Issue First Activate Later Certificates for V2X

– Combining ITS efficiency with privacy –

Eric R. Verheul[⋆]

KeyControls, Radboud University Nijmegen,
P.O. Box 9010, NL-6500 GL Nijmegen, The Netherlands.
`eric.verheul@[keycontrols.nl,cs.ru.nl]`

Draft 18th December 2016

Abstract We specify Issue First Activate Later (IFAL). This is an ETSI [11] type of V2X Public Key Infrastructure based on short-lived pseudonymous certificates without Certificate Revocation Lists. IFAL certificates are valid in the future but can only be used together with periodically provided activation codes. IFAL supports controlled de-pseudonymization enabling provisioning to stop for misbehaving vehicles. IFAL allows for flexible policies, trade-offs between three essential V2X properties: trust, privacy and usability. IFAL activation codes are small and can be sent in an SMS, through roadside equipment or even broadcasted. Like the Butterfly scheme [32], IFAL uses key derivation with one base private/public key pair. However in IFAL the security module can be simple as it can be kept oblivious of key derivation.

Keywords: deniability of implicit certificates, elliptic curve cryptography, ITS, privacy, pseudonyms, V2X

1 Introduction

1.1 Background

Intelligent Transport Systems (ITS) refer to the use of ICT in road transport. An important ITS application consists of sharing information among road users and roadside infrastructure, e.g. on each other's position, dynamics and attributes. In this context vehicles (e.g. cars) can communicate to other vehicles (V2V), to the infrastructure (V2I) or vice versa (I2V). ITS network components are known as *ITS Stations*, cf. [6]. The ITS Station inside vehicles is commonly known as On-Board Equipment (OBE). Within OBEs a hardware security module resides managing cryptographic keys that we refer to as *Trusted Element* (TE). The combination of V2V and V2I is called V2X. According to [28] up to 75% of roadway crashes can be prevented through V2X usage. Other applications include enhanced traffic congestion and traffic light management. In these applications,

[⋆] Work done for the Dutch Ministry of Infrastructure and the Environment information from traditional inductive-loop traffic detectors is replaced by information sent by vehicles. This allows the road infrastructure to take decisions based on actual road usage, e.g. setting traffic lights on green or to open peak hour lanes. In V2X applications it is essential that the receiver can validate the authenticity of the sent message, i.e. that the message is sent by a genuine vehicle. For instance, it is obviously not acceptable in the traffic light and peak hour use cases if one vehicle would be able to impose as many vehicles. In other words, there should be a V2X mechanism allowing vehicles to assert message *trustworthiness*.

1.2 A simple setup and its drawbacks

A simple setup to achieve trust is by introducing a V2X Public Key Infrastructure (PKI). Here one lets the vehicle, or rather its TE, sign its messages using a digital certificate, cf. [12]. This setup, however, introduces privacy issues. Indeed, placing directly identifying information in the certificate, e.g. a number plate, would allow vehicle tracking by simply listing to ITS communications. Note that due to the nature of V2X, the intended recipients of messages are typically not known, implying that message encryption is not feasible. Placing non-identifiable information in certificates, i.e. pseudonyms, is not sufficient. Indeed, if the certificates would be used for a long time, then this would introduce tracking issues. That is, one would be able to gather messages from various locations and link them through the certificates used by the vehicle. By linking the certificate to the vehicle, e.g. through visual observation, the vehicle movements are trackable again.

1.3 The V2X PKI approach of ETSI

The European Telecommunications Standards Institute (ETSI) has published various (draft) standards on V2X PKIs. These standards include communication channels [10], cryptographic formats [12] and message services and formats required for this, e.g. [13] and [14]. Trust and privacy issues are specifically addressed by ETSI in [11] and [12]. The latter document specifies pseudonymous certificates and the first document specifies process in which they are issued. Pseudonymous certificates are issued under control of two parties called Enrolment Authority (EA) and Authorization Authority (AA). Both authorities are part of a broader public key infrastructure, e.g. share a common root. The EA can authenticate a vehicle under its canonical identifier, e.g. a number plate or a VIN number, based on a long-term certificate. The idea of [11] is that an OBE of a vehicle can request the EA for a pseudonymous certificate whereby authenticating with its long-term certificate. The EA verifies the request and if successful provides the OBE an enrolment credential to be used at the AA. The OBE then uses this enrolment credential to request the AA for a pseudonymous certificate. After successful validation, this certificate is then returned to the OBE by the

AA.[1] For privacy reasons, cf. Section 2.2, a vehicle should be able to periodically change pseudonymous certificates. A drawback of the ETSI specifications is that it does not cater for the request of a batch of certificates. This implies that the above protocol needs to executed several times, which can be cumbersome. Another drawback is that the direct communication between OBE and AA introduces the risk that the AA gathers indirectly identifying information such as IP addresses. We do note that all communications between the OBE and the EA and AA are encrypted.The setup described in [32] caters for certificates batches and lets all OBE interaction be conducted by the EA function (called Registration Authority).

In a simple setup the vehicle's TE would only need to manage one private signing key in the TE, whereas in the described approach it would need to manage as many signing keys as there are pseudonymous certificates. In essence a pseudonymous certificate is a structured message signed by the AA that holds the following fields: version, signer info, subject info, subject attributes and a validity period. Compare [12, Section 6.1]. As part of the subject attributes, a certificate typically contains a public key of the vehicle. To limit the data in the certificate, this public key can also serve as the pseudonym in the certificate.

Trust and privacy are somewhat conflicting. If the vehicle is able to frequently change certificates it might also be able to impose as multiple vehicles, also known as a Sybil attack [20]. The traffic light and peak hour use cases illustrate that Sybil attacks are a genuine concern. Indeed, through a Sybil attack one car might for instance set the traffic light to green by imposing as many cars. On the other hand, the vehicle must be able to do a controlled certificate roll-over. This means that it should be possible, at least for a short time, that two (and only two!) pseudonymous certificates are simultaneously valid. For optimal unlinkability one would use pseudonymous certificates that are only valid for a very short time period, or even only once.[2] However, practical use cases require that vehicles are uniquely identifiable for a "short" time period by other parties. This is also illustrated by the traffic light and peak hour use cases. Additionally, in case of systematically misbehaving vehicles, e.g. causing problems or even accidents, it is important that pseudonym certificates can be deactivated. The regular PKI deactivation mechanism for this would be distribution of Certificate Revocation Lists (CRLs) to vehicles. However, timely CRL distribution is considered too challenging for pseudonymous certificates, cf. [15]. This is not only due to the size of the CRLs but also as it requires the vehicle to regularly connect to the infrastructure to download CRLs. Additional to certificate revocation it might in some use cases be desirable that law enforcement can assess the identity of a vehicle corresponding to a pseudonym certificate, i.e. to perform de-pseudonymization. Compare [20]. We note that the ETSI term *misbehaving vehicle* also includes misbehaving (malicious) drivers for which law enforcement action might be required. Compare [7, Section 11.3.21].

[1] ETSI actually uses the terminology enrolment certificates and authorization tickets.

[2] We note that this is cryptographically possible based on so-called Idemix technology. However it seems not yet practically feasible. See [31].

A commonly chosen approach for this issue is to deploy short-lived pseudonyms that need to be regularly issued again, cf. [15]. Instead of revoking pseudonymous certificates of misbehaving vehicles one arranges that new certificates are not issued to them. Within this approach a mechanism needs to be in place whereby the EA is informed of a misbehaving vehicle (e.g. under his canonical identifier) and then ensures that no new enrolment certificates are issued. Although this approach avoids CRL distribution, it still requires the vehicle to periodically connect to the infrastructure to request and download new pseudonymous certificates. As this can be a data intensive operation it forms a barrier for vehicles with none or only limited connectivity capabilities. This can correspond with low-end vehicles but also with vehicles that are not in reach of good quality communication networks.

The approach chosen in this paper is to issue large batches of short-lived certificates to vehicles that are valid in the (far) future but that can only be used when the vehicle is periodically provided short activation codes. Misbehaving vehicles will not be provided such codes. Based on this approach we develop a V2X PKI scheme called Issue First Activate Later (IFAL) as first introduced in [31].

1.4 Paper outline

In Section 2 we further formalize IFAL requirements. A first high level IFAL overview is given in Section 3 whereby we also introduce the IFAL policy notion. Section 4 gives a detailed description of the basic IFAL scheme and its use of cryptography. In Section 5 we discuss IFAL implementation choices on certificates and the distribution of IFAL activation codes. Section 6.1 contains a comparison of IFAL with the requirements from Section 2. Here we also describe some IFAL extensions. In Section 7 we compare IFAL with the Butterfly scheme [32]. Section 8 contains conclusions. Finally, the appendix demonstrates a signature repudiation issue with pseudonymous implicit certificates defined in the ETSI specifications [12] and used in the Butterfly scheme [32].

