Zone Encryption with Anonymous Authentication for V2V Communication

Jan Camenisch[1], Manu Drijvers[1], Anja Lehmann[2], Gregory Neven[1] and Patrick Towa[2,3]

[1] DFINITY*
[2] IBM Research – Zurich
[3] ENS and PSL Research University

Abstract. Vehicle-to-vehicle (V2V) communication systems are currently being prepared for real-world deployment, but they face strong opposition over privacy concerns. Position beacon messages are the main culprit, being broadcast in cleartext and pseudonymously signed up to 10 times per second. So far, no practical solutions have been proposed to encrypt or anonymously authenticate V2V messages. We propose two cryptographic innovations that enhance the privacy of V2V communication. As a core contribution, we introduce zone-encryption schemes, where vehicles generate and authentically distribute encryption keys associated to static geographic zones close to their location. Zone encryption provides security against eavesdropping, and, combined with a suitable anonymous authentication scheme, ensures that messages can only be sent by genuine vehicles, while adding only 224 Bytes of cryptographic overhead to each message. Our second contribution is an authentication mechanism fine-tuned to the needs of V2V which allows vehicles to authentically distribute keys, and is called dynamic group signatures with attributes. Our instantiation features unlimited locally generated pseudonyms, negligible credential download-and-storage costs, identity recovery by a trusted authority, and compact signatures of 216 Bytes at a 131-bit security level.

1 Introduction

The automotive industry and several governments around the world have made substantial progress towards deploying Cooperative Intelligent Transport Systems (C-ITSs), with the first deployment planned to start in 2019 [52]. In C-ITSs, vehicles communicate with other vehicles (V2V) and with road-side infrastructure (V2I) to improve traffic safety and efficiency.

V2V and V2I communication, together often referred to as V2X, mainly consists of two types of messages: occasional event-triggered safety messages (e.g., emergency braking maneuver) and regular position beacon messages that each vehicle typically broadcasts 1–10 times per second. The latter category, known in the European C-ITS as Cooperative Awareness Messages (CAMs) [25]

* Work partially done while the corresponding authors were at IBM Research – Zurich.

and in the US as Basic Safety Messages (BSMs) [42], carry dynamic information about the vehicle such as its position, speed, and heading, as well as (semi-)static information about the vehicle such as its length, width, and sensor accuracy.

Unencrypted Broadcast Messages. The CAMs are primarily broadcast in plaintext over an unprotected short-range radio channel (ETSI ITS-G5), and these messages are therefore easy to intercept and potentially leak sensitive information about people's whereabouts, travel itineraries, and driving habits. The current C-ITS proposals [25, 42] have therefore raised serious privacy concerns among civil right unions, scientists [46], and data protection authorities [1]. Concrete threats include burglars tracking which houses are left unoccupied in a neighborhood, stalkers following their victims from an out-of-sight location, and mass surveillance of entire cities (e.g., through connected road infrastructure) at an estimated amortized cost of dollar-cents per vehicle per year [46]. Privacy regulations prohibiting misuse of CAM data are hard to enforce because rogue eavesdropping devices are easy to build and nearly impossible to detect or localize.

Due to the open nature of C-ITSs and the problems of managing encryption keys among constantly changing groups of vehicles, encryption in V2X has mostly been considered impractical and of little use (see Section 1.2 for a more detailed discussion).

Privacy-Preserving Authentication. Most research in V2X security and privacy has focused on the authentication aspect, ensuring that messages originate from genuine vehicles without making individual vehicles traceable throughout the system. The work of Petit et al. [43] provides an excellent survey of this field.

The practical C-ITS systems which are currently considered for deployment in Europe [26] and the US [42] take a similar approach to authentication by letting vehicles sign outgoing V2X messages with short-lived pseudonym certificates. Some degree of privacy is obtained by letting vehicles frequently change or rotate their certificates from a small pool of pseudonyms. In the European approach, vehicles periodically reload unrevocable pseudonyms from an online authorization authority. In the American approach, vehicles come preloaded with three years' worth of revocable pseudonym certificates [55].

However, both approaches are forced into an uncomfortable trade-off between security, privacy, and efficiency. A larger pseudonym pool size gives more privacy, but is expensive to store or download, and provides less protection against Sybil attacks in which the keys of a single compromised vehicle are used to simultaneously impersonate several vehicles. Indeed, the compromises of 100 pseudonyms per vehicle per week (EU) or 20 (US) essentially combine poor privacy guarantees (especially for frequent drivers) with high bandwidth-and-storage costs, and no meaningful Sybil resistance.

Solutions that realize anonymous authentication via group signatures [10, 56], or privacy-preserving credentials [20, 41, 51] provide stronger security and privacy guarantees. However, none of them fits the *stringent bandwidth constraint* of 300 Bytes per CAM, and they are therefore not suitable for practical deployment.

No Privacy Without Encryption. Finally, even though so much effort has been spent on privacy-preserving authentication in V2X, the main privacy problem in CAMs is actually the transmitted data itself. Indeed, when vehicles broadcast their position, speed, heading, and acceleration up to 10 times per second, then linking messages sent by the same vehicle is trivial simply by physical limitations, regardless of how often vehicles switch pseudonyms. The largely static information included in the CAMs such as the dimensions of a vehicle and its sensor accuracy further facilitates fingerprinting vehicles and tracking them over longer periods of time.

1.1 Our Contributions

We tackle the problem of privacy in V2X communication by addressing, for the first time, the problem of authenticity and confidentiality in combination. As a result we present a protocol to encrypt and authenticate CAMs that is suitable for the stringent 300 Bytes bandwidth requirement of C-ITSs, and arguably offers better privacy than the existing proposal. Namely, we propose an authentication scheme based on compact group signatures that combines unlimited privacy with negligible bandwidth-and-storage costs, and based on this authentication mechanism, we give a practical way to encrypt CAMs to hide their content from eavesdroppers. Interestingly, this combination not only improves the security and privacy of C-ITSs, but the careful composition of symmetric and asymmetric building blocks even leads to better efficiency.

Zone Encryption. We introduce the novel concept of *zone encryption* as a practical means to transmit V2X data authentically and confidentially. The core idea of zone encryption is to rely on *symmetric authenticated* encryption for protecting V2X communication, using temporary keys that are exchanged among vehicles in the same vicinity. Only the key-exchange messages are signed with short-lived certificates, which results in an important efficiency gain compared to the existing solutions which sign *every* outgoing CAM. For short-lived certificates, we use a new type of group signatures tailored to the need of V2X communication. Relying on group signatures instead of a pre-fetched batch of pseudonym certificates overcomes the trade-off between privacy and security of the existing approaches: vehicles only need to store a single credential, but have full privacy that is equivalent to an unlimited pseudonym pool size. We formally define the desired security and privacy properties and propose an efficient, provably secure protocol that achieves them. Each CAM is 240 Bytes long in our instantiation, which is compliant with the stringent bandwidth requirements of C-ITSs.

Zone encryption certainly does not solve all privacy issues concerning V2X communication, but it does raise the bar of eavesdropping on CAM data to a level that is unaffordable for occasional criminals and notably more expensive for mass surveillance. Private criminals will rely on a black market to obtain eavesdropping devices; and to offset the costs of compromising hardware-protected vehicle keys, the market will most likely share the same long-term credentials

across many rogue devices. Once the police confiscates rogue devices, it can run in a controlled setting to trace and revoke their underlying long-term credentials, thereby disabling all devices in the field that use the same credentials. This will in turn increase the costs of producing such devices until they become too expensive for private criminals.

In the current plaintext-broadcasting C-ITS proposals, mass surveillance through sensitive antennas or traffic infrastructure is fairly cheap to deploy and hard to detect. Being inherently a semi-open system that enables all vehicles to communicate, mass surveillance of C-ITS data through a network of hidden or moving transmitters will always remain possible. Zone encryption cannot prevent this, but increases the cost of operating such a network.

Namely, central surveillance antennas have to send strong signals to engage in key exchanges in the observed zones, making it harder for them to covertly operate. A distributed network of less powerful relay stations, e.g., in driving vehicles or road infrastructure, is considerably more expensive to set up. Besides, most road infrastructure (e.g., traffic signs) has no need for privacy, nor to receive CAM data. Such infrastructure can thus be given a different type of credentials that enables it to broadcast unencrypted authenticated information, but not obtain zone keys. Any piece of infrastructure that is nevertheless caught engaging in zone-key exchanges would be considered suspect, and requires explanation from road operators.

Compact Group Signatures. An important building block of our zone encryption construction is a compact *dynamic group signature scheme with attributes*. It enables an authority to issue attribute-carrying membership credentials to users. In our application, users are vehicles and the certified attribute is a short time epoch during which the credential is valid. These credentials allow to create any number of unlinkable signatures to authenticate zone-key exchanges. Moreover, the authority can recover the identity of a sending user if needed. We also describe a variant of the scheme that distributes the power to trace users over multiple authorities.

We formally define the security properties of group signatures with attributes and propose a provably secure instantiation based on modified Pointcheval–Sanders (PS) signatures [45]. Our scheme has signatures of 216 Bytes on a Cocks–Pinch curve with a 131-bit security level.

1.2 Related Work

As priorly mentioned, the bulk of the literature in security and privacy of V2X communication focuses on anonymous authentication. Solutions have been proposed based on different concepts including MACs [18], digital signatures [9, 32, 36, 55], identity-based cryptography [37], group signatures [10, 56], and privacy-preserving attribute-based credentials [20, 41, 51]. The latter two come closest to our concept of dynamic group signatures with attributes, but all have signatures at least twice as long (for up-to-date security parameters) as our solution. A detailed overview is given in [43].

Some work has also been concerned with encryption for V2X communication. Encrypting CAMs seems an obvious choice, but doing so in a practical and useful way is not straightforward. The necessarily open nature of C-ITSs requires that all nearby vehicles can decrypt. Embedding the same symmetric key in all units is not feasible as no revocation is possible when the key gets compromised. Possibly better solutions such as multi-sender broadcast encryption [29] or public-key traitor tracing [12, 19] do not scale to a setting with hundreds of millions of vehicles. Another drawback of symmetric encryption, broadcast encryption, or traitor tracing used alone is that it is almost impossible to detect, let alone localize, a rogue wiretapping device which eavesdrops on the communication. Public-key encryption is in this respect since the receiving device has to make itself known to the senders so that these latter know which public key to encrypt to. However, bandwidth restrictions prohibit one-to-one connections, and the CAM length of around 300 Bytes is too short to include a separate ciphertext for each receiving vehicle.

A number of previous proposals let vehicles organize dynamically into groups according to their speed and driving direction, and establish a common key to encrypt communication [47, 53]. These schemes are not practical, however, since key management in highly dynamic groups of vehicles is intricate. For instance, it is not clear whether the protocol to join a new group is fast enough to give a timely warning in case of a head-on collision with a group member.

Freudiger et al. [30] proposed to use cryptographic "mix zones" where V2X-enabled vehicles briefly encrypt all communication under a key provided by a traffic-infrastructure beacon to switch pseudonyms in an unlinkable way. Zone encryption scheme could be seen as an extreme extension of that concept where the entire surface is covered in mix zones, but without relying on infrastructure support.

