index}, P_d)
⟺ SpaceNotDefined(r_{index})

A.2 Towards Near Zero-Trust Assumptions

Assuming near-zero trust assumptions for the vehicle requires additional validation of several processes that are executed in the host to protect against compromised vehicle hosts (e.g., ECUs) that try to manipulate the parameters given to the attached trusted component. This essentially considers a Dolev-Yao adversarial model, which allows an adversary to monitor and modify all interactions between the host and the TC. The critical operation to verify is the management of the policies generated on the host and that these have been correctly calculated for protecting the appropriate indexes and keys linked to active pseudonyms (an adversary can create a policy for inactive pseudonyms in which case a revocation message will be received and handled correctly but without any actual revocation results). The second operation to protect is the content of the ACI in order to validate the authorization limit. Finally, pseudonyms should be verified to be governed by the correct RAs whose revocation hashes have been calculated correctly. These proofs can be done using the TPMs certification functionality and validated by a trusted entity, such as the Issuer. However, to acquire a high level of trust, the trusted party should not immediately release the correct pseudonym certificates. Instead, it should verify that the AK has been used to replace an ephemeral key and use its last authorization to write to the next-round ACI. Once finalized, the host cannot misuse the final authorization, and the system can be trusted entirely.

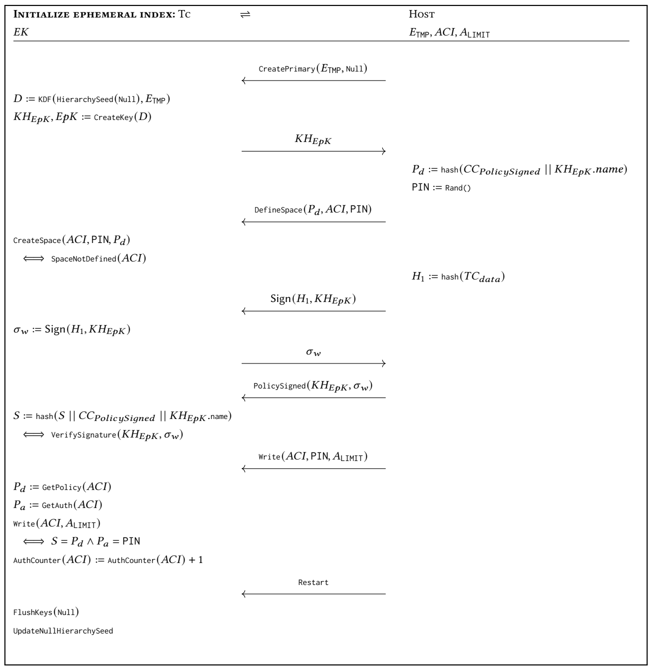
Figure 8: Structure of the policy to be authorized



A.3 Offline operation models

This section provides the break-down of the models and figures for the initialization and setup phases of our protocol, as described in Section 3.2. More specifically, Figures 7 and 9 depict the execution steps for the initialization of both revocation (Section 4.2) and authorization counter (Section 4.1) indexes that occurs right after

Figure 9: Initialize Primordial Authorization Counter Index


Initialize ephemeral index: Tc
⇌
Host
EK
E_{TMP}, ACI, A_{LIMIT}
CreatePrimary(E_{TMP}, Null)
D := KDF(HierarchySeed(Null), E_{TMP})
KH_{EpK}, EpK := CreateKey(D)
KH_{EpK}
P_d := hash($CC_{PolicySigned}$ || $KH_{EpK}.name$)
PIN := Rand()
DefineSpace(P_d, ACI, PIN)
CreateSpace(ACI, PIN, P_d)
⟺ SpaceNotDefined(ACI)
H_1 := hash(TC_{data})
Sign(H_1, KH_{EpK})
σ_w := Sign(H_1, KH_{EpK})
σ_w
PolicySigned(KH_{EpK}, σ_w)
S := hash(S || $CC_{PolicySigned}$ || KH_{EpK}.name)
⟺ VerifySignature(KH_{EpK}, σ_w)
Write(ACI, PIN, A_{LIMIT})
P_d := GetPolicy(ACI)
P_a := GetAuth(ACI)
Write(ACI, A_{LIMIT})
⟺ $S = P_d \wedge P_a =$ PIN
AuthCounter(ACI) := AuthCounter(ACI) + 1
Restart
FlushKeys(Null)
UpdateNullHierarchySeed

the pseudonym creation. Figure 8 visualizes the check of the policies that need to occur before the policy digest, to be authorized by the AK, is calculated (Section 4.3).

Algorithm 1 Calculate Final Policy Digest

```
(1) Initialize $\mathcal{P}$ as an array of hashes (capable of holding $n$ hashes where $n$ is the number of pseudonyms) and anonymizer as clear byte. Calculate activation branch digest $b_0 := H(H(b_0 || CC_{PolicySigned} || AK_{name}))$ and add to $\beta$
(2) For $k \in \mathcal{K}$:
    (a) Initialize $l_1, l_2$ as a two digest buffers. Initialize $S_{data}$ and $H_{data}$ as 8 byte buffers and set all bytes to zero.
    (b) Set bit identified by $k.s_{bit}$ high in byte number 8 in $S_{data}$
    (c) Increment anonymizer and set $H_{data}$ to be the binary representation if it. Set bit identified by $k.h_{bit}$ high in byte number 8 in $H_{data}$
    (d) Compute soft- and hard revocation hashes
        $H_s := H(CC_{NV\_SetBits} || x, || k.sri.name || s_{data})$
        $H_h := H(CC_{NV\_SetBits} || k.hri.name, || k.hri.name || h_{data})$
    (e) Compute soft revocation leaf digest as
        $l_1 := H(H(l_1 || CC_{PolicySigned} || k.RA.name))$
        $l_1 := H(l_1 || CC_{PolicyCpHash} || H_s)$
    (f) Compute hard revocation leaf digest as
        $l_2 := H(H(l_2 || CC_{PolicySigned} || k.RA.name))$
        $l_2 := H(l_2 || CC_{PolicyCpHash} || H_h)$
    (g) Add branch digest $b := H(CC_{PolicyOR} || l_1 || l_2)$ to $\beta$
(3) Compute finalPolicy := $H(CC_{PolicyOr} || \beta_1 || \beta_2 ... || \beta_n)$
(4) Output finalPolicy
```