2 IFAL Requirements and options

In Section 1 we have introduced two basic properties of V2X: Trust and Privacy. These properties are explicitly discussed by ETSI in [11]. However, there is a very important third V2X property: Usability. Trust and Privacy can be conflicting with Usability. Indeed, we have indicated in the introduction that if vehicles always have a high bandwidth connection with the internet one can implement a V2X PKI meeting Trust and Privacy properties relatively easy. However such internet connectivity will hamper usability of the scheme as such connectivity is not always available, e.g. in certain areas or in low-end vehicles. On the other hand, the low capabilities of some vehicles should not hamper the trust and privacy of *all* vehicles. In this paper we take the position that various, compatible V2X PKIs can exist next to each other. This allows users and relying parties to choose a trade-off between the identified three properties. Moreover, this choice should be transparent for both the user and relying parties. From a privacy perspective this transparency is actually a legal requirement [16].

To summarize, our first V2X PKI scheme requirement is to allow for various and transparent scheme policies allowing for particular trade-offs between Trust, Privacy and Usability. In the three sections 2.1, 2.2 and 2.3 below we further elaborate on the requirements and options on Trust, Privacy and Usability.

2.1 IFAL trust requirements and options

Trustworthy ITS message signing This is a minimal trust requirement; the vehicle is able to digitally sign ITS messages in a reliable fashion. Also, the PKI infrastructure shall reliably bind the public signing key to the vehicle pseudonym and to the vehicle itself. In some cases, e.g. in traffic accidents handling, it is crucial that an ITS message can be traced back to a vehicle in an undeniable way. See [20]. This implies that setup shall support non-reputation. High trust can be achieved by using a Trusted Element to securely manage (generate, store and use) ITS private keys. Lower trust is provided when the ITS private keys are managed in the OBE itself.

Sybil attack resistance It should be possible to limit the number of simultaneously valid pseudonymous certificates in a vehicle. Trust in this aspect is inversely proportional to this number. The highest trust is achieved when this number is one. We remark that in trade-off with usability we choose to have this number equal to two albeit in a configurable manner.

PKI removal of misbehaving vehicles For the trust of users in the ITS messages it is essential that misbehaving vehicles are removed, e.g. by revocation, from the PKI. Such vehicles can for instance correspond to vehicles that systematically send erroneous messages. Trust in this aspect is inversely proportional to the elapsed time of removal after reporting.

Law Enforcement support (Optional) Trust of ITS users might be further enhanced if misbehaving vehicles that send erroneous messages with malicious intend can also be prosecuted. For this controlled de-pseudonymization of certificates would be required. We consider this an option instead of a requirement, as societal acceptability of this functionality is culturally based.

2.2 IFAL Privacy requirements and options

Pseudonym unlinkability The V2X certificates should be as unlinkable as possible. This property only partly corresponds to the cryptographic quality of the pseudonym in the certificate. It also corresponds to the vehicle data in the certificate and the life-time of the certificate. As (unique) vehicle data in the certificate can lead to indirect linking or even identification, less vehicle data corresponds to higher privacy. Moreover, the less data a certificate contains, the lower the linking risk becomes. Generally speaking, for this aspect privacy quality is inversely proportional to the lifetime of the certificates.

Pseudonym unlinkability at EA and AA Depending of the V2X design the EA or the AA might be able to link different pseudonymous certificates or even to identify the vehicle. As a minimal requirement we set that the EA and the AA should not be able to link pseudonymous certificates used in operation.

Pseudonym unlinkability at EA and AA technically enforced (Optional) Higher privacy is achieved if it technically enforced that the AA cannot link pseudonyms. The highest privacy level is achieved if even the EA and AA *together* would not technically be able to link pseudonyms. IFAL can optionally support this privacy level. However, one can argue if requiring this privacy level at the (trusted) EA and AA is proportional to the trust one has to place in other ITS parties anyway. For instance, a malicious OBE supplier could supplement ITS communication with hidden identification information. As privacy is culturally based we have stated this property as an option only.

2.3 IFAL usability requirements

ETSI conformity For compatibility reasons, we strive for conformity with the ETSI approach [11] as much as possible. In essence we will only supplement the ETSI approach with the ability to provide a batch of certificates in one protocol execution.

Certificate pre-install From a usability perspective it would be ideal if a vehicle would be equipped with all certificates it ever needs as part of the vehicle manufacturing process. That is, all required certificates are installed in the car factory. This is ideal as then the vehicle never needs to contact the EA/AA again. More generally, we consider the total lifetime of the certificates that can be installed in the vehicle in one EA/AA interaction as a measure of usability.

No CRL distribution The management of a V2X PKI CRL will be quite complex when the number of certificates increases due to their short lifetime. By requiring CRL distribution one also requires the capability of a vehicle to download substantial large (CRL) files. A scheme that does not require CRLs scores higher on usability than one that requires them.

Low vehicles connectivity Posing high connectivity and/or bandwidth requirements on vehicles hampers usability. Usability corresponds inversely proportional with the requirements on connectivity and bandwidth. Optimal usability would be achieved if no connectivity would be required at all. Usability in this aspect also corresponds to the flexibility of communicational channels, e.g. that not only point-to-point connections are usable but also broadcast channels through radio (RDS) or roadside equipment.

TE Simplicity An OBE in a vehicle can be compared with a generic platform such as a PC or a smartphone. As such, periodic updates of the OBE software are common, e.g. to improve functionality or security. By their nature, periodically updating the software in a Trusted Element is not common. This is partly due to the high security assurance level a TE needs to provide. By the same reason, it is not easy to get new functionality in place in TEs. Consequently, if a scheme requires new TE functionality it will not be able to support the installed base or to quickly have support in all TEs. This means that from a usability perspective it is best if a scheme only requires basic TE functionality whereby all complexity is done in the OBE that is flexible by nature. There might be motivations to perform all cryptographic operations in the TE; we only require it is not mandatory. In principle this approach can be conflicting with scheme trustworthiness, cf. Section 2.1. We show that in IFAL very basic TEs suffice without influencing trustworthiness.

Low certificate OBE storage If one stores many certificates in an OBE, it is important that this imposes as little as possible storage requirements on the OBE.

3 High level IFAL description

3.1 IFAL basic concepts

The approach in IFAL is to issue short-lived certificates to vehicles that are valid in the far future but that can only be used when the vehicle periodically receives activation codes. Misbehaving vehicles will not be sent such codes. Based on the current time the vehicle's OBE can select the valid IFAL certificates and use them. IFAL certificates have a (small) overlap in validity to support controlled certificate rollover and to make IFAL less dependent on system-wide time synchronisation. IFAL certificates are issued by the AA in the form of an *IFAL certificate file* and are activated in groups corresponding with *epochs*, i.e. periods of time. Associated to a certificate file is an *IFAL policy* defining choices with respect to the requirements in Section 3.1. Specific for IFAL four choices exist:

- the validity period of an IFAL certificate, e.g. 12 minutes,
- the time overlap of successive IFAL certificates, e.g. 2 minutes
- the number of certificates in an epoch and thus implicitly the time span of an epoch, e.g. 90 days,
- the number of epoch in the file and thus implicitly the total number of IFAL certificates in the file. This implicitly also corresponds to the total time span of the file, e.g. 10 years.

The lifecycle of an IFAL certificate consists of the following phases: Issuing, Usage, Activation and De-Authorization.

3.2 High level description of IFAL Issuing

We have outlined the issuance process in the diagram 1. The OBE instructs the TE to generate a *base* public/private key pair. The OBE itself also generates a public key pair allowing the EA and AA to encrypt data for the OBE in an end-to-end fashion. Of course, this can also be handled inside the TE. Next the OBE forms an IFAL certificate request that includes these public keys and sends

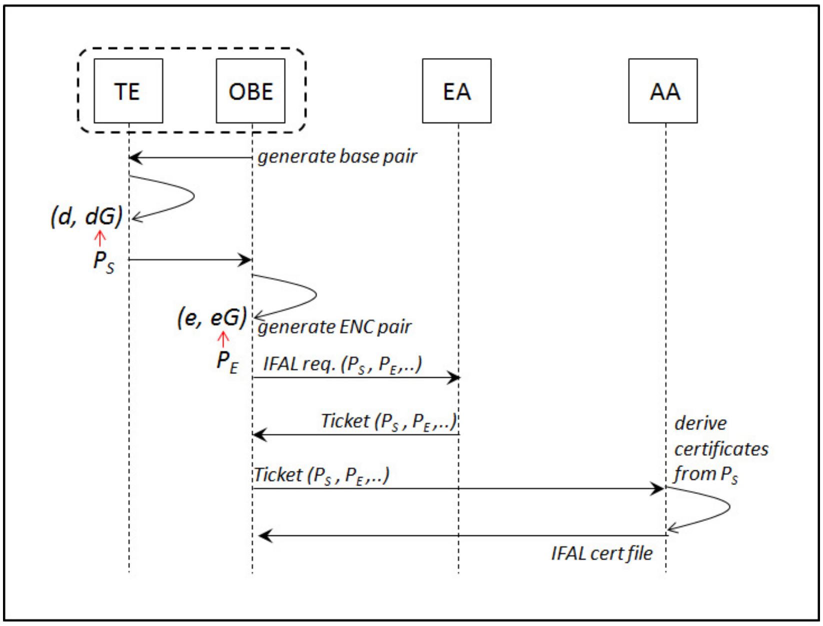

Figure 1. IFAL issuance process

this to the EA together with a reference to an IFAL policy, cf. Section 3.1. The OBE authenticates the request, e.g. by signing it with a long-term certificate issued by the EA managed by the TE. As part of this step the EA also agrees with the OBE on a channel over which IFAL epoch activation codes will be sent.