2 Preliminaries

This section introduces the cryptographic building blocks needed in our constructions.

2.1 Pairing Groups

Given a group $\mathbb{G}$ with neutral element $1_{\mathbb{G}}$, $\mathbb{G}^*$ denotes $\mathbb{G}\backslash\{1_{\mathbb{G}}\}$. An asymmetric pairing group consists of a tuple $\left(p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e\right)$ such that p is a prime number, $\mathbb{G}$, $\tilde{\mathbb{G}}$ and $\mathbb{G}_T$ are p-order groups, and such that $e\colon \mathbb{G} \times \tilde{\mathbb{G}} \to \mathbb{G}_T$ is a pairing, i.e., an efficiently computable non-degenerate ($e \neq 1_{\mathbb{G}_T}$) bilinear map. Type-3 pairing group are pairing groups for which there is no known efficiently computable homomorphism from $\tilde{\mathbb{G}}$ to $\mathbb{G}$.

2.2 Hardness Assumptions

This section introduces the hardness assumptions on which our constructions rely.

Definition 1 (SDL Assumption [8]). *Let* G *be a type-3 pairing-group generator. The Symmetric Discrete-Logarithm (SDL) assumption over* G *is that for all* $\lambda \in \mathbb{N}$*, for all* $\Gamma = \left(p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e\right) \leftarrow \mathsf{G}\left(1^\lambda\right)$*,* $g \in_R G^*$*,* $\tilde{g} \in_R \tilde{\mathbb{G}}^*$*,* $x \in_R \mathbb{Z}_p$*, given* $(\Gamma, g, \tilde{g}, g^x, \tilde{g}^x)$ *as an input, no efficient adversary can return* x *with a non-negligible probability.*

Pointcheval and Sanders introduced a new non-interactive q-type of assumption that they call the Modified q-Strong Diffie–Hellman assumption, and proved that it holds in the generic bilinear group model [45]. Note that it implies the SDL assumption.

Definition 2 (q-MSDH-1 Assumption [45]). *Let* G *be a type-3 pairing-group generator. The q-MSDH-1 assumption over* G *is that for all* $\lambda \in \mathbb{N}$*, for all* $\Gamma = \left(p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e\right) \leftarrow \mathsf{G}(1^\lambda)$*, given* Γ*,* $g \in_R \mathbb{G}^*$*,* $\tilde{g} \in_R \tilde{\mathbb{G}}^*$*, and two tuples* $\left(g^{x^l}, \tilde{g}^{x^l}\right)_{l=0}^{q} \in \left(\mathbb{G} \times \tilde{\mathbb{G}}\right)^{q+1}$ *and* $(g^a, \tilde{g}^a, \tilde{g}^{ax}) \in \mathbb{G} \times \tilde{\mathbb{G}}^2$ *for* $x, a \in_R \mathbb{Z}_p^*$*, no efficient adversary can return a tuple* $\left(w, P, h^{1/x+w},\ h^{a/P(x)}\right)$ *with* $h \in \mathbb{G}^*$*,* P *a polynomial in* $\mathbb{Z}_p[X]$ *of degree at most* q *and* $w \in \mathbb{Z}_p$ *such that the polynomials* $X + w$ *and* P *are coprime.*

2.3 Deterministic Authenticated Encryption

A Deterministic Authenticated Encryption (DAE) scheme [48] is a symmetric encryption scheme which supports auxiliary information or header. It guarantees two properties: privacy and authenticity. Privacy simply means that for uniformly random keys, the encryption of a new message is computationally indistinguishable from a uniformly random bit string. As for authenticity, it ensures that no efficient adversary can compute, with non-negligible probability, a valid ciphertext (i.e., for which decryption does not fail) without knowledge of the key. See Appendix A for formal definitions of these properties.

Formally, a DAE scheme is a tuple of algorithms $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$: a setup algorithm $\mathsf{Setup}(1^\lambda) \to pp$ which generates public parameters; a key-generation algorithm $\mathsf{KG}(pp) \to K$ which returns a key K chosen uniformly at random from a key space; an encryption algorithm $\mathsf{Enc}(K, H, M) \to C$ which takes as input a key K, a header H and a message M, and returns a ciphertext C; and a decryption algorithm $\mathsf{Dec}(K, H, C) \to \{M/\perp\}$.

2.4 Signatures

Given public parameters pp, a signature scheme consists of a key-generation algorithm $\mathsf{KG}(pp) \to (vk, sk)$, a signing algorithm $\mathsf{Sign}(sk, m) \to \sigma$, and a verification algorithm $\mathsf{Vf}(vk, m, \sigma) \to \{0, 1\}$.

Pointcheval–Sanders Signatures. Pointcheval and Sanders [45] introduced an efficient signature scheme that allows to sign message blocks $(m_1, \ldots, m_k)$ at once. As their signatures are randomizable and as the verification equation of their scheme does not involve any hash-function evaluation, one can efficiently prove in zero-knowledge knowledge of signatures.

Given a type-3 pairing-group generator G and a security parameter $\lambda \in \mathbb{N}$, the PS signature scheme in a pairing-group $\Gamma = \left(p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e\right) \leftarrow \mathsf{G}\left(1^\lambda\right)$ consists of the following algorithms.

$\mathsf{PS.KG}(\Gamma, k) \rightarrow (vk, sk)$: Generate $\tilde{g} \in_R \tilde{\mathbb{G}}^*$, x, $y_1, \ldots, y_{k+1} \in_R \mathbb{Z}_p$, compute $\tilde{X} \leftarrow \tilde{g}^x$, $\tilde{Y}_j \leftarrow \tilde{g}^{y_j}$ for $j \in [k+1]$. Set and return $vk \leftarrow \left(\tilde{g}, \tilde{X}, \tilde{Y}_1, \ldots, \tilde{Y}_{k+1}\right)$ and $sk \leftarrow (x, y_1, \ldots, y_{k+1})$.

$\mathsf{PS.Sign}\,(sk, (m_1, \ldots, m_k)) \rightarrow \sigma$: Generate $h \in_R \mathbb{G}^*$, $m' \in_R \mathbb{Z}_p$, and return $\sigma \leftarrow \left(m', h, h^{x+\sum_{j=1}^{k} y_j m_j + y_{k+1} m'}\right)$.

$\mathsf{PS.Vf}\,(vk, (m_1, \ldots, m_k), \sigma) \rightarrow b$: Parse σ as (m', σ_1, σ_2), verify that $\sigma_1 \neq 1_{\mathbb{G}}$ and that $e\left(\sigma_1, \tilde{X} \prod_{j=1}^{k} \tilde{Y}_j^{m_j} \tilde{Y}_{k+1}^{m'}\right) = e(\sigma_2, \tilde{g})$. If so, return 1, otherwise return 0.

Pointcheval and Sanders showed [45] that this signature scheme is existentially unforgeable under the q-MDSH-1 assumption (see Definition 2).

3 Group Signatures with Attributes

In this section, we introduce an important building block for our Zone-Encryption protocol, namely *dynamic group signatures with attributes* (DGS+A). We formally define DGS+A as an extension of conventional dynamic group signatures, and propose a secure and highly efficient instantiation from PS signatures.

3.1 Definition of DGS+A

Dynamic group signatures (DGS) [6] allow users to join a group of signers at any time, and then sign anonymously on behalf of the group. That is, a verifier is assured that a signature stems from a group member but learns nothing about the identity of the signer. Only the group manager (also called issuer) or a dedicated opening authority can recover the identity behind a valid signature.

In our DGS+A extension, users obtain membership credentials which are associated to a set of attributes by interacting with an issuer. Signatures, further referred to as *authentication tokens*, are verified w.r.t. to those attributes, i.e., a message m can only be signed for a set of attributes A if the signer has a valid membership credential for A. A similar generalization of group signatures with attributes was already introduced by Camenisch, Neven and Rückert [17], but without interactive credential issuance.

3.1.1 Syntax. Formally, a DGS+A scheme consists of

$\mathsf{Setup}(1^\lambda, aux) \to pp$: Generates public parameters on the input of a security parameter and of auxiliary inputs. These public parameters are assumed to be an implicit input to all the other algorithms.

$\mathsf{KG}(pp) \to (pk, (sk, st))$: A key-generation algorithm for the issuer. It is assumed that pk can be recovered from sk. Variable st represents a state.

$\langle \mathsf{Issue.U}(id, A, pk) \rightleftharpoons \mathsf{Issue.I}(sk, st, id, A) \rangle \to \langle cred, st' \rangle$: A credential-issuance protocol for an attribute set A and user identity id. At the end of the protocol, the user outputs a membership credential $cred$ (or $\perp$ if the protocol fails) and the issuer updates its state to st'. Credential $cred$ is assumed to contain the attributes A.

$\mathsf{Auth}(pk, cred, m) \to tok$: A probabilistic authentication algorithm which signs a message m w.r.t. A and returns tok.

$\mathsf{Vf}(pk, m, A, tok) \to b \in \{0, 1\}$: Returns $b = 1$ if tok is a valid token for message m and attributes A w.r.t. pk.

$\mathsf{Open}(sk, st, m, A, tok) \to id/\perp$: An opening algorithm which allows the issuer to identify the user who generated a valid authentication token. The algorithm returns an identity id or $\perp$.

3.1.2 Security Properties. A DGS+A scheme should satisfy correctness, anonymity and traceability. Our definitions follow the security notions of conventional dynamic group signatures, which we adapt to a setting with attributes. The formal definitions are given in Appendix B.1 and we sketch their intuitions below.

Correctness. Correctness captures the idea that a truthfully generated authentication token should be accepted by the verification algorithm. Moreover, if all the algorithms are honestly executed and the opening protocol is run on a token, then all the opening algorithm should output the identity of the user who computed the token. These properties should hold independently of the order in which credentials are issued for user–attribute pairs and with overwhelming probability.

Anonymity. Anonymity ensures that an authentication token reveals no information about the identity of the user who computed it if the issuer is honest and the token has not been opened. Note that user identities are not hidden during the credential issuance protocol, and in fact need to be revealed for the issuer to be able to open group signatures. That is, anonymity is only considered w.r.t. tokens.

Traceability. Traceability captures the expected unforgeability guarantees of our group signatures. It guarantees that as long as the issuer is honest, for any valid token tok^*, message m^* and attribute set A^*, opening can neither fail nor reveal an incorrect honest identity id. The latter means that the user id either never joined the group w.r.t. A^*, or has joined but never signed m^*.

3.2 Our DGS+A Scheme

The high-level idea of our DGS+A scheme is to compute a user membership credential as a PS signature on her identity and her (public) attributes. To compute an anonymous authentication token for a message, the user re-randomizes the group elements of her signature and computes a signature of knowledge, on the message, of her signed identity.

To allow for compact authentication tokens yet enable traceability, we follow the approach by Bichsel et al. [8] where the issuer maintains a list of the credentials that it generated and traces a token by testing the re-randomized PS signature in the token against each entry. This approach makes tracing more expensive for the benefit of having short tokens, which perfectly fits our application to V2V communication in which bandwidth is limited and tracing an uncommon practice.