The EA validates if the OBE is authorized for the IFAL certificate file (and the IFAL policy). If this is the case the EA returns a signed authorization response (credential) to the OBE which also includes the base certificate request and public key. The credential also contains a unique, random identifier generated by the EA allowing the EA and AA to later exchange activation information related to the IFAL certificate file.

The OBE then requests the AA for an IFAL certificate file and accompanies that with the credential. The AA then validates this request. If this is successful the AA forms the requested IFAL certificates in accordance with the requested IFAL policy. The public key P in each such a certificate is derived by the AA from the base public key the OBE sent. The OBE and TE together, and only together, are able to determine the private key of P as a key derivation. This derivation is based on the base private key, the start date/time of the certificate and a secret cryptographic key from the AA. This derivation is such that only with the base private key and this secret cryptographic key one is able to derive the private key of P. Such a secret cryptographic key is valid for a subset of certificates corresponding with an *epoch*. To this end, epoch secret cryptographic keys are periodically provided to the OBE, i.e. as part of activation of an epoch. For this the IFAL certificate file is accompanied with an IFAL transport key under which the epoch secret keys are encrypted resulting into *IFAL activation codes*. The encryption allows for communication of these codes over open (public) channels. The transport key is also registered at the AA under the identifier shared with the EA. The activation codes are formed by the AA but provided to the OBE by the EA.

On receipt of the IFAL certificate file and the first IFAL activation code the OBE together with TE are able to derive the private keys related to the first epoch. To reduce the linkablility risk at the AA, the AA is required to delete the IFAL certificates produced as well as the vehicle basic public key on which they are based.

3.3 High level description of IFAL Usage

We have outlined the usage (signing) process in the diagram 2. When the OBE

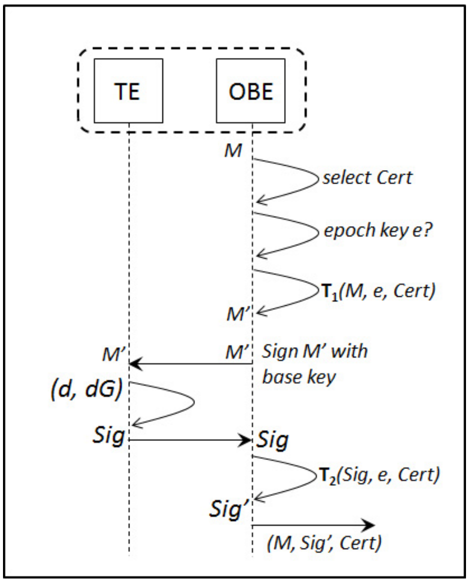

Figure 2. IFAL usage (signing) process

wants to sign an ITS message M it selects the relevant IFAL certificate based on the current time. As we allow two certificates to be overlapping the OBE might have a choice typically corresponding to using a previously used certificate or rolling over to a new one. The OBE next determines if it possesses the required epoch key otherwise it will provide an error to the user. With the epoch key the

OBE transforms the message M to another message M' based on the selected certificate and the epoch key. The OBE then requests the TE to sign M' with its base private key. The returned signature is then transformed by the OBE to the requested one, i.e. corresponding to the selected certificate. Note that this setup imposes minimal restrictions on the TE; the TE is only signing messages with its base private key.

3.4 High level description of IFAL Activation

We have outlined the IFAL activation process in the diagram 3. Periodically the AA compiles a list of all (still) active OBEs. Then the AA forms and distributes the epoch activation codes to active OBEs. For each such OBE the AA determines (generates) the epoch key, encrypts it with the transport key and sends the resulting activation code to the EA accompanied with the OBE identifier Id_O. The EA then delivers the activation code to the vehicle using the channel agreed during the IFAL request phase.

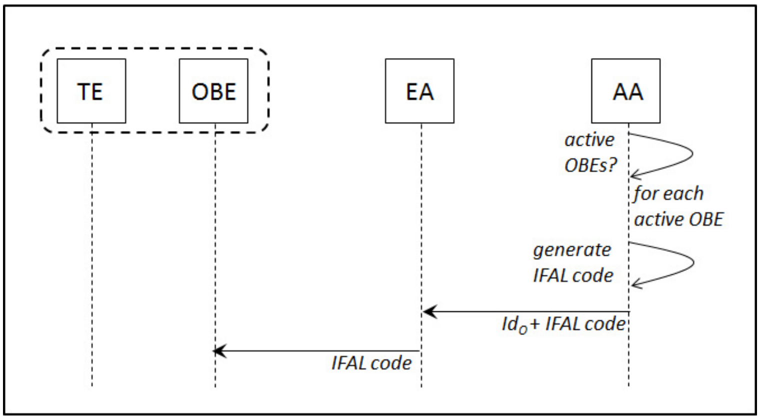

Figure 3. IFAL activation process

3.5 High level description of IFAL De-Authorization

In this phase designated OBE/TE combinations that will no longer be provided with epoch activation codes. This could be temporary (corresponding to suspension) or definitive (corresponding to revocation). These combinations can emerge from two sources. The first source is the EA. Here the trigger for invalidation is from the non-pseudonymous world, e.g. the vehicle owner selling its car or the police noticing abuse with a particular vehicle. For this the EA simply sends an invalidation message to the AA accompanied with the OBE/TE identifier shared with the AA. The second source is based on an IFAL certificate of which an authorized party reports that it corresponds to a misbehaving OBE/TE. Typically the authorized party will be the police that struck upon the certificate as part of an inquiry. Here the AA is able to determine the OBE/TE identifier from the certificate and invalidate it. In the basic IFAL scheme the AA will be able to determine the OBE/TE identifier on its own. As an IFAL extension this will be performed under dual control, i.e. together with another party.

4 Detailed description of IFAL

4.1 IFAL data structures

As explained in Section 3.1, IFAL certificates are associated with an IFAL Policy corresponding to the following parameters of type integer. Compare Figure 4.

- *Certificate Validity* T_V the time (e.g. in seconds) an IFAL certificate is valid,
- *Certificate Overlap* T_O the time that two consecutive certificates are allowed to overlap,
- *Number of Certificates* N_C the number of IFAL certificates in an IFAL file,
- *Number of Epochs* N_E the number of epochs in an IFAL file.

The number of certificates in an epoch divides N_C and is equal to N_C/N_E. Here we let "/" represent integer (Euclidian) division. We will number the certificates and epochs starting at zero. We also require that:

$$0 \leq T_O \leq 0,5 * T_V. \tag{1}$$

The first inequality in Formula (1) ensures that always at least one certificate is valid, the inequality ensures at most two certificates can be simultaneously valid. Based on the certificate validity and overlap we also have the parameter *Certificate Step* given by $T_S = T_V - T_O$, which gives the time difference between the start dates of consecutive certificates in the sequence. From formula (1) it follows:

$$0,5 * T_V \leq T_S \leq T_V. \tag{2}$$

An IFAL certificate file consists of:

- an IFAL Policy,
- a Start time S corresponding to start time of the first certificate in the file, i.e. the *start_validity* of [12],
- a sequence of certificates $C_0, C_1, C_2, ...$

The expiry time of the first certificate is equal $S + T_V$. The start time of the second certificate in the IFAL file is equal to $S + T_S$ and so on. An IFAL file can provide valid certificates in the time period $[S, S + N_C * T_S + T_O]$. The validity period of the i-th certificate C_i is $[S + i * T_S, S + i * T_S + T_V]$. On time t in validity period of the IFAL file, the OBE can determine the following operational parameters.

- the *current* certificate C_j by calculating $j = (t - S)/T_S$,
- the *next* certificate C_{j+1}. If $t \geq (j+1) * T_S$ then the OBE is able to use (role over) to this certificate,
- the epoch number of the current certificate j/N_C,
- the epoch number of the next certificate $(j+1)/N_C$.

Conversely, based on the certificate number $i \in [0, N_C]$ one can calculate the start validity and end validity times as

$$\text{start\_validity}(i) = i * T_S \ ; \ \text{end\_validity}(i) = i * T_S + T_O. \tag{3}$$

In the sections below we now discuss the cryptographic details of the four IFAL phases identified in Section 3, i.e.: Issuing, Usage, Activation and De-Authorization.

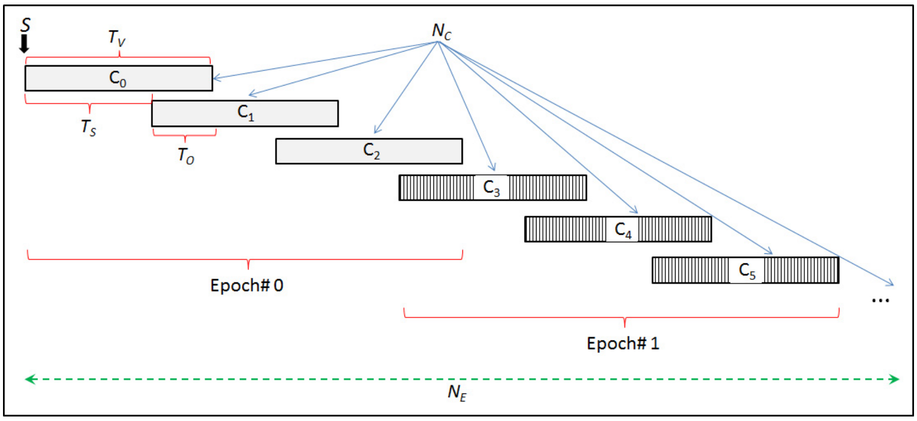

Figure 4. IFAL policy

4.2 Cryptographic assumptions and notation

Before we discuss the IFAL details we describe the cryptographic context we use. The cryptography in this paper is based on three categories: asymmetric (public key) cryptography, key derivation techniques and symmetric cryptography. Below we have indicated the relevant assumptions for each category.

Symmetric and asymmetric (public key) cryptography
Throughout this paper we let $\mathcal{E}(K, M)$ represent a symmetric encryption of a message M under a key K, e.g. [21]. Asymmetric cryptography in this paper will be based on an additive group $(\langle G \rangle, +)$ of order q generated by a generator element G. We use additive notation as this is customary in the context of elliptic curve groups which are stipulated by ETSI, cf. [12]. We assume that q

is prime and by $\mathrm{GF}(q)$ we denote the Galois field of the integers modulo q. We also assume that the group $(\langle G\rangle, +)$ is cryptographically secure, i.e. that the so-called discrete logarithm and Diffie-Hellman problem are intractable, cf. [18]. This implies that the size of the group order q in bits should be at least 256 bits. For practical implementations one can use a group of points G on one of the Brainpool curves [5] or NIST curves [22].

Key derivation functions
In this paper we deploy various cryptographic algorithms ideally requiring random cryptographic keys of a certain form and length. With a *key derivation function* $\mathcal{K}(.,.)$ or simply KDF one can deterministically derive such keys pseudorandomly. This is based on a suitably chosen(“master”) key K and an arbitrarily sized *derivation string* D. The basic KDF security property is that provided K is suitable chosen using derived keys from different derivation strings is as secure as using truly random keys. In [24] the US National Institute of Standards and Technology (NIST) has specified several key derivation functions. In this paper we need three types of KDFs, which we represent by $\mathcal{K}_1(), \mathcal{K}_2()$ and $\mathcal{K}_3()$. Only the first two key derivation functions will be used by the OBE; the AA will typically use all three functions. The first function $\mathcal{K}_1()$ provides (pseudorandom) keys K for the encryption algorithm $\mathcal{E}(K,.)$ we have introduced above. When this algorithm is based on AES-128 then $\mathcal{K}_1()$ would thus need to provide binary strings of size 128 bits. Key derivation function $\mathcal{K}_1()$ is a standard application of [24]. We remark that functionally we can use the key derivation function of [32] as $\mathcal{K}_1()$ for AES-256. This function takes the form

$$\mathcal{K}(K, D) = \mathrm{AES}_K(0^{128} \oplus D) \mid\mid \mathrm{AES}_K(1^{128} \oplus D), \tag{4}$$

where AES represents the AES blockcipher [21] and || represents concatenation. Moreover, in this setup D is a bit string of size 128 bits and 0^{128} (respectively 1^{128}) represents a bit string of 128 zero bits (respectively 128 one bits). However, the function $\mathcal{K}()$ does not meet the formal requirements. Indeed, for any D, the last 128 bits of $\mathcal{K}(K, D)$ are equal to the first 128 bits of $\mathcal{K}(K, 0^{128} \oplus D)$ and vice versa. This is not allowed behavior for a key derivation function. It depends on the usage of the key derivation function whether this introduces a security risk. To avoid such issues we prefer the use of standardized key derivation functions.

The second key derivation function $\mathcal{K}_2()$ we require provides non-zero elements in $\mathrm{GF}(q)$. One could take any of the KDFs specified in [24] and take its output modulo q. However, in this setup there is a (very small) probability that this results into zero output. In [4] the German Bundesamt für Sicherheit in der Informationstechnik (BSI) states two approaches to address this issue which we will both use. For $\mathcal{K}_2()$ we choose any KDF, e.g. from [24] and a key K of appropriate size. Based on a derivation string D we generate a binary string of size $|q| + 64$ bits. We interpret that as an integer r of size $|q| + 64$ bits and let

$$r \bmod (q-1) + 1,$$

be the value of $\mathcal{K}_2()$. This does not produce a uniform distribution but the deviation from it is assumed not to be exploitable.

The third function $\mathcal{K}_3()$ also provides non-zero elements in $\mathrm{GF}(q)$ but has an additional *recovery* property. This property is that with the value $\mathcal{K}_3(K, D)$ and the master key K one is able to derive the derivation string D. This essentially means that $\mathcal{K}_3()$ produces symmetric encryptions of D as derived keys. Of course, this imposes restrictions on the length of D as the length of the key derivation function is fixed. As an example we show such an construction for a group order q of size 256 bit which is most relevant for ITS, cf. [19]. This construction is also compliant with the key derivation specifications [24] of NIST. To this end, we choose the following configuration from [24] (using its terminology):

- as the pseudorandom function choose CMAC based on AES,
- use this function in Counter Mode,
- as length r of the internal counter choose 8, i.e. fits in one hex nibble, namely 0x8.
- only allow an empty Label and a Context string of precise size 14 bytes
- do not use an indicator,
- choose the length of the derived keys to be equal to 256 which fits in 3 hex nibbles, namely 0x100

We also only allow derivation strings D of size 14 bytes and use this as the Context string. What then happens is that this KDF generates its 256 bit values as two concatenated AES-CMAC values (one with counter equal to 1 and one equal to 2). Each is a CMAC value based on only one AES input block. This means that the Context string is decryptable from each of these values using the derived key and the master key K. Next we consider the output as a 256 number k and take $k' = k \bmod q$ as the output. Now, if $k < q$ we have $k = k'$, otherwise we have $k = k' + q$. This means that the original value k' can always be constructed from the key derivation output and from that the derivation string D by inverting the CMAC operation. This inversion also allows to asses which of the two values is the correct one. Finally, the constructed value k' could be zero. In this case, one needs to choose another derivation string in line with the second approach of BSI [4]. In our application of $\mathcal{K}_3()$ an incrementing sequence number is a part of the derivation string will consist which allows for this. Actually, one can easily show that is can only happen at most once implying that this event will not ever occur in practice.