Pointcheval and Sanders described a similar group signature scheme [44, Appendix A.1] based on the CT-RSA'16 version of their signature scheme. The security of their group-signature scheme thus relies on an interactive assumption. Our scheme is based on the modified PS signature scheme [45] which allows to prove traceability from a q-type assumption instead of an interactive one. Moreover, we add attributes to the membership credentials, as they are needed in our V2X scenario in which credentials are short-lived and periodically issued.

Scheme Description. Let G be a type-3 pairing-group generator, $\mathcal{H}$ a random oracle and PS the modified Pointcheval–Sanders signature scheme (Section 2.4). Denoting by k the number of attributes of each user, our DGS+A scheme DGSA is the following:

$\mathsf{Setup}(1^\lambda, k) \to pp$: Generate $\Gamma = \left(p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e\right) \leftarrow \mathsf{G}\left(1^\lambda\right)$. Return $pp \leftarrow (\Gamma, k+1)$.

$\mathsf{KG.I}(pp) \to (pk, (sk, st))$: Generate $\tilde{g} \in_R \tilde{\mathbb{G}}^*$, $(x, y_{id}, y_1, \ldots, y_{k+1}) \in_R \mathbb{Z}_p^{k+3}$, compute $\tilde{X} \leftarrow \tilde{g}^x$, $\tilde{Y}_{id} \leftarrow \tilde{g}^{y_{id}}$, and $\tilde{Y}_j \leftarrow \tilde{g}^{y_j}$ for $j = 1, \ldots, k+1$, and return $pk \leftarrow \left(\tilde{g}, \tilde{X}, \tilde{Y}_{id}, \tilde{Y}_1, \ldots, \tilde{Y}_{k+1}\right)$, $sk \leftarrow (pk, x, y_{id}, \ldots, y_{k+1})$ and an intially empty state $st \leftarrow \emptyset$.

Issue : For issuance between a user $\mathcal{U}$ and an issuer $\mathcal{I}$, we assume a secure channel. If a party aborts the protocol, it returns $\perp$. We further assume that the identity space ID is a polynomial-size (in λ) subset of $\mathbb{Z}_p$.

1. $\mathsf{Issue.I}\left(sk, st, id, A = (a_i)_{i=1}^k\right)$,
   - abort if a record $(id, A, *)$ exists in st
   - compute $\sigma = (a', \sigma_1, \sigma_2) \leftarrow \mathsf{PS.Sign}(sk, (id, a_1, \ldots, a_k))$
   - send σ to $\mathcal{U}$ and return $st' \leftarrow st \cup (id, A, a')$
2. $\mathsf{Issue.U}(id, A, pk)$ upon receiving σ from $\mathcal{I}$:
   - verify that $\mathsf{PS.Vf}(pk, (id, A), \sigma) = 1$ and abort if not
   - return $cred \leftarrow \left(id, A, \sigma, e\left(\sigma_1, \tilde{Y}_{id}\right), e\left(\sigma_1, \tilde{Y}_{k+1}\right)\right)$. It is actually not necessary to store $e(\sigma_1, \tilde{Y}_{id})$ and $e(\sigma_1, \tilde{Y}_{k+1})$, but it helps avoiding pairing computations when tokens are generated.

$\mathsf{Auth}(pk, cred, m) \to tok$: Parse $cred = (id,\ A,\ \sigma,\ e\left(\sigma_1, \tilde{Y}_{id}\right), e\left(\sigma_1, \tilde{Y}_{k+1}\right))$ with $A = (a_i)_{i=1}^k$, generate $r \in_R \mathbb{Z}_p^*$, compute $(\sigma_1', \sigma_2') \leftarrow (\sigma_1^r, \sigma_2^r)$ and a non-interactive proof of knowledge π of (id, a') such that

$$e\left(\sigma_1', \tilde{X}\tilde{Y}_{id}^{id}\prod_{i=1}^{k}\tilde{Y}_i^{a_i}\tilde{Y}_{k+1}^{a'}\right) = e(\sigma_2', \tilde{g}).$$

That is, compute $u \leftarrow e\left(\sigma_1^{rs_{id}}, \tilde{Y}_{id}\right) e\left(\sigma_1^{rs_{a'}}, \tilde{Y}_{k+1}\right)$ for $s_{id}, s_{a'} \in_R \mathbb{Z}_p$, compute a challenge $c \leftarrow \mathcal{H}(u, A, m, \sigma_1', \sigma_2', pk) \in \mathbb{Z}_p$ and a response $\mathbf{v} \leftarrow (s_{id} - cid, s_{a'} - ca')$. Set $\pi \leftarrow (c, \mathbf{v})$, and return $tok \leftarrow (\sigma_1', \sigma_2', \pi)$.

$\mathsf{Vf}(pk, m, A, tok) \to b$: Parse $tok = (\sigma_1, \sigma_2, \pi)$ with $\pi = (c, v_{id}, v_{a'})$, $A = (a_i)_{i=1}^k$, and return 1 if $\sigma_1 \neq 1_{\mathbb{G}}$ and $c = \mathcal{H}_0(u, A, m, \sigma_1, \sigma_2, pk)$ for $u \leftarrow e\left(\sigma_1^{v_{id}}, \tilde{Y}_{id}\right) e\left(\sigma_1^{v_{a'}}, \tilde{Y}_{k+1}\right) e(\sigma_2^c, \tilde{g})\, e\left(\sigma_1^c, \tilde{X}^{-1}\prod_{j=1}^{k}\tilde{Y}_j^{-a_j}\right)$.

$\mathsf{Open}(sk, st, m, A, tok) \to id/\perp$: Recovers the identity id of the user who generated an authentication token $tok = (\sigma_1, \sigma_2, \pi)$ for a message m and attribute set A. It first verifies that tok is valid for m and A. If so, it goes through (in lexicographic order of the identities) the tuples (id, A, a') in st until it finds one such that (a', σ_1, σ_2) is a valid PS signature on (id, A), and then returns id. If no such tuple is found, it returns $\perp$.

Algorithm 1 Open.

Require: (sk, st, m, A, tok)
Ensure: An identity id or $\perp$.
1: **if** $\mathsf{Vf}(pk, m, A, tok) = 0$ **then**
2: **return** $\perp$
3: **end if**
4: **for all** id such that $(id, A, a') \in st$ **do**
5: **if** $e\left(\sigma_1, \tilde{X}\tilde{Y}_{id}^{id}\prod_{i=1}^{k}\tilde{Y}_i^{a_i}\tilde{Y}_{k+1}^{a'}\right) = e(\sigma_2, \tilde{g})$ **then**
6: **return** id
7: **end if**
8: **end for**
9: **return** $\perp$

To open a signature on a given message, the opening algorithm loops over all id such that a credential was issued for a tuple (id, A). The complexity of the opening algorithm is then of order $O(|ID|)$. This approach allows to have much shorter group signatures than those obtained with the traditional sign-and-encrypt paradigm. An expensive opening procedure seems appropriate to the case of V2X communication, the target application of this paper, as the issuer should revoke the anonymity of vehicles only on solid grounds.

Correctness & Security. The proofs that scheme DGSA satisfies correctness and the security properties stated in Section 3.1 are deferred to Appendix B.2.1. We here simply state the theorems.

Theorem 1 (Correctness). DGSA *is correct.*

Theorem 2 (Anonymity). *In the random oracle model,* DGSA *satisfies anonymity if both the first-group DDH and the SDL assumptions holds over the group generator* G*.*

Theorem 3 (Traceability). *Denoting by q the amount of queries to oracles* Issue *and* Issue.I*, scheme* DGSA *satisfies traceability under the q-MSDH-1 assumption (which implies the SDL assumption) over the group generator* G *in the random oracle model.*

3.2.1 Efficiency. With a Cocks–Pinch pairing curve [34] defined over a field of order 2^{544} and with embedding degree 8, group elements in $\mathbb{G}$ take 68 Bytes for a group of 256-bit order. Note that this curve provides 131 bits of security [34].

An authentication token consists of two $\mathbb{G}$ elements and three $\mathbb{Z}_p$ elements, totalling 232 Bytes. The hash value in the proof of knowledge of a multi-signature can actually be shortened to second-preimage resistant length, further shortening a group signature to 216 Bytes.

Application to Zone Encryption. With a token size of 216 Bytes, our pairing-based instantiation is sufficiently compact to be used in combination with our zone-encryption scheme. Therein, tokens are only computed and sent during key requests and responses. Compared to the 160-Byte overhead of ECDSA signatures with certificates, our scheme could even be considered to sign each individual CAM.

3.2.2 DGS+A with Threshold Opening. In the above definition and scheme, the issuer can alone open all tokens, which makes him a single point of failure in terms of privacy. In some applications, including zone encryption, one may want to distribute the authority to open tokens over a group of n authorities, so that at least a threshold $\tau + 1$ of them must collaborate to open a token. In Appendix C, using threshold cryptography [21,50], we show how to do so with a slight modification of the above scheme.

4 Zone Encryption

This section introduces a novel mechanism to authentically and confidentially send CAMs between vehicles, and called *zone encryption*. It lets a vehicle securely communicates with the other vehicles in its vicinity, encrypting all CAMs. A vehicle can do so only after *anonymously authenticating* itself to the other vehicles.

To authenticate itself, a vehicle uses a short-term credential that it requests at regular intervals from an issuer to whom it authenticates with a long-term

credential. If necessary, the issuer can revoke the anonymity of a vehicle and potentially ban it from the system by revoking its long-term credential.

Overall, the goal is that the V2X communication is authenticated, confidential, i.e., only authorized vehicles can decrypt messages, and that the privacy of vehicles in this communication is preserved. We start by describing the high-level concept of zone encryption for V2X communication, then formally define the desired properties and finally propose a provably secure instantiation.

Geo-Local Shared Keys. The core idea behind zone encryption is to leverage the fact that only vehicles in close proximity need to communicate. More precisely, zone encryption assumes that the surface of the earth is divided into disjoint zones, and lets the vehicles that are present in a particular zone agree on the shared encryption key for that zone. For example, the zone boundaries could be derived statically from the GPS coordinates and are chosen so that the longest straight-line distance within a zone is less than the transmission radius of a radio signal (typically 300–500m), so that any two vehicles in the same zone should be able to communicate.

Of course, it should be avoided that two vehicles that are physically close but at opposite sides of a zone boundary cannot communicate because they broadcast to different zones. On this account, vehicles broadcast to multiple zones simultaneously.

Short-Lived Zone Keys. We also impose that zone keys are periodically refreshed, e.g., every 15 minutes. This ensures that a rogue eavesdropping device cannot simply stay silent and listen to ongoing traffic, but has to send key requests or responses to other vehicles, exposing itself to detection and localization through triangulation.

Authenticated encryption. Zone encryption takes a significantly different approach for authentication than the existing C-ITS proposals. Instead of signing every CAM with an anonymous authentication scheme, we simply use authenticated symmetric encryption with the short-lived zone keys. Anonymous credential-based authentication is only necessary when a vehicle enters a zone and keys are exchanged in an authenticated manner. Given that each vehicle has to process up to 3000 incoming CAMs per second, relying (mostly) on symmetric primitives instead of asymmetric authentication leads to a significant computational speed-up.