4.3 Issuing of IFAL certificates

We refer to the process of Section 3.2 and fill in the cryptographic details. The base private/public key pair generated by the TE takes the form (d, dG) where $d \in \mathrm{GF}(q)$ is the base private signing key and $P_S = dG$ the corresponding base public key. For later reference we describe how an OBE uses the TE to sign a message M with its private key d. To this end, the OBE computes the hash h of the message M and sends a request to the TE to sign this with its private key. So the OBE is computing the hash value and not the TE. This is a common setup as the computational power of the OBE is typically higher than that of the TE. Moreover, there is no security advantage to let the TE calculate the hash as the OBE can send any message M anyway.[3] The TE then proceeds as specified in the pseudo-code of Algorithm 1. This description is taken from [18, Section 4.4.1]. IFAL is flexible on the asymmetric encryption scheme allowing the EA

Algorithm 1 Standard ECDSA signing of a hash value h by TE

```
1: procedure ECDSA Sign(h)
2:     Select random $k \in [1, q-1]$.
3:     Compute $kG = (x, y)$ and convert $x$ to an integer $\bar{x}$.
4:     Compute $r = \bar{x} \bmod q$. If $r = 0$ then go to Line 2.
5:     Compute $s = k^{-1}(h + d \cdot r) \bmod q$. If $s = 0$ then go to Line 2.
6:     Return $(r, s)$.
7: end procedure
```

and AA to encrypt data for the OBE/TE. However, to stay in line with ETSI we choose the Elliptic Curve Integrated Encryption Scheme (ECIES) scheme, based on the same curve used for the signatures. ECEIS is based on the Diffie-Hellman key exchange, which we assumed to be secure, cf. Section 4.2. To this end, the OBE/TE randomly chooses $e \in \mathrm{GF}(q)$ and forms the ECIES public key $P_E = eG$. Based on this, the OBE requests the EA for an IFAL certificate file of a certain IFAL policy. The EA registers the requests and validates it. On success it will provide the OBE with an authorization response (credential) to the OBE which also includes the base certificate request and the public key. The credential also contains a unique identifier Id_O, of say eight bytes in size. This identifier does not contain (in)direct identifying information and is only shared by the EA and AA to later exchange (activation) information related to the IFAL certificate file. Next the OBE sends the credential to the AA and requests the actual IFAL certificate file.

The AA uses the identifier Id_O to register a new entry for the generation of a new IFAL certificate file. Based on the IFAL policy in the certificate request, the AA determines the number N_E of epochs the file needs to cover. For each epoch the AA generates an epoch symmetric key, i.e. $K_0, K_1, ..., K_{N_E-1}$ and stores that in the registry under the new entry. The AA then generates the IFAL certificate file as specified in pseudo-code in Algorithm 2.

In Line 1 a new IFAL file is created in which metadata is written in Line 2 allowing it be parsed. This metadata includes an IFAL policy identifier as specified in Section 3.1, a start validity time of the first IFAL certificate and information on how to decode individual certificates in the file. The file also contains a secret symmetric transport key K_T encrypted with the public key P_E that was part of the certificate request.

[3] The message hash $h \in \mathrm{GF}(q)$ has the role of *verifier challenge* in $\mathrm{GF}(q)$ Schnorr's three pass protocol [29] on which ECDSA is based. The hash makes the protocol non-interactive and ensures this challenge is unpredictable for the prover, i.e. the signer in our context. As this is unchanged, there is no cryptographic risk in letting the card sign hash values it did not compute itself.

The parameter i in the "for loop" in Lines 3-12 corresponds with the i-th certificate to be generated. Line 4 determines the epoch number j the i-th certificate corresponds to in accordance with the formulas in Section 3.1. In Lines 6-7 the validity times of the i-th certificate are calculated in accordance with Section 3.1. The calculation in Line 7 is fundamental in our construction. Here the OBE public key in the certificate is formed based on the base public key and a derived secret $\mathcal{K}_2(K_j, \texttt{ToString}(i))$. This secret is based on the j-th epoch key K_j and the certificate number i represented as a string of fixed size. The private key corresponding to this public key is equal to the product of the vehicle base private key d and the derived secret. As only the vehicle TE has access to d, only this can use the private key provided it has access to the derived secret. Line 8 forms the certificate contents based on the mandatory and optional fields. This content is then signed by the AA in Line 9 using its private key in the CSIGN procedure. This procedure also takes Id_O as input which we explain in the context of Algorithm 3. Then Line 10 forms the actual certificates and is added to the file in Line 11. The generated IFAL certificate file is then returned in Line 12.

Algorithm 2 Generation of IFAL certificate file for entry Id_O

```
 1: Generate new FILE
 2: Add metadata to FILE
 3: for $i = 0$ to $i = N_C - 1$ do
 4:     $j = i/N_E$;
 5:     start_validity $= S + i * T_S$;
 6:     end_validity $= S + i * T_S + T_V$;
 7:     $P_i = \mathcal{K}_2(K_j, \texttt{ToString}(i)) * P_S$;
 8:     Cont = Cert_Fields(start_validity, end_validity, $P_i$, other fields)
 9:     Sig = CSIGN(Cont, $Id_O$)
10:     Cert = Form_Cert(Cont, Sig)
11:     Add Cert to FILE
12: end for
13: return FILE
```

We denote the AA private/public signing key pair by (a, aG). The AA signature in Line 9 of Algorithm 2 is of a particular ECDSA type as it supports de-pseudonymization. For this we let the AA certificate signatures be formed in a deterministic variant of ECDSA, similar to [19]. This is detailed in Algorithm 3. To this end, the AA deploys a key K_a, a *secure counter* SC_a of fixed size (we suggest 6 bytes) associated with its private signing key a. Both are initialized as part of generation of the private key a, i.e. the key K_a is then randomly selected and the counter SC_a is then initialized to zero. This counter is used as part of the signature creation and is securely incremented with each signature. When the counter reaches its maximum value SC_{MAX} the AA needs to generate a fresh private key and set the secure counter to zero again. With a secure counter consisting of six bytes as we suggest this will happen after the generation of $2^{48} \approx 3.10^{14}$ certificates. So this will not be an issue in practice; the signing key will be renewed before this number is achieved. We note that it is good practice in the context of electronic passports, cf. [17, Section 4.1.2] to limit the number of signatures placed with a document signer, a key comparable with the AA signing key. The counter needs to be *secure* in the sense that it shall not be possible that two signatures are formed with the same value of the secure counter. A secure counter is not a new concept. Typically each smartcard with PIN protection uses such a counter to register the number of incorrect PIN entries. The smartcard needs to withstand attacks preventing it to increment the counter as this would allow brute-forcing the PIN. A standard counter security mechanism is to first increment the counter and then perform the operation instead of vice versa. We will envision Hardware Security Modules (HSMs, i.e. "large" smartcards) for the protection of the counters; these are typically used to protect the AA signing key anyway. In principle HSM protection is easier to arrange than smartcard protection; HSMs can have more technical capabilities and can also reside in a controlled environment. Compare [23]. The signature algorithm is specified in pseudo-code in Algorithm 3.

The input to Algorithm 3 is the certificate contents to be signed together with the identifier Id_O of the certificate file. This algorithm follows Algorithm 1 with the difference that there parameter k is chosen randomly instead of based on a key derivation. In lines 2-4 the pseudorandom number k is generated using the key derivation function $\mathcal{K}_3()$. In a regular setup, k would be randomly chosen. Lines 5-9 follow the regular signing ECDSA signing process based on k. The result is then returned in Line 10.

Algorithm 3 IFAL Certificate Signing by AA

```
1: procedure CSIGN(Cont, $Id_O$)
2:     Compute $SC_a = SC_a + 1$.
3:     If $SC_a = SC_{MAX}$ then generate new AA signing key.
4:     Compute $k = \mathcal{K}_3(K_a, SC_a||Id_O)$. If $k = 0$ then go to Line 2.
5:     Compute $kG = (x, y)$ and convert $x$ to an integer $\bar{x}$.
6:     Compute $r = \bar{x} \bmod q$. If $r = 0$ then go to Line 2.
7:     Compute $h$ = HASH(Cont).
8:     Compute $s = k^{-1}(h + a \cdot r) \bmod q$. If $r = 0$ then go to Line 2.
9:     Return $(r, s)$.
10: end procedure
```

4.4 Usage of IFAL certificates

In this use case the vehicle wants to sign a message M. For this, the OBE computes the hash h of the message and has this signed by the TE. This is specified in pseudo-code in Algorithm 4. Lines 1-5 follow Section 3.3; here the OBE selects the appropriate certificate number i', epoch number j' and epoch key $K_{j'}$. The cryptography starts in Line 7; here a temporary, derived key drv is computed by the OBE that allows the TE to be oblivious of IFAL pseudonymization. Next the OBE divides the hash h by drv modulo q in Line 8. This value is then signed by the TE in Line 9 using its base private key d. After multiplying the second part of this signature with drv in Line 10, the signature is returned in Line 11. Note that the division with drv (respectively multiplication with drv) corresponds with transformation $T_1()$ (respectively $T_2()$) from Figure 2.

Algorithm 4 IFAL Signing of hash value h by Vehicle's OBE using TE

```
 1: procedure MSIGN(Mesg)
 2:     Determine current time $t$.
 3:     Determine current certificate number $i$ and epoch $j = i/N_C$.
 4:     Determine next certificate number $i+1$ and epoch $j = (i+1)/N_C$.
 5:     If next certificate is also valid on $t$ let $i', j'$ be the choice.
 6:     If epoch key $K_{j'}$ is not available return error.
 7:     Compute $drv = \mathcal{K}_2(K_{j'}, \texttt{ToString}(i'))$.
 8:     Compute $\hat{h} = h \cdot drv^{-1} \bmod q$.
 9:     The OBE lets the TE perform MSIGN($\hat{h}$) resulting in $(r, s)$.
10:     Compute $s' = s \cdot drv \bmod q$
11:     Return $(r, s')$.
12: end procedure
```

The value (r, s') returned by Algorithm 4 in Line 12 indeed is a signature with respect to the public key certified in the i-th certificate. Indeed, for the signature in Line 10 holds (cf. Algorithm 1):

$$(r, s) = (\bar{x} \bmod q, k^{-1}(\hat{h} + d \cdot r),$$

for some random $k \in [1, q-1]$ with $k * G = (x, y)$. This means that

$$\begin{aligned}(r, s') = (r, s \cdot drv) &= (\bar{x} \bmod q, k^{-1}(\hat{h} \cdot drv + drv \cdot d \cdot r) \\ &= (\bar{x} \bmod q, k^{-1}(h + \mathcal{K}_2(K_{j'}, \texttt{ToString}(i')) \cdot d \cdot r).\end{aligned}$$

Here the last equality follows from Line 7 of Algorithm 4. By construction the private key corresponding to the i'-the certificate is equal to

$$\mathcal{K}_2(K_{j'}, \texttt{ToString}(i')) \cdot d.$$

This means that (r, s') is a signature with respect to this certificate.

4.5 Activation of IFAL certificates

Compare Section 3.4. Periodically, e.g. near the end of epochs, the AA will create a list of identifiers Id_O corresponding with vehicles that are still activated.

For each such identifier, the AA picks the epoch key for the coming epoch and encrypts that with the symmetric transport key K_T that was part of the IFAL certificate file. The resulting IFAL activation code is then sent to the EA accompanied with the identifier Id_O. The EA will then send the encrypted epoch key to the vehicle.

4.6 De-Authorization of vehicles

De-Authorization can be triggered in two ways. The first way is through the EA. Here the EA gets a request to block sending activation codes to a certain vehicle. The EA then maps this to the internal identifier Id_O shared with the AA and sends a De-Authorization request to the AA referring to this identifier. This allows the AA to deactivate the vehicle in its registration implying that it will not be sent new epoch keys.

The second way is through an IFAL certificate issued by the AA corresponding to a misbehaving vehicle. If (r, s) is the certificate signature then this will take the form (cf. Algorithm 1):

$$(r, s) = (r, k^{-1}(h + a \cdot r) \bmod q)$$

where a is the AA private signing key and h the hash of the certificate. As the AA has possession of a it can then determine parameter k as

$$k = s^{-1}(h + a \cdot r) \bmod q.$$

Parameter k is formed in Algorithm 3 using the key derivation function $\mathcal{K}_3$. That is, $k = \mathcal{K}_3(K_a, SC_a||Id_O)$ for some values SC_a and Id_O. As $\mathcal{K}_3$ was designed recoverable in Section 4.2 this allows the AA to decrypt k and to determine Id_O (and SC_a). This enables the AA to proceed as above and to ensure the vehicle is no longer sent encrypted epoch keys.

5 IFAL implementation

5.1 Size of IFAL certificate files

IFAL certificate files can be quite small and in this section we quantify that. An IFAL certificate consists of static fields that are the same for all certificates and dynamic fields that are different for all certificates. Static fields include version numbers and vehicle information. To limit the risk of indirect identification, it is best to keep the vehicle information as minimal as possible. Typically, the dynamic fields in an IFAL certificate consist of the validity information (start validity, end validity), the vehicle public key and of course the AA signature. By design the validity information can be reconstructed from the position of the certificate in the IFAL certificate file together with the IFAL policy and the start validity of the first certificate. Therefore the validity information does not need to be explicitly contained in the IFAL certificate file. Similarly, the epoch numbers corresponding to the certificates can be computed. It follows that with the epoch keys, the OBE is able to reconstruct the IFAL vehicle public keys from the base public key. Compare Line 5 of Algorithm 2. We conclude that an IFAL certificate file consists of a small static part and a dynamic part consisting of the AA certificate signatures. As an illustration, suppose we want to issue IFAL certificates:

- that are valid for 12 minutes with a 2 minute overlap,
- with a total life span of 10 years.

In essence this requires that the IFAL certificate file contains $10 * 365 * 24 * 6 = 525.600$ AA certificate signatures. If we use an elliptic curve of size 256 bits then each such signature takes $2 * 256 = 512$ bits, i.e. 64 bytes. That is, the whole IFAL certificate file would consist of about 34 megabytes which is quite modest.

We remark it is good practice that certificate signatures are generated using a Hardware Security Module (HSM) certified against FIPS 140-2 Level 3. Compare [8, Section 6.5.2] and [23]. Modern HSMs also support custom modules to be built allowing Algorithm 3, most notably the key derivation and counter management, to completely take place inside an HSM. In this setup a calling application sends i and N_E to the HSM, which then calculates j in Line 4, determines K_j, then executes Lines 4-10 of Algorithm 3 returning the certificate. The calling application can also send its request in bulk, i.e. a range of i in one call, thereby reducing communicational overhead. Compare for instance [30] where a number of 2.400 ECDSA-384 signatures per second can be achieved this way. This indicates that the IFAL certificate file can be produced in about 3 minutes using an HSM.

Prior to an epoch, or even during the epoch, the OBE could reconstruct the required IFAL certificates. Note that the AA certificate signatures allow the authenticity and integrity of the reconstructed certificates to be checked.

5.2 Distribution of IFAL activation codes

In IFAL the EA needs to periodically distribute IFAL activation codes to vehicle OBEs, containing the epoch keys required to use the certificates in an epoch. The confidentiality of these keys is protected by the IFAL transport key that is included in the IFAL certificate file. The authenticity and integrity of these keys are implicity protected by validation of the reconstructed certificates. That is, IFAL activation codes can be distributed over unprotected channels. In practice we suggest to use 128 bit epoch keys and to let an IFAL activation code take the form of an one block AES encryption using the transport key. That is, the cryptographic core of an IFAL activation code is of size 128 bit, i.e. 16 bytes. For robustness we could add five bytes to identify the epoch and the IFAL certificate file is more than one file was issued to the OBE in the past. This could be represented as a “printable” code of 42 hexadecimal characters or 28 BASE64 characters. In essence there are two ways to let the EA distribute IFAL activation codes to vehicle OBEs: point-to-point or through a broadcasting channel.

In point-to-point, the EA has the possibility to directly send information to the vehicle's OBE. This is trivial to arrange if the vehicle is internet connected. However, this will typically only be the case for very modern, high-end vehicles. Also internet connectivity is not supported in all regions. A more practical point-to-point setup is the use of the Short Message Service (SMS). SMS is already commonly in mobility and modern vehicles are already issued with mobile phone Subscriber Identity Module (SIM) for this. Actually, starting from April 2018 European vehicles need to be equipped with an eCall device equipped with SIM. This device will bring rapid assistance to motorists involved in a collision anywhere in the European Union. SMS allows sending messages up to 160 alphanumeric characters. That is, one SMS message fits to send an IFAL activation code leaving 132 characters for additional information. The IFAL activation codes could be automatically sent through SMS by the EA to vehicles. To handle transmission errors, the OBE should be able to request a resend of the code by sending an SMS to the EA. One can also make use of the road side equipment in a point-to-point fashion. This is equipment along the road that can wirelessly communicate with OBEs, e.g. using Dedicated short-range communications (DSRC). Compare [9]. In this setup the OBE requests the EA for new activation codes through the road-side equipment. These codes are then also delivered to the OBE trough the road side equipment. As a backup mechanism, and not very user friendly, a 28 character IFAL activation code can also be manually entered in the OBE by its owner, e.g. typed in or through an SD card. In this setup the codes would be periodically sent to the vehicle owner by the EA, e.g. by email.