Besides, smart traffic infrastructure that has no need to receive CAMs can be equipped with certificates only for broadcasting authenticated but unencrypted messages (as their content is not privacy sensitive), so that it cannot be abused for mass surveillance.

Identity Resolution & Revocation. In case of dispute or malicious activity in a certain zone at a given time, the messages that each vehicle had to send to receive the zone key can be *opened* by a dedicated entity. The opening algorithm run by this entity reveals the identity vehicle that computed a message, which in turn allows to revoke its long-term credential. Recovering the long-term

identity of rogue vehicles is commonly known as *identity resolution*. It has been established as an essential requirement to balance the privacy and accountability needs in vehicular communication systems [43, 49, 55]. In the current C-ITS proposal, identity resolution is realized by keeping mappings between pseudonyms and long-term identities [55, Section IV.D].

For revocation, we follow the *passive revocation* approach advocated by the European standard [27, Section 6.1.4], meaning that vehicles must regularly request new short-term credentials. These requests which will be rejected once the corresponding long-term credential has been revoked. Revocation of the long-term credential does not only disable the decryption capabilities of the detected device, but also of any other rogue devices based on the same compromised credential, making mass production of rogue devices less lucrative.

Privacy & Efficiency vs. Sybil Resistance & Non-Repudiation. Zone encryption does pay a price in some other security aspects, though. By relying on symmetric authenticated encryption to authenticate CAMs, we achieve neither Sybil resistance nor non-repudiation. The former is not a major change since with a pseudonym pool size of up to 100 simultaneously valid certificates, the current proposals essentially gave up on Sybil resistance as well. The loss of non-repudiation should only have minor effects: V2X logs will still be a useful tool to analyze accidents in court, and transmitters of false information can still be uncovered, albeit with slightly more effort, by tracing key requests and responses at the time of the accident. The loss of non-repudiation is, on this account, a small price to pay for the privacy gains that zone encryption achieves.

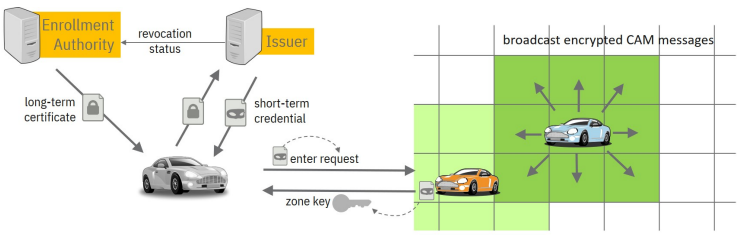

Fig. 1. Illustration of Zone Encryption with its Anonymous-Authentication Approach.

4.1 Syntax of Zone Encryption Schemes

A *Zone-Encryption* (ZE) scheme allows vehicles in a geographic zone at a given time to securely and anonymously communicate which each other. A ZE scheme features an *enrollment authority* $\mathcal{E}$, an *issuer* $\mathcal{I}$, and *vehicles* with unique identities $\mathcal{V} \in \{0,1\}^*$. The enrollment authority provides vehicles with revocable *long-term credentials* and may in practice be a state authority. A vehicle that has obtained a long-term credential is considered enrolled, and a vehicle identity can be enrolled (only once) in the system at any time.

The long-term credential is used to obtain *short-term credentials* from the issuer (which may in practice be another legal authority or a representative of a car-manufacturer consortium). That is, we assume that time is divided into (revocation) *epochs*, and all parties are assumed to be roughly synchronized, i.e., they share a common clock (e.g., a network clock). The duration of an epoch (e.g., a week) is the validity period of short-term credentials, and before the beginning of each epoch, a vehicle must interact with the issuer to obtain the short-term credential. These short-term credentials are irrevocable as they have limited validity anyway. However, the issuer learns the identity $\mathcal{V}$ of the vehicle and can check if its long-term credential has been revoked by the enrollment authority. For the sake of simplicity, we do not explicitly model revocation. As revocation would only be needed for standard, i.e., non-anonymous authentication, this can be added in a straightforward way.

A vehicle being equipped with a short-term credential can, during the epoch for which the credential was issued, communicate with other vehicles in an authenticated yet anonymous manner. More precisely, it uses the short-term credential to exchange so-called *zone keys* with other (anonymously) authenticated vehicles. These keys are valid for a particular zone and short time period, e.g., 15 minutes and enable vehicles to securely send and receive payloads that are encrypted under these keys.

A ZE scheme allows vehicles to communicate anonymously, but if need be, the issuer can recover the identity of the vehicle which computed a certain message. It can then revoke (i.e., blacklist) the vehicle identity and reject its authorization requests in the future.

To formally define a ZE scheme, let Z be a set of zones that cover the road network and let $\mathcal{P}$ be the payload space. Consider also a set of epochs *Epoch* and a set of time periods T, both non-empty finite integer sets such that for all $t \in T$, there exists a unique $e \in \mathit{Epoch}$ for which $e \leq t < e + 1$. Denote it by $e(t)$. These are parameters for the scheme. A ZE scheme then consists of the following algorithms:

Setup & Key Generation. A ZE scheme features an algorithm generating public parameters $\mathsf{Setup}\left(1^\lambda, Z, \mathit{Epoch}, T\right) \to pp$, as well as key-generation algorithms $\mathsf{KG.E}(pp) \to (pk_\mathcal{E}, (sk_\mathcal{E}, st_\mathcal{E}))$ and $\mathsf{KG.I}(pp) \to (pk_\mathcal{I}, (sk_\mathcal{I}, st_\mathcal{I}))$ respectively for the enrollment authority and the issuer. The private outputs also contain state $st_\mathcal{E}$ and $st_\mathcal{I}$ that are used to keep track of enrolled vehicles and open messages sent during key requests. Moreover, we assume that the public keys can be recovered from the secret keys.

Receiving Long-term and Short-term Credentials. A ZE scheme has two interactive protocols for $\mathcal{V}$ to obtain authentication credentials.

$\langle \mathsf{Enroll.V}(pk_\mathcal{E}, \mathcal{V}) \rightleftharpoons \mathsf{Enroll.E}(sk_\mathcal{E}, st_\mathcal{E}, \mathcal{V}) \rangle \to \langle cert_\mathcal{V}, st'_\mathcal{E} \rangle$: The Enroll protocol is run between a vehicle $\mathcal{V}$ and enrollment authority $\mathcal{E}$. If successful, $\mathcal{V}$ obtains a long-term certificate $cert_\mathcal{V}$.

$\langle \mathsf{Authorize.V}(cert_\mathcal{V}, e, pk_\mathcal{I}) \rightleftharpoons \mathsf{Authorize.I}(sk_\mathcal{I}, st_\mathcal{I}, \mathcal{V}, e, pk_\mathcal{E}) \rangle \to \langle cred_\mathcal{V}, st'_\mathcal{I} \rangle$: A vehicle $\mathcal{V}$ can use its long-term certificate to obtain from issuer $\mathcal{I}$ a short-term credential $cred_\mathcal{V}$ for an epoch e by running protocol $\mathsf{Authorize}$.

Entering and Exiting Zones. Protocol Enter is run when a vehicle $\mathcal{V}$ enters a zone z at time t. It is run with other responding vehicles $\mathcal{W}_i$, all authenticated via their short-term credentials $cred_{\mathcal{W}_i}$. If successful, the protocol allows the entering vehicle $\mathcal{V}$ to obtain the zone key $K_{z,t}$ for the zone–time pair (z, t). Algorithm Exit is used to remove key material from the zone-key list L_K of a vehicle when it exits a zone or when the time period has expired. The latter is crucial for our security model in which a vehicle, *after* leaving a zone, should no longer be able to decrypt messages or compute valid ciphertexts for it.

$\langle \mathsf{Enter.V}(cred_{\mathcal{V}}, L_K, pk_{\mathcal{I}}, z, t, requester)$
$\rightleftharpoons \mathsf{Enter.W}(cred_{\mathcal{W}_i}, L_{K_i}, pk_{\mathcal{I}}, z, t, responder_i)_{i\geq 0}\rangle \rightarrow \langle L_K, \perp\rangle$: Protocol Enter is run between a requesting vehicle $\mathcal{V}$ and other responding vehicles $\mathcal{W}_i$. List L_K consists of tuples $(z', t', K_{z',t'})$ to which, if the protocol is successful, a new key $K_{z,t}$ for the requested zone–time pair is added.

$\mathsf{Exit}(L_K, z, t) \rightarrow L'_K$: Removes $(z, t, K_{z,t})$ from L_K.

Sending and Receiving Payloads. Algorithms Send and Receive are used by a vehicle to exchange encrypted payloads. Note that these algorithms only need to access the zone keys stored in L_K, but not the short-and-long-term credentials, which is a security benefit compared with existing C-ITS solutions.

$\mathsf{Send}(L_K, P, Y \subseteq Z, t) \rightarrow \gamma/\perp$: Computes a ciphertext γ for a payload P for all zones Y in time period t (if L_K contains the corresponding keys). The ciphertext γ is assumed to carry public information about t and Y, i.e., it can be parsed as (t, Y, γ').

$\mathsf{Receive}(L_K, \gamma) \rightarrow P/\perp$: Recovers the payload P from ciphertext γ if L_K contains a zone key under which γ is encrypted.

Identity Escrow. When suspicious behaviour is detected or when an accident occurs, the issuer $\mathcal{I}$ of the short-term credentials can reveal the identity of a vehicle that sent a given message during an execution of protocol Enter.

$\mathsf{Open}(sk_{\mathcal{I}}, st_{\mathcal{I}}, m) \rightarrow \mathcal{V}/\perp$**:** returns the identity of a vehicle that it identifies as the sender of a message m during an execution of protocol Enter, or $\perp$.

Note that Open runs on a single anonymous message sent during an execution of protocol Enter, not on a full record of all messages ever sent by vehicles. It means that in practice, in case of a dispute or a suspicious event in a certain zone the issuer only needs to de-anonymize the messages sent during executions of protocol Enter for that zone at the time (period) of the event.

Correctness of Zone Encryption. A ZE scheme should satisfy *correctness*, i.e., if a vehicle is authorized during a given epoch and has entered a zone in a certain time period, then every message sent by that vehicle to this zone should be successfully received by any other vehicle in the zone in that time period. Moreover, the identity of a vehicle that sent a given message during an Enter protocol execution should be recoverable by the issuer. These properties should hold independently of the order in which certificates and credentials are issued for vehicles, and with overwhelming probability.

4.2 Security of Zone Encryption Schemes

We now describe the security and privacy properties a ZE scheme must satisfy.

The payloads sent by the vehicles should be confidential. This property is formalized as **Payload-Hiding security against Chosen-Ciphertext Attacks** (PH-CCA). Intuitively, PH-CCA security ensures that no efficient adversary can infer any information about the payload underlying a ciphertext, unless it has entered the zone in the time period of the ciphertext.