In principle one can also use a broadcast channel to send IFAL activation codes. This setup is hampered in practice by the situation that OBEs are typically only active when the vehicle is. That is, without additional timing mechanism it seems difficult to ensure activation codes are received by the vehicle. One might deploy OBEs that are partly active when the vehicle is off. Also hybrid setups might be feasible. Here the vehicle drives triggers the broadcasting of the activation code or is informed that his activation code will be broadcasted in a certain period. We further discuss the broadcast possibility mainly for academic reasons. In this situation the IFAL activation code needs to be supplemented with an OBE identifier. With this identifier the OBE would be able to identify the IFAL activation code meant for it. This identifier is similar with the International Mobile Equipment Identity (IMEI) used in mobil communication, which uses 15 digit numbers, i.e. less than 6 bytes. If we consider a 7 byte identifier, the total size of a broadcast IFAL code is 27 byte fitting in 36 printable characters (using BASE64 encoding). Also, as vehicles will not always be listing to the broadcasting channel the codes would need to be sent several times. Broadcasting could be facilitated by road side equipment. In principle one could also use the Radio Data System (RDS) [25] that allows embedding small amounts of digital information in conventional FM radio broadcasts. RDS is already used in mobility to distribute traffic information. Current RDS has a bandwidth of about 700 bit/second, its successor RDS has a bandwidth of about 3.500 bit/second, cf. [26]. This means that it takes about 3,6 days to broadcast a million IFAL codes using current RDS. By using RDS2 this would be reduced to 17,1 hours. Current analog radio is transforming to digital radio or Digital Audio Broadcasting (DAB) which also supports a variant of RDS (RT+). See [33]. DAB is typically based on a bandwidth of 128 kb/s implying that one could broadcast a million IFAL activation keys in less than half an hour using the full DAB potential. DAB also supports the Transport Protocol Experts Group (TPEG) protocol for broadcasting information over various delivery channels including DAB and radio.

5.3 Management of the epoch keys

In Section 4.3 we let the AA explicitly generate the epoch keys $K_0, K_1, ..., K_{N_E-1}$. In a practical implementation it is more convenient to simply generate one master key M_{Id_O} for each Certificate File and let the epoch keys be derived from this masterkey and the epoch number. For this, one uses the $\mathcal{K}_2$ function specified in Section 4.2. Actually, one could also generate one AA wide master key M and let each master key M_{Id_O} be derived from M and Id_O.

6 Comparison with requirements and IFAL extensions

In this section we compare IFAL with each of the requirements from Section 2 and discuss related extensions.

6.1 Comparing IFAL with trust requirements

Trustworthy ITS message signing In IFAL the TE can be kept oblivious of the use of pseudonymous certificates. Also, as indicated in Section 4.3, IFAL vehicle signatures solely rely on the TE trustworthiness. The IFAL secrets the OBE manages do not enable it to generate signatures without the TE. In other words, the TE trust role is the same as in a basic setup with only one static vehicle certificate. Of course one can also implement the IFAL cryptographic logic of the OBE inside the TE. The security benefit for this would be limited as it is still the OBE that forms the ITS messages to be signed. In other words, the OBE plays a crucial ITS security role anyway.

Sybil attack resistance By design IFAL has resistance against Sybil attacks; at most two IFAL certificates can be valid simultaneously and only for a small, configurable time.

PKI removal of misbehaving vehicles Conformity of this requirement is also by design and described in Section 4.6.

Law Enforcement support (Optional) In Section 4.6 we described how the AA can determine the vehicle's identifier Id_O for deactivation purposes. However, this mechanism can be optionally supplemented with an interaction with the EA such that the identity of the vehicle can be determined and provided to law enforcement in a controlled fashion. This functionality can be placed best at a separate authority, compare the discussion on the "Pseudonym unlinkability by EA and AA technically enforced" requirement below.

6.2 Comparing IFAL with privacy requirements

Pseudonym unlinkability The vehicle public keys in IFAL certificates are unlinkable as they are based on different private keys. The linking risk related to the lifetime of certificates is configurable by the choice of IFAL certificates validity period. The amount of indirectly identifying information in an IFAL certificate is an implementation choice and independent of IFAL. Implementing the IFAL cryptographic logic of the OBE inside the TE does not enhance privacy protection. Indeed, the OBE plays an fundamental privacy role anyway: a malicious OBE could simply broadcast the vehicle identity.

Pseudonym unlinkability at EA and AA By design the EA the cannot link IFAL certificates as these are directly sent to the vehicle/OBE by the AA. Also as the EA has no access to the epoch keys he is unable to reconstruct the IFAL certificates. Technically, the AA is able to link the IFAL certificates as he forms the IFAL certificate file. We already reduced this risk by requiring in Section 3.2 that the AA deletes the certificates and the vehicle base public key after issuing. Also, as part of the de-authorization process the EA is able to link an IFAL certificate to the vehicle's identifier Id_O and is thus also able to link them. Of course, procedures need to be in place at the AA controlling these capabilities. However, this is all based on procedures and not technically enforced.

Pseudonym unlinkability at EA and AA technically enforced (Optional) Technical enforcement of unlinkability at EA and AA is also possible by deploying a Hardware Security Module (HSM) at the AA. This was already discussed at the end of Section 5.2. In this setup the HSM then additionally encrypts the certificates using the key P_E generated by the OBE before providing them to the calling application. That is, after Line 10 in Algorithm 2 a Line 11 is added for this. In this fashion AA staff will not have access to the IFAL certificates in plaintext. Moreover, the HSM will not give AA staff access to the recovery operation, i.e. decrypting the vehicle's identifier Id_O from the certificate. For this another party, e.g. the *Misbehaving Authority (MA)* from [32], will be setup with its own HSM that will allow the recovery operation in a controlled fashion. That is, De-Authorization is then under dual control of AA and MA whereas law enforcement support is under tripartite control of AA, MA and EA. The HSM development and deployment will be under control of an independent trusted party.

6.3 Comparing IFAL with usability requirements

ETSI conformity IFAL follows the ETSI approach [11] with the exception that it allows for the issuance of certificate batches instead of only single certificates.

Certificate pre-install By design IFAL allows for certificate pre-install.

No CRL distribution By design IFAL does not make use of CRLs, instead it uses de-authorization.

Low vehicle connectivity requirements IFAL poses minimal vehicle connectivity requirements, cf. Section 5.2. In fact, IFAL does not require any connectivity requirement as in principle activation codes can be entered manually by the vehicle holder.

TE Simplicity By design the TE can be obvious of IFAL allowing basic TEs in IFAL.

Low certificate OBE storage IFAL imposes only minimal requirements on OBE storage as it allows certificates to be reconstructed in the OBE. See Section 5.1

7 Related work

In [32] the Butterfly scheme is presented. Like IFAL this scheme is also based on a key derivation function allowing a vehicle to derive many secret keys on only one base private key. Actually, we discussed the Butterfly derivation function in Section 4.2, cf. Formula (4). Unlike IFAL the Butterfly scheme is based on the conventional setup where a vehicle periodically needs to run a certificate issuing protocol and to download CRLs. This imposes significant communicational requirements on the vehicle which are avoided in IFAL. Handling of misbehaving vehicles and placing them on a CRL in Butterfly is rather complex as it involves five parties: the RA, PCA, two Linking Authorities and a Misbehaviour Authority. Butterfly certificates also contain an 8 byte linking value for this which is avoided in IFAL. Another difference with IFAL is that in the Butterfly scheme the TE cannot be oblivious of the key expansion function. That is, the TE needs to form the private key corresponding to the Butterfly certificate based on the base private key and the so-called "private key reconstruction value". Although this is a simple operation in principle, effectively an addition modulo the group order q, TEs need to support this operation. The form of this operation is due to the usage of the Butterfly key derivation function which is based on modular addition. By contrast, IFAL key derivation is based on modular multiplication which allows the TE to be oblivious of the key derivation. See Algorithm 2.

An advantage of using the Butterfly addition based key derivation is that it allows the use of so-called Implicit Certificates [1,27] instead of ECDSA based signatures. With implicit certificates a relying party is provided a "reconstruction value" P from a user together with its identity U normally inside a certificate. Based on P, U and the public key of the certification authority a relying party can then compute the user public key. If the user can prove possession of the corresponding private key, this also proves authenticity of the public key and the identity U. An advantage of implicit certificates is that the reconstruction value P is only half the size of an ECDSA signature, i.e. where such a signature is 256 bits the reconstruction value is only 128 bits. Due to the multiplication based key derivation in IFAL we cannot use implicit certificates. However, in IFAL we actually use the extra length of the ECDSA signatures for de-pseudonymization purposes allowing the AA to deactivate misbehaving vehicles based on their pseudonymous certificate. It seems this is not possible using implicit certificates. Apart from this, and the fact that implicit certificates is patented technology, we have two technical objections against the use of implicit certificates in ITS. First of all, it is not proven that using implicit certificates in combination with ECDSA is secure. This concern is explicitly stated by the inventors of implicit certificates themselves in the original paper [1]. In fact, an attack on a very similar scheme is known to exist, cf.[2]. Our second objection is a legal nature. The basic security objective of a certificate is strongly binding a public key to a user whereby supporting non-repudiation. That is, a signature on a document (ITS message in our context) should unrefutable link back to the user, i.e. to its registration. This basic property is not met by implicit certificates when used with pseudonymous certificates. In the appendix we sketch a context where a implicit certificate based signature links back to two users each of which can plausibly deny having signed the message.