The privacy of vehicles should also be preserved, and this requirement is defined through an **anonymity** game. Essentially, anonymity guarantees that ciphertexts and enter-protocol messages do not reveal any information about the identity of the sending vehicle.

Note that there is no anonymity requirement for the authorization process, i.e., for receiving short-term credentials, as it is performed once per epoch (e.g., a week) and leaks very little information about the whereabouts of the vehicles. It is not an issue assuming that users have control on when it occurs.

Further, despite strong privacy properties, zone-encryption should ensure that only legitimate vehicles can send valid ciphertexts. This is captured via two related security definitions.

First, the **traceability** notion guarantees that if a vehicle knows a key $K_{z,t}$ for zone z at time t, then it must have explicitly entered the zone z at time t, meaning that it must have sent an enter message that can be traced back by the issuer to its long-term identity.

Secondly, the related notion of **ciphertext integrity** guarantees that an adversary cannot compute a valid ciphertext γ for a particular zone–time pair without knowing the zone key.

4.2.1 Common Oracles. We first introduce the oracles we give the adversary in all our security games. In the formal definitions, $\mathcal{O}(sk_{\mathcal{E}}, st_{\mathcal{E}}, sk_{\mathcal{I}}, st_{\mathcal{I}})$ denotes that the adversary is given access to oracles $\{\mathsf{Enroll.E}, \mathsf{Enroll.V\&E}, \mathsf{Authorize.I}, \mathsf{Authorize.V\&I}, \mathsf{Enter}, \mathsf{Exit}, \mathsf{Send}, \mathsf{Receive}, \mathsf{Open}, \mathsf{Corrupt}\}$ as defined hereunder and initialized with secret keys $sk_{\mathcal{E}}, st_{\mathcal{E}}, sk_{\mathcal{I}}, st_{\mathcal{I}}$. The public keys $pk_{\mathcal{E}}, pk_I$ are not made explicit, but are assumed to be recoverable from the corresponding secret keys.

Throughout the security experiments, the challenger maintains several lists which reflect the information $\mathcal{A}$ learns through his interaction with the oracles. These are summarized on Figure 2.

Notation. "$\mathcal{O}.\mathsf{algorithm.P}$" denotes the oracle which lets the adversary interact with honest party $\mathcal{P}$ running $\mathsf{algorithm.P}$. Similarly, an oracle "$\mathcal{O}.\mathsf{algorithm.P\&R}$" lets the adversary trigger the interactive protocol $\langle \mathsf{algorithm.P} \rightleftharpoons \mathsf{algorithm.R} \rangle$ between two honest parties $\mathcal{P}$ and $\mathcal{R}$. In the latter case, the adversary does not learn the outputs of honest parties, but their internal states are updated accordingly. Moreover, when an oracle is said to be running an algorithm on behalf of an *honest* vehicle $\mathcal{V}$, it is implicitly assumed that the oracle checks that $\mathcal{V} \in \mathcal{L}_{\mathsf{honest}}$. Finally, The state of an honest vehicle $\mathcal{V}$ is referred to as $\mathcal{V}[st_{\mathcal{V}}]$, e.g., $\mathcal{V}[L_K]$ denotes the zone keys L_K maintained by $\mathcal{V}$.

$\mathcal{L}_{\mathsf{honest}}$	list of all enrolled honest vehicles $\{(\mathcal{V})\}$
$\mathcal{L}_{\mathsf{corrupt}}$	enrolled vehicles $\{(\mathcal{V})\}$ that where corrupt either from the beginning or later on
$\mathcal{L}_{\mathsf{auth}}$	authorized vehicles $\{(\mathcal{V}, e)\}$ per epoch e
$\mathcal{L}_{\mathsf{enter}}$	contains all messages $\{(\{\mathcal{V}, \mathcal{A}\}, z, t, m)\}$ that honest vehicles or the adversary $\mathcal{A}$ exchanged during executions of protocol Enter
$\mathcal{L}_{\mathsf{sent}}$	ciphertexts $\{\gamma\}$ generated by honest vehicles
$\mathcal{L}_{\mathsf{received}}$	decrypted ciphertexts $\{\gamma = (t, Y, \gamma')\}$
$\mathcal{L}_{\mathsf{opened}}$	opened transcripts m
$\mathcal{L}_{\mathsf{keys}}$	zone-keys $\{(z, t, K_{t,z})\}$ that the adversary $\mathcal{A}$ learned by corrupting honest vehicles

Fig. 2. Lists maintained by the Challenger in the ZE Security Experiments.

Oracles for Obtaining Credentials. There are a number of oracles to model enrollment and issuance of short-term credentials, depending on whether the requesting vehicle is honest or corrupt.

$\mathcal{O}.\mathsf{Enroll.V\&E}(sk_{\mathcal{E}}, st_{\mathcal{E}}, \cdot)$ on input $\mathcal{V}$, lets the adversary trigger the enrollment protocol between an honest vehicle with identity $\mathcal{V}$ and the honest enrollment authority $\mathcal{E}$. If Enroll.V ends with a private output $\mathcal{V}[cert]$, it adds $\mathcal{V}$ to $\mathcal{L}_{\mathsf{honest}}$.

$\mathcal{O}.\mathsf{Enroll.E}(sk_{\mathcal{E}}, st_{\mathcal{E}}, \cdot)$ on input $\mathcal{V}$, lets the adversary run an enrollment protocol (in the role of the corrupt vehicle $\mathcal{V}$) with the honest enrollment authority. If Enroll.E ends with a private output $st'_{\mathcal{E}}$, it adds $\mathcal{V}$ to $\mathcal{L}_{\mathsf{corrupt}}$.

$\mathcal{O}.\mathsf{Authorize.V\&I}(sk_{\mathcal{I}}, st_{\mathcal{I}}, \cdot)$ on input $(\mathcal{V}, e)$, triggers an Authorize protocol between the honest vehicle $\mathcal{V}$ and honest issuer $\mathcal{I}$. If Authorize.V ends with private output $\mathcal{V}[e, cred_{\mathcal{V}}]$, it adds $(\mathcal{V}, e)$ to $\mathcal{L}_{\mathsf{auth}}$.

$\mathcal{O}.\mathsf{Authorize.I}(sk_{\mathcal{I}}, st_{\mathcal{I}}, \cdot)$ on input $(\mathcal{V}, e)$, allows a corrupt vehicle $\mathcal{V}$, played by the adversary, to run the Authorize protocol with the honest issuer $\mathcal{I}$. If Authorize.I ends with private output $st'_{\mathcal{I}}$, it adds $(\mathcal{V}, e)$ to $\mathcal{L}_{\mathsf{auth}}$.

Oracles for Entering and Exiting Zones. The adversary is further given access to an oracle which lets it actively participate in the Enter protocol as well as eavesdrop on enter-protocol executions between honest vehicles. Another oracle lets the adversary make an honest vehicle exit a zone.

$\mathcal{O}.\mathsf{Enter}(\cdot)$ on input $(\mathcal{V}, z, t, role)$, triggers a zone-key request or response protocol (according to *role*) for a honest vehicle $\mathcal{V}$ in zone z and time period t.

- For $role = requester$, the oracle starts Enter.V for $\mathcal{V}$ in the *requester* role and also internally invokes all other honest vehicles $\mathcal{W}_i$ which have zone keys for (z, t) to run Enter.W with $role = responder$. The adversary can intercept and inject messages sent by these honest vehicles, and also participate in the *responder* role with a corrupt vehicle. Eventually, the key state $\mathcal{V}[L_K]$ of the honest requester is updated to include $K_{z,t}$.
- For $role = responder$ the oracle lets an honest vehicle $\mathcal{V}$ respond to a zone-key request that the adversary runs for a corrupt vehicle.

All messages $(\mathcal{V}, z, t, m)$ sent by honest vehicles $\mathcal{V}$ are tracked with list $\mathcal{L}_{\mathsf{enter}}$. Similarly, when an honest vehicle receives a message m that no other honest vehicle has sent, the message is recorded as adversarial by adding $(\mathcal{A}, z, t, m)$ to $\mathcal{L}_{\mathsf{enter}}$.

Note that this oracle captures both *active* and *passive* attacks. The latter can be done if the adversary queries $\mathcal{O}.\mathsf{Enter}$ for *role* = *requester* and does not participate as corrupt responder or manipulates messages, but merely observes the traffic between the honest vehicles in a certain zone and time period (z, t). There is then no message $(\mathcal{A}, z, t, m) \in \mathcal{L}_{\mathsf{enter}}$; and $\mathcal{A}$ is considered successful if it infers information for (z, t) that is supposed to only be known to vehicles which entered the zone, e.g., if it manages to distinguish ciphertexts encrypted for (z, t) (PH-CCA) or if it can produce a valid ciphertext (ciphertext integrity).

$\mathcal{O}.\mathsf{Exit}(\cdot)$ on input $(\mathcal{V}, z, t)$, deletes key $K_{z,t}$ from the key state of the honest vehicle $\mathcal{V}$.

Oracles for Sending and Receiving Payloads, Opening and Corruption. Finally, the adversary is given access to oracles that can trigger honest vehicles to encrypt or decrypt messages, recover the identity of sending vehicles, and adaptively corrupt vehicles.

$\mathcal{O}.\mathsf{Send}(\cdot)$ on input $(\mathcal{V}, P, Y, t)$ for an honest vehicle $\mathcal{V}$, returns $\gamma/\perp \leftarrow \mathsf{Send}(\mathcal{V}[L_K], P, Y, t)$ and adds γ to $\mathcal{L}_{\mathsf{sent}}$.

$\mathcal{O}.\mathsf{Receive}(\cdot)$ on input $(\mathcal{V}, \gamma)$ for an honest vehicle $\mathcal{V}$, returns $m/\perp \leftarrow \mathsf{Receive}(\mathcal{V}[L_K], \gamma)$ and adds γ to $\mathcal{L}_{\mathsf{received}}$.

$\mathcal{O}.\mathsf{Open}(sk_{\mathcal{I}}, st_{\mathcal{I}}, \cdot)$ on input m, returns $\mathcal{V}/\perp \leftarrow \mathsf{Open}(sk_{\mathcal{I}}, st_{\mathcal{I}}, m)$ and adds m to $\mathcal{L}_{\mathsf{opened}}$.

$\mathcal{O}.\mathsf{Corrupt}(\cdot)$ on input $\mathcal{V}$, returns the current state of the honest vehicle $\mathcal{V}$, i.e., it returns $\mathcal{V}[cert_{\mathcal{V}}]$, all $\mathcal{V}[\{(e_j, cred_{\mathcal{V},j})\}]$, and $\mathcal{V}[L_K]$ to the adversary. It also adds $\mathcal{V}$ to $\mathcal{L}_{\mathsf{corrupt}}$ and all keys $(z, t, K_{z,t})$ in $\mathcal{V}[L_K]$ to $\mathcal{L}_{\mathsf{keys}}$.

4.2.2 Payload Hiding. Payload-hiding security against chosen-ciphertext attacks (see Figure 3) guarantees that an adversary cannot infer any information about messages encrypted for a zone it is not supposed to be in. Our definition follows the classical CCA definition and requires the adversary to output two payloads P_0, P_1 together with a time t^* and zones Y^*, upon which it receives the zone encryption of P_b for a random $b \in \{0, 1\}$. The adversary must then determine b better than by guessing. We give the adversary access to honest participants in the system, e.g., by allowing it to enroll and authorize vehicles, enter zones, encrypt and decrypt messages of its choice, and to corrupt vehicles.

The adversary wins as long as its interactions with these oracles do not lead to a trivial win. Clearly, the adversary is not allowed to query the decryption oracle $\mathcal{O}.\mathsf{Receive}$ on (parts of) the challenge ciphertext (condition 1), or corrupt an honest vehicle that has a zone-key for one of the challenge zones in Y^* at time t^* (condition 2). Furthermore, the adversary must not have entered any challenge zone at time t^* with a corrupt vehicle, or if it did, it must not have a

valid authorization credential for epoch $e(t^*)$ (condition 3). The latter condition is crucial as our PH-CCA notion should guarantee message confidentiality for all zones the adversary was not *supposed* to be in.

Experiment $\mathbf{Exp}^{\text{ph-cca-}b}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A})$:

$pp \leftarrow \mathsf{Setup}(1^\lambda, Z, Epoch, T)$
$(pk_{\mathcal{E}}, (sk_{\mathcal{E}}, st_{\mathcal{E}})) \leftarrow \mathsf{KG.E}(pp)$, $(pk_{\mathcal{I}}, (sk_{\mathcal{I}}, st_{\mathcal{I}})) \leftarrow \mathsf{KG.I}(pp)$

initialize all oracles as $\mathcal{O}(sk_{\mathcal{E}}, st_{\mathcal{E}}, sk_{\mathcal{I}}, st_{\mathcal{I}})$
$(\mathcal{V}^*, P_0, P_1, Y^*, t^*, state_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{choose}, pp, pk_{\mathcal{E}}, pk_{\mathcal{I}})$
abort if $\mathcal{V}^* \in \mathcal{L}_{\mathsf{corrupt}}$

$\gamma^* \leftarrow \mathsf{Send}(\mathcal{V}^*[L_K], P_b, Y^*, t^*)$ with $\gamma^* = (t^*, Y^*, \gamma^{*'})$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{guess}, \gamma^*, state_{\mathcal{A}})$
return b' if $\mathcal{A}$ did not trivially win, i.e.:
1) $\forall (t^*, Y, \gamma) \in \mathcal{L}_{\mathsf{received}} : \gamma \cap \gamma^{*'} = \emptyset$ and

2) $\forall y^* \in Y^* : (y^*, t^*, \cdot) \notin \mathcal{L}_{\mathsf{keys}}$, i.e., *$\mathcal{A}$ has not corrupted a vehicle in a challenge zone*, and

3a) $\forall y^* \in Y^* : ((\mathcal{A}, y^*, t^*, \cdot) \notin \mathcal{L}_{\mathsf{enter}})$ **or**

3b) $\exists (\mathcal{A}, y^*, t^*, \cdot) \in \mathcal{L}_{\mathsf{enter}}$ and $\forall \mathcal{V}_j \in \mathcal{L}_{\mathsf{corrupt}}, \nexists (\mathcal{V}_j, e(t^*)) \in \mathcal{L}_{\mathsf{auth}}$
i.e., *$\mathcal{A}$ has not entered a challenge zone in time t^* or entered but was not authorized*

Fig. 3. PH-CCA Experiment for ZE Schemes.

Definition 3 (PH-CCA Security). *A ZE scheme $\mathcal{Z}$ is PH-CCA secure if there exists a negligible function* negl *such that for all efficient adversary $\mathcal{A}$, for all $\lambda \in \mathbb{N}$, zone-set Z, epoch set Epoch and time-period set T,*

$$\left|\Pr\left[\mathbf{Exp}^{\text{ph-cca-}0}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A}) = 1\right] - \text{pr}\left[\mathbf{Exp}^{\text{ph-cca-}1}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A}) = 1\right]\right| \leq \text{negl}(\lambda).$$

4.2.3 Anonymity. Anonymity (see Figure 4) captures the idea that ZE ciphertexts and the messages sent during executions of protocol Enter do not reveal any information about the identity of the sending vehicle. This includes *unlinkability*, i.e., the adversary cannot tell whether two ciphertexts or two enter-protocol messages stem from the same vehicle.

Our definition follows the indistinguishability style, and it grants the adversary oracle access to honest participants. In particular, the adversary can enter and exit zones with honest vehicles, as well as send payloads and receive ciphertexts with them. The adversary must eventually output two challenge vehicle identities $\mathcal{V}_0$ and $\mathcal{V}_1$, after which it gets access to vehicles $\mathcal{V}_b$ and $\mathcal{V}_{1-b}$ and has to determine b. In the experiment, it is captured by turning all oracles that should not leak information about the vehicles identity into challenge oracles.

That is, the oracles to enter and exit zones, or to send payloads and receive ciphertexts are restricted to no longer respond to queries for identities $\mathcal{V}_0$ or $\mathcal{V}_1$. If the adversary wants to make such a query for either of them, it has to provide a bit d and the query is answered with vehicle $\mathcal{V}_{d\oplus b}$, i.e., either the chosen vehicle $\mathcal{V}_b$ (for $d = 0$) or its counterpart $\mathcal{V}_{1-b}$ (for $d = 1$).

To avoid trivial wins, the oracles to enter and exit zones cannot be queried at a time t for a challenge vehicle if $\mathcal{V}_0$ and $\mathcal{V}_1$ have not *both* been authorized in epoch $e(t)$. Besides, the adversary can never open a message sent by one of the challenge vehicles during an execution of protocol Enter.

Note that this definition does not require the authorization protocol to be anonymous, but only ZE ciphertexts and messages exchanged during executions of protocol Enter. As authorization is performed only once per epoch, it is not critical for the privacy guarantees we aim for in V2X communication.

Experiment $\mathbf{Exp}^{\mathrm{ano-b}}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A})$:

$pp \leftarrow \mathsf{Setup}(1^\lambda, Z, Epoch, T)$
$(pk_{\mathcal{E}}, (sk_{\mathcal{E}}, st_{\mathcal{E}})) \leftarrow \mathsf{KG.E}(pp)$, $(pk_{\mathcal{I}}, (sk_{\mathcal{I}}, st_{\mathcal{I}})) \leftarrow \mathsf{KG.I}(pp)$

initialize all oracles as $\mathcal{O}(sk_{\mathcal{E}}, st_{\mathcal{E}}, sk_{\mathcal{I}}, st_{\mathcal{I}})$
$(\mathcal{V}_0, \mathcal{V}_1, state_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{choose}, pp, pk_{\mathcal{E}}, pk_{\mathcal{I}})$
abort if the vehicles have different cred/key states, i.e., check that:
  for $d \in \{0,1\}$: $\nexists e_i$ s.t. $(\mathcal{V}_d, e_i) \in \mathcal{L}_{\mathsf{auth}} \wedge (\mathcal{V}_{1-d}, e_i) \notin \mathcal{L}_{\mathsf{auth}}$ and $\mathcal{V}_0[L_K] = \mathcal{V}_1[L_K]$

use challenge oracles $\mathcal{O}^*(b)$ for $\{\mathsf{Enter}^*, \mathsf{Exit}^*, \mathsf{Send}^*, \mathsf{Receive}^*\}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^*}(\mathsf{guess}, state_{\mathcal{A}})$
return b' if $\mathcal{A}$ did not trivially win, i.e.,
  for $d \in \{0,1\}$: $\mathcal{V}_d \in \mathcal{L}_{\mathsf{honest}}$ and $\forall m \in \mathcal{L}^*_{\mathsf{enter}} : m \notin \mathcal{L}_{\mathsf{opened}}$

Fig. 4. Anonymity Experiment for ZE Schemes.

Definition 4 (Anonymity). *A ZE scheme $\mathcal{Z}$ satisfies anonymity if there exists a negligible function* negl *such that for all efficient adversary $\mathcal{A}$, for all $\lambda \in \mathbb{N}$, zone-set Z, epoch set Epoch and time-period set T,*

$$\left|\Pr\left[\mathbf{Exp}^{\mathrm{ano-0}}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A}) = 1\right] - \Pr\left[\mathbf{Exp}^{\mathrm{ano-1}}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A}) = 1\right]\right| \leq \mathrm{negl}(\lambda).$$

4.2.4 Traceability. The notion of traceability ensures that if a vehicle knows a secret key $K_{z,t}$ for a zone–time pair, then it must have entered the zone–time pair by sending a message that can be traced back to the sending vehicle. This is captured via a game (on Figure 5) where the adversary is must output a key K_{z^*,t^*} for a zone z^* and time t^* of its choice. The adversary wins if at least one honest vehicle $\mathcal{V}$ has accepted the key, but none of the messages in executions of protocol Enter for (z^*, t^*) can be traced with algorithm Open to a corrupt vehicle

(that was authorized to enter). To avoid trivial wins we further request that the adversary has not corrupted an honest vehicle that held the key K_{z^*,t^*} output by the adversary (condition 1). Moreover, no corrupt vehicle can be authorized in epoch $e(t^*)$ (conditions 2) as otherwise the adversary would be able to impose a zone key.

In particular, for a ZE scheme that satisfies traceability, if an efficient adversary knows the zone key of an honest vehicle, then it has either corrupted another honest vehicle in the zone, or it must have sent at least one message that traces back to a corrupt vehicle that was authorized in epoch $e(t^*)$.

Experiment $\mathbf{Exp}^{\text{trace}}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A})$:

$pp \leftarrow \mathsf{Setup}(1^\lambda, Z, Epoch, T)$
$(pk_{\mathcal{E}}, (sk_{\mathcal{E}}, st_{\mathcal{E}})) \leftarrow \mathsf{KG.E}(pp)$, $(pk_{\mathcal{I}}, (sk_{\mathcal{I}}, st_{\mathcal{I}})) \leftarrow \mathsf{KG.I}(pp)$

initialize all oracles as $\mathcal{O}(sk_{\mathcal{E}}, st_{\mathcal{E}}, sk_{\mathcal{I}}, st_{\mathcal{I}})$
$(z^*, t^*, K_{z^*,t^*}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{forge}, pp, pk_{\mathcal{E}}, pk_{\mathcal{I}})$
look up $\mathcal{V} \in \mathcal{L}_{\mathsf{honest}}$ with $K_{z^*,t^*} \in \mathcal{V}[L_K]$, abort if no such $\mathcal{V}$ exists

return 1 if knowledge of K_{z^*,t^*} cannot be traced to a corrupt vehicle:
1) $K_{z^*,t^*} \notin \mathcal{L}_{\mathsf{keys}}$ and 2) $\forall(\cdot, z^*, t^*, m_j) \in \mathcal{L}_{\mathsf{enter}}$ with $\mathcal{V}_j \leftarrow \mathsf{Open}(sk_I, st_I, m_j)$:
$\mathcal{V}_j \notin \mathcal{L}_{\mathsf{corrupt}}$ or $((\mathcal{V}_j \in \mathcal{L}_{\mathsf{corrupt}})$ and $(\nexists(\mathcal{V}_j, e(t^*)) \in \mathcal{L}_{\mathsf{auth}}))$

Fig. 5. Traceability Experiment for ZE Schemes.