In the basic IFAL scheme the AA is technically able to link certificates issued to the same vehicle. By using HSMs we have indicated in Section 6.2 that the AA is no longer able to link certificates. The Butterfly scheme states that its AA (called PCA) is not able to link certificates. To this end, the Butterfly EA actually mixes the certificate requests of vehicles before they are sent to the AA. In IFAL is it essential that the certificates of one vehicle are generated in one batch. This implies that mixing is not possible in IFAL. However, one can argue how effective the Butterfly mixing is against a (malicious) AA wanting to link certificates. As the AA forms the certificates, it will be able to assess *when* a Butterfly certificate was produced. Typically, the vehicles using the Butterfly certificates will be scattered over the country. Suppose that a malicious AA is eavesdropping on ITS certificates on various places. If it notices two certificates being used physically close to each other that were produced on the same day, they probably belong to the same vehicle. Perhaps this issue can be resolved by various additional controls, e.g. by mixing of many days. However, the Butterfly paper does not mention this issue and is lacking statistical analysis on it.

8 Conclusion

We have specified Issue First Activate Later (IFAL) as introduced in [31] and shown how it allows for flexible IFAL policies, trade-offs between three essential V2X properties: trust, privacy and usability. IFAL is not based on certificate revocation lists and imposes minimal requirements on the connectivity of the vehicle and its trusted element. This allows IFAL to be easily and widely deployable in ITS.

References

1. D. Brown, R. Gallant and S. Vanstone. Provably secure implicit certificate schemes. FC 2001, LNCS 2339, pp. 156165. Springer, Feb. 2001.

2. D. Brown, M. Campagna and S. Vanstone. Security of ECQV-certified ECDSA against passive adversaries. Cryptology ePrint Archive Report 2009/620, IACR, 2009.
3. Bundesamt für Sicherheit in der Informationstechnik, Group Signatures: Authentication with Privacy, 2012.
4. Bundesamt für Sicherheit in der Informationstechnik, Elliptic Curve Cryptography, TR-03111, 2012.
5. ECC Brainpool, ECC Brainpool standard curves and curve generation, October 2005. See http://www.ecc-brainpool.org.
6. ETSI, TS 102 636-3 V1.1.1 (2010-03), Intelligent Transport Systems (ITS); Vehicular Communications;GeoNetworking; Part 3: Network architecture.
7. ETSI, TR 102 893 V1.1.1 (2010-03) Intelligent Transport Systems (ITS); Security; Threat, Vulnerability and Risk Analysis (TVRA).
8. ETSI, EN 319 411-2 V1.1.1 (2013-01), Electronic Signatures and Infrastructures (ESI); Policy and security requirements for Trust Service Providers issuing certificates; Part 1: General requirement
9. ETSI, EN 300 674-2-2 V2.1.1 (2016-11), Transport and Traffic Telematics (TTT); Dedicated Short Range Communication (DSRC).
10. ETSI, TS 102 940 V1.2.1 (2016-11), Intelligent Transport Systems (ITS); Security; ITS communications security architecture and security management.
11. ETSI, TS 102 941 V1.1.1 (2012-06), Intelligent Transport Systems (ITS); Security; Trust and Privacy Management.
12. ETSI, TS 103 097 V1.2.1 (2015-06), Intelligent Transport Systems (ITS); Security; Security header and certificate formats.
13. ETSI, EN 302 637-2 V1.3.2 (2014-11), Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service.
14. ETSI, EN 302 637-3 V1.2.2 (2014-11) Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service.
15. ETSI, Advances in ITS security standards Public Workshop C2C-CC, ETS and HTG#6, Stockholm, 17 June 2015.
16. The European parliament and the Council, Regulation (EU) 2016/679 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), 27 April 2016.
17. International Civil Aviation Organization, Machine Readable Travel Documents, Part 12: Public Key Infrastructure for MRTDs, Seventh Edition, 2015.
18. D. Hankerson, A.J. Menezes, S. Vanstone , Guide to elliptic curve cryptography, Springer, 2003.
19. T. Pornin, Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA), Request for Comments: 6979, IETF, August 2013.
20. R. Moalla, B. Lonc, H.Labiod, N. Simoni, Risk Analysis of ITS Communication Architecture, 2012 Third International Conference on The Network of the Future, 2012.
21. National Institute of Standards and Technology (NIST), Advanced Encryption Standard (AES), FIPS 197, November 26, 2001.
22. National Institute for Standards and Technology, Digital signature standard, Federal Information Processing Standards Publication 186-2. 2000.

23. National Institute of Standards and Technology (NIST), Security requirements for cryptographic modules, FIPS 140-2, May 25, 2001.
24. National Institute for Standards and Technology, Recommendation for Key Derivation Using Pseudorandom Functions, Special Publication 800-108, 2009.
25. RDS Forum, see http://www.rds.org.uk.
26. Personal communication with RDS forum.
27. SEC, Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV), SEC Std. 4, 2011.
28. U.S. Department of Transportation - Research and Innovative Technology Administration. Vehicle-to-Vehicle (V2V) Communications for Safety. Available from: www.its.dot.gov/research/v2v.htm
29. C. P. Schnorr, Efficient signature generation for smart cards. Journal of Cryptology, 4(3):239-252, 1991.
30. See https://hsm.utimaco.com/cryptoserver/securityserver-se-gen2/.
31. E. Verheul, Applicability of ABC4Trust in ITS, May 10, 2016. See http://rondetafels.ditcm.eu/sites/default/files/Security_20160510_ABC4Trust.pdf
32. W. Whyte, A. Weimerskircht, V. Kumar, T. Hehn, A Security Credential Management System for V2V Communications, 2013 IEEE Vehicular Networking Conference, 2013, pp. 1-8.
33. WorldDAB, see http://www.worlddab.org.

A Deniability of signatures based on implicit certificates

We first briefly explain the working of implicit certificates following [27] using the notation from Section 4.2. The public key Q_{CA} of the Certificate Authority (CA) is of the form $d_{CA}G$ where $d_{CA} \in [1, q-1]$ is the CA's private key. The user generates its *original* public key $R_U = d_U G$ where $d_U \in [1, q-1]$ is its private key. The user sends R_U together with its identity U to the CA. The user shows the CA that he possesses private key d_U and that its identity U is correct. If this is the case, the CA generates the *reconstruction data* $P_U = R_U + kG$ and the *internal signature* $r = e \cdot k + d_C A \bmod q$. Here $e = H(P_U, U)$ for some hash function $H(.)$ and k is chosen randomly from $[1, q-1]$ by the CA. Both the reconstruction data and the internal signature are provided to the user by the CA. Based on these, the user's public key Q_U now takes the form $e \cdot P + Q_{CA}$ and the implicit signature actually gives the user access to the private key which is of the form $r + e \cdot k_U$. The crucial observation is that the information P_U, U and r provided by the CA are not irrefutably associated to the original user public key R_U. Indeed, one can easily verify that for any $j \in [1, q-1]$ the reconstruction value $P' = P$ and internal signature $r' = r - e \cdot j \bmod q$ link to another original public key $R' = R_U + jG = (d_U + j)G$.

Seemingly this observation does not lead to any issues when the implicit certificate is coupled to a real identity U, e.g. as validated by an identity document as part of the registration phase. However, in the case of ITS the identity U only consists of non identifying information. Indeed, the implicit pseudonyms certificate should not be (indirectly) identifying. That is, the only link between the implicit certificate and the user is the original public key R_U sent by the user to the Certificate Authority (AA in our ITS context). Now from the observation, the implicit certificate can be linked to different original public keys if users conspire. Indeed, two users can share d_U, choose a random $j \in [1, q-1]$ and both register the original public key R_U and $R' = R_U + jG$ at the CA. This would then lead to two (different) internal signatures r, r' and reconstruction data P_U, P' that could be used by both users. In this situation it can never be decided which of the two users actually used the implicit certificate. To further illustrate, suppose that one of the users has signed a message with Q_U and wants to deny this later. To settle the dispute in court, the CA provides the original public key R_U, Q_U, P_U, U and r to the judge. Now the CA cannot claim that this irrefutably binds the implicit public key Q_U to the original user public key R_U. Indeed, by the above observation the user can disprove this claim. For this he provides R', Q, P, U and r' to the judge and states that the implicit public key Q_U actually binds to another original public key, namely public key R' of the other user. Although the judge might decide that the two users are conspiring, he is not able to decide which of the two users actually signed the message. Regardless of the judicial system, this is a situation one does not want to can occur. This scenario can evidently be extended to the situation where a large group of users conspire allowing deniability of having signed a message. We have effectively devised a group signature scheme based on implicit certificates without the possibility of assessing the actual signer in the group. Compare [3].