Definition 5 (Traceability). *A ZE scheme $\mathcal{Z}$ satisfies traceability if there exists a negligible function* negl *such that for all efficient adversary $\mathcal{A}$, for all $\lambda \in \mathbb{N}$, zone-set Z, epoch set Epoch and time-period set T,*

$$\Pr\left[\mathbf{Exp}^{\text{trace}}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A}) = 1\right] \leq \text{negl}(\lambda).$$

4.2.5 Ciphertext Integrity. The notion of ciphertext integrity (see Figure 6) complements the traceability property as it guarantees that without knowing a secret zone key $K_{z,t}$ an adversary should not be able to compute a valid ciphertext for that zone and time. This is modeled by asking the adversary to produce a fresh and valid ciphertext γ^* for zones for which it is not supposed to know the keys. The adversary also outputs an honest vehicle $\mathcal{V}$ that must decrypt γ^* to $P \neq \perp$.

Freshness means that γ^* should not contain any honestly generated ciphertexts (or parts thereof) the adversary has received via oracle $\mathcal{O}.\mathsf{Send}$ (condition 1). Moreover, the same conditions as in the PH-CCA game are used to check that the adversary is *not supposed* to know the key. That is, the adversary must not have corrupted an honest vehicle that knows a valid key for the forged ciphertext (condition 2). Besides, it must not have entered any challenge zone at

time t^* with a corrupt vehicle, or if it entered, it must not have a valid authorization credential for epoch $e(t^*)$ (condition 3). The last condition means that the adversary *does* win the game if it knows the key of a zone–time pair it was *not allowed* to enter.

Experiment $\mathbf{Exp}^{\text{integrity}}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A})$:

$pp \leftarrow \mathsf{Setup}(1^\lambda, Z, Epoch, T)$
$(pk_{\mathcal{E}}, (sk_{\mathcal{E}}, st_{\mathcal{E}})) \leftarrow \mathsf{KG.E}(pp)$, $(pk_{\mathcal{I}}, (sk_{\mathcal{I}}, st_{\mathcal{I}})) \leftarrow \mathsf{KG.I}(pp)$

initialize all oracles as $\mathcal{O}(sk_{\mathcal{E}}, st_{\mathcal{E}}, sk_{\mathcal{I}}, st_{\mathcal{I}})$
$(\mathcal{V}, \gamma^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{forge}, pp, pk_{\mathcal{E}}, pk_I)$
parse $\gamma^* = (t^*, Y^*, \gamma^{*'})$, abort if $\mathcal{V} \in \mathcal{L}_{\mathsf{corrupt}}$

return 1 if $\mathsf{Receive}(\mathcal{V}[L_K], \gamma^*) \neq \perp$ and $\mathcal{A}$ did not trivially win, i.e.,
1) $\forall (t^*, Y, \gamma) \in \mathcal{L}_{\mathsf{sent}} : \gamma \cap \gamma^{*'} = \emptyset$ and

2) $\forall y^* \in Y^* : (y^*, t^*, \cdot) \notin \mathcal{L}_{\mathsf{keys}}$ and
i.e., *$\mathcal{A}$ has not corrupted a vehicle in a challenge zone*

3a) $\forall y^* \in Y^* : ((\mathcal{A}, y^*, t^*, \cdot) \notin \mathcal{L}_{\mathsf{enter}})$ **or**

3b) $\exists (\mathcal{A}, y^*, t^*, \cdot) \in \mathcal{L}_{\mathsf{enter}}$ and $\forall \mathcal{V}_j \in \mathcal{L}_{\mathsf{corrupt}} : \nexists (\mathcal{V}_j, e(t^*)) \in \mathcal{L}_{\mathsf{auth}}$
i.e., *$\mathcal{A}$ has not entered a challenge zone in time t^* or entered but was not authorized*

Fig. 6. Ciphertext-Integrity Experiment for ZE Schemes.

Definition 6 (Ciphertext Integrity). *A ZE scheme $\mathcal{Z}$ satisfies ciphertext-integrity if there exists a negligible function* negl *such that for every efficient adversary $\mathcal{A}$, for all $\lambda \in \mathbb{N}$, zone-set Z, epoch set Epoch and time-period set T,*

$$\Pr\left[\mathbf{Exp}^{\text{integrity}}_{\mathcal{Z},\lambda,Z,Epoch,T}(\mathcal{A}) = 1\right] \leq \text{negl}(\lambda).$$

4.3 Our Zone Encryption Scheme

We here describe a generic zone-encryption scheme constructed from authenticated encryption, public-key encryption, digital signatures and a dynamic group signature scheme with attributes. For the latter, see the practical, pairing-based instantiation in Section 3.2. Our ZE scheme involves the following building blocks:

- SIG a signature scheme to generate long-term credentials and thereby certify vehicle identities
- DGSA a group-signature scheme (Section 3.1) used to compute short-term authentication credentials. Group-membership credentials are issued w.r.t. the current time epoch $e(t)$

– PKE a public-key encryption scheme to encrypt zone keys during key requests and responses
– SE a symmetric-key encryption scheme to encrypt payloads
– DAE a deterministic authenticated encryption scheme to wrap payload keys with each zone key, and thereby bind payload ciphertexts to their zones.

The reason we use symmetric encryption to encrypt payloads and only authenticate key wraps is that payloads may a priori be long. Authenticating the payload part of the ciphertext would increase its length. Only authenticating the key wraps and bind the payload part to them results in shorter ciphertext.

4.3.1 Formal Description. Recall that each short-term credential is only valid in a certain epoch (e.g., a week), and that zone keys must be refreshed at the beginning of every time period (e.g., every 15 minutes). It is assumed that, during protocol executions, whenever an algorithm receives an abort or invalid message, or a verification step fails, it aborts by returning $\perp$. Likewise, when an algorithm must retrieve a key from its internal state, it aborts if no such key can be found.

Our ZE scheme $\mathcal{Z}$, parametrized by a zone set Z, an epoch set $Epoch$ and a time-period set T, is defined as follows.

Setup & Key Generation. The setup and key generation algorithms simply run the respective algorithms of the building blocks and generate the keys and parameters accordingly.

$\mathsf{Setup}\left(1^\lambda, Z, Epoch, T\right)$: Generate public parameters for SIG, DGSA (with one attribute), PKE, SE and DAE and return $pp \leftarrow (pp_{\mathsf{SIG}}, pp_{\mathsf{DGSA}}, pp_{\mathsf{PKE}}, pp_{\mathsf{SE}}, pp_{\mathsf{DAE}}, Z, Epoch, T)$.

$\mathsf{KG.E}(pp)$: Run $(vk, sk) \leftarrow \mathsf{SIG.KG}(pp_{\mathsf{SIG}})$, set keys as $(pk_{\mathcal{E}}, sk_{\mathcal{E}}) \leftarrow (vk, sk)$, $st_{\mathcal{E}} \leftarrow \emptyset$, and return the tuple $(pk_{\mathcal{E}}, (sk_{\mathcal{E}}, st_{\mathcal{E}}))$.

$\mathsf{KG.I}(pp)$: Run and return $(pk_{\mathcal{I}}, (sk_{\mathcal{I}}, st_{\mathcal{I}})) \leftarrow \mathsf{DGSA.KG}(pp_{\mathsf{DGSA}})$.

Issuance of Long-Term and of Short-Term Credentials. To enroll in the communication system, a vehicle with identity $\mathcal{V}$ must request a long-term certificate, which is simply a signature on $\mathcal{V}$ by the enrollment authority. From then on, it can request short-term credentials at the beginning of each epoch from issuer $\mathcal{I}$. Credentials are DGS+A membership credentials for an epoch e as attribute, and are used to authenticate vehicles during protocol Enter.

$\mathsf{Enroll.V} \rightleftharpoons \mathsf{Enroll.E}$: The vehicle and the EA proceed as follows.

1. $\mathsf{Enroll.V}(pk_{\mathcal{E}}, \mathcal{V})$:
   – $(vk_{\mathcal{V}}, sk_{\mathcal{V}}) \leftarrow \mathsf{SIG.KG}(pp_{\mathsf{SIG}})$, send $(\mathcal{V}, vk_{\mathcal{V}})$ to $\mathcal{E}$
2. $\mathsf{Enroll.E}(sk_{\mathcal{E}}, st_{\mathcal{E}}, \mathcal{V})$ upon receiving $(\mathcal{V}, vk_{\mathcal{V}})$:
   – check that $\mathcal{V} \notin st_{\mathcal{E}}$ (to ensure that a vehicle identity can be enrolled only once), send a signature $\sigma_{\mathcal{E}} \leftarrow \mathsf{SIG.Sign}(sk_{\mathcal{E}}, (\mathcal{V}, vk_{\mathcal{V}}))$ to $\mathcal{V}$ and return $st'_{\mathcal{E}} \leftarrow st_{\mathcal{E}} \cup \mathcal{V}$
3. Enroll.V upon receiving $\sigma_{\mathcal{E}}$ from $\mathcal{E}$:

– if $\mathsf{SIG.Vf}(pk_{\mathcal{E}}, (\mathcal{V}, vk_{\mathcal{V}}), \sigma_{\mathcal{E}}) = 1$, return $cert_{\mathcal{V}} \leftarrow (sk_{\mathcal{V}}, vk_{\mathcal{V}}, \sigma_{\mathcal{E}})$.

$\mathsf{Authorize.V} \rightleftharpoons \mathsf{Authorize.I}$: 1. $\mathsf{Authorize.V}(cert_{\mathcal{V}}, e, pk_{\mathcal{I}})$ with $cert_{\mathcal{V}}$ parsed as $(sk_{\mathcal{V}}, vk_{\mathcal{V}}, \sigma_{\mathcal{E}})$:
– compute $\sigma_{\mathcal{V}} \leftarrow \mathsf{SIG.Sign}(sk_{\mathcal{V}}, e)$
– send $(vk_{\mathcal{V}}, \sigma_{\mathcal{E}}, \sigma_{\mathcal{V}})$ to $\mathcal{I}$
2. $\mathsf{Authorize.I}(sk_I, st_{\mathcal{I}}, \mathcal{V}, e, pk_{\mathcal{E}})$ upon receiving $(vk_{\mathcal{V}}, \sigma_{\mathcal{E}}, \sigma_{\mathcal{V}})$:
– abort if $\mathcal{V}$ is revoked (this is handled outside the scheme)
– test whether $\mathsf{SIG.Vf}(pk_{\mathcal{E}}, (\mathcal{V}, vk_{\mathcal{V}}), \sigma_{\mathcal{E}}) = 1$ and $\mathsf{SIG.Vf}(vk_{\mathcal{V}}, e, \sigma_{\mathcal{V}}) = 1$
3. $\mathcal{I}$ and $\mathcal{V}$ run the issuance protocol of DGSA with $id = \mathcal{V}$ and attribute $A = e$, i.e., $\langle \mathsf{DGSA.Issue.U}(\mathcal{V}, e, pk_{\mathcal{I}}) \rightleftharpoons \mathsf{DGSA.Issue.I}(sk_{\mathcal{I}}, st_{\mathcal{I}}, \mathcal{V}, e) \rangle$ and return their respective outputs *cred* and $st'_{\mathcal{I}}$.

Entering and Exiting Zones. A vehicle which approaches a zone z in time period t obtains the key for (z, t) by sending an anonymously authenticated key request wich includes a fresh public-key encryption key ek. Any vehicle which receives the request and knows the zone key $K_{z,t}$ can send an anonymously authenticated response which contains an encryption of $K_{z,t}$ under ek. The tokens authenticate messages consisting of z, t, and the fresh key ek for requests or encryptions ct under ek of the zone key $K_{z,t}$.

If the requesting vehicle receives no response, it generates a random key $K_{z,t}$ and waits for requests from new vehicles that join the zone.

A vehicle determines whether it should reply according to a predetermined strategy, e.g., the vehicle closest to the requesting vehicle should reply to the key request. The optimal strategy depends on the zone structure, the traffic and other practical factors, and it is an engineering problem on its own. We here assume the existence of such a pre-established key-response strategy among all vehicles. Finally, once the time period t has elapsed, $\mathcal{V}$ simply deletes $K_{z,t}$ from its internal key state.

$\mathsf{Enter.V} \rightleftharpoons [\mathsf{Enter.W}_i]$: The inputs are assumed to be well-formed, i.e., credentials $cred_{\mathcal{V}}$ and $cred_{\mathcal{W}_i}$ are valid for epoch $e(t)$.
1. $\mathcal{V}$ running $\mathsf{Enter.V}(cred_{\mathcal{V}}, L_K, pk_{\mathcal{I}}, z, t, requester)$:
– return L_K if $\exists\ (z, t, K_{z,t}) \in L_K$
– $(ek, dk) \leftarrow \mathsf{PKE.KG}(pp_{\mathsf{PKE}})$
– $tok_{\mathcal{V}} \leftarrow \mathsf{DGSA.Auth}(pk_{\mathcal{I}}, cred_{\mathcal{V}}, (z, t, ek))$
– broadcast $(z, t, ek, tok_{\mathcal{V}})$
2. $\mathcal{W}_i$ running $\mathsf{Enter.W}(cred_{\mathcal{W}_i}, L_{K_i}, pk_{\mathcal{I}}, z, t, responder_i)$ upon receiving $(z, t, ek, tok_{\mathcal{V}})$ from a vehicle $\mathcal{V}$:
– verify that $\mathsf{DGSA.Vf}(pk_{\mathcal{I}}, (z, t, ek), e(t), tok_{\mathcal{V}}) = 1$
– retrieve $(z, t, K_{z,t}) \in L_{K_i}$
– $ct \leftarrow \mathsf{PKE.Enc}(ek, K_{z,t})$
– $tok_{\mathcal{W}} \leftarrow \mathsf{DGSA.Auth}(pk_{\mathcal{I}}, cred_{\mathcal{W}_i}, (z, t, ct))$
– send $(z, t, ct, tok_{\mathcal{W}})$ to vehicle $\mathcal{V}$

(3a) Vehicle $\mathcal{V}$ upon receiving $(z, t, ct, tok_{\mathcal{W}})$ from a vehicle $\mathcal{W}_i$:
– verify that $\mathsf{DGSA.Vf}(pk_{\mathcal{I}}, (z, t, ct), e(t), tok_{\mathcal{W}}) = 1$
– decrypt $K_{z,t} \leftarrow \mathsf{PKE.Dec}(dk, ct)$, return $L_K \leftarrow L_K \cup (z, t, K_{z,t})$

(3b) If $\mathcal{V}$ does not receive a response after a predetermined waiting time:
– $K_{z,t} \leftarrow \mathsf{DAE.KG}(1^\lambda)$, return $L_K \leftarrow L_K \cup (z, t, K_{z,t})$

$\mathsf{Exit}(L_K, z, t)$: If $(z, t, K_{z,t}) \in L_K$ return $L'_K \leftarrow L_K \backslash (z, t, K_{z,t})$.

Sending and Receiving Payloads. To encrypt CAMs, referred to as payloads in the construction, a vehicle generates a fresh symmetric key K_P and encrypts the payload with it. It then wraps the payload key with the key $K_{z,t}$ of each of the zones to which it intends to send the CAM. By using a deterministic[4]authenticated encryption scheme [48] to wrap fresh payload keys, it is guaranteed that the message originates from a genuine vehicle, as it had to authenticate itself to obtain the zone key. We therefore eliminate the need for a separate signature on each CAM message, yielding considerable savings in terms of computation. In addition to that, the authentication provided by scheme DAE is extented to the payload ciphertext ct by including ct as the header when K_P is encrypted under the zone keys.

$\mathsf{Send}(L_K, P, Y \subseteq Z, t)$: to send a payload P,

1. retrieve keys $\{K_{y,t}\}$ for all zones $y \in Y$ and time t from L_K
2. $ct \leftarrow \mathsf{SE.Enc}(K_P, P)$ with $K_P \leftarrow \mathsf{SE.KG}(pp_{\mathsf{SE}})$
3. for all $y \in Y$: $\gamma_{y,t} \leftarrow \mathsf{DAE.Enc}(K_{y,t}, ct, K_P)$
4. return $\gamma \leftarrow (t, Y, ((y, \gamma_{y,t})_{y \in Y}, ct))$

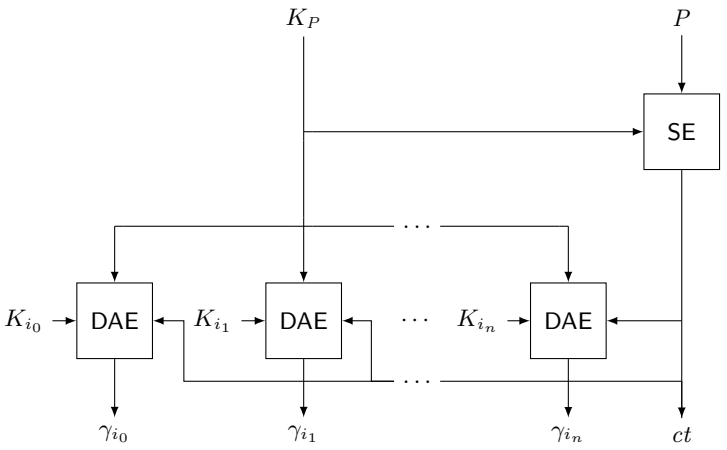

Fig. 7. Encryption Procedure with $Y \coloneqq \{y_1, \ldots, y_n\}$ and $i_j \coloneqq (y_j, t)$.

$\mathsf{Receive}(L_K, \gamma)$: To recover the payload of a ciphertext γ,

1. parse $\gamma = (t, Y, ((y, \gamma_{y,t})_{y \in Y}, ct))$
2. retrieve from L_K a key $K_{y,t}$ for a zone $y \in Y$
3. $K_P \leftarrow \mathsf{DAE.Dec}(K_{y,t}, ct, \gamma_{y,t})$
4. return $P \leftarrow \mathsf{SE.Dec}(K_P, ct)$.

[4] We use a deterministic authenticated encryption scheme as a key-wrapping algorithm should in practice not rely on nonces [22].

Identity Escrow. If needed, the issuer can recover the identity of a vehicle that sent an authenticated key request or response during an execution of protocol Enter. He does so by executing the opening protocol of DGSA.

$\mathsf{Open}(sk_{\mathcal{I}}, st_{\mathcal{I}}, m)$: parse message m as (z, t, m', tok) with $m' \in \{ek, ct\}$ and return identity $\mathcal{V}/\perp \leftarrow \mathsf{DGSA.Open}(sk_{\mathcal{I}}, st_{\mathcal{I}}, (z, t, m'), e(t), tok)$.

Distributed Identity Resolution. In our scheme, the issuer can alone deanonymize messages sent during executions of protocol Enter, making it a single point of failure. To distribute the opening capabilities over several authorities, one can instead use our DGS+A scheme with threshold opening (see Section 3.2.2) so that at least a threshold number of authorities must collaborate to link a message to a vehicle long-term credential.

4.3.2 Correctness & Security.

$\mathcal{Z}$ is correct and satisfies the security requirements introduced in Section 4.2. The full proofs of the following theorems are given in Appendix D.

Theorem 4 (Correctness). *$\mathcal{Z}$ is correct if the signature scheme SIG, the DGS+A scheme DGSA, the public-key encryption scheme PKE, the symmetric encryption scheme SE and the authenticated encryption scheme DAE are correct.*

Proof. If the signature scheme SIG is correct, vehicles that honestly execute the vehicle enrollment algorithm obtain a certificate that is accepted in the authorization protocol with probability 1. If scheme DGSA is also correct, the credentials obtained during the authorization protocol in an epoch allows the vehicles to generate authentication tokens that are later accepted when they enter new zones in the same epoch with overwhelming probability. If the encryption scheme PKE is correct, during the Enter protocol for a zone in a given time period, the same vehicles can successfully unwrap the authenticated-encryption key for the zone in that time period. The correctness of the symmetric encryption scheme SE and that of the authenticated encryption scheme DAE then suffice to conclude that scheme $\mathcal{Z}$ is correct.

Theorem 5 (PH-CCA Security). *$\mathcal{Z}$ is PH-CCA secure if SIG is EUF-CMA secure, if DGSA satisfies traceability, if SE is IND-CPA secure, if PKE is IND-CPA secure, and if DAE satisfies privacy and authenticity.*

Proof (Sketch). Under the assumptions of the theorem, the PH-CCA security of $\mathcal{Z}$ can be proved via the following hybrid argument. Let $\mathcal{A}$ be an adversary for the PH-CCA security distinction experiment (i.e., $\mathcal{A}$ tries to tell apart the PH-CCA challenger $\mathcal{C}_0^{\mathrm{ph-cca}}$ that encrypts P_0 and the challenger $\mathcal{C}_1^{\mathrm{ph-cca}}$ that encrypts P_1). Denote by Y^* its challenge zone set. Number the zones in Y^* from 1 to $n_{Y^*} \coloneqq |Y^*|$, i.e., $Y^* = \{y_1, \ldots, y_{n_{Y^*}}\}$. For $i = 0, \ldots, n_{Y^*}$, consider the hybrid algorithm Δ_i that proceeds exactly like the PH-CCA challenger $\mathcal{C}_0^{\mathrm{ph-cca}}$, except that to compute the challenge ciphertext, it encrypts the first i zones with a payload key K' and the remaining $z - i$ zones with another key K, and encrypts

P_0 with K. Consider also, for $i = 0, \ldots, n_{Y^*}$, the hybrid algorithm Δ'_i that proceeds exactly like the PH-CCA challenger $\mathcal{C}_1^{\text{ph-cca}}$, except that to compute the challenge ciphertext, it encrypts the first i zones with a payload key K and the remaining $z - i$ zones with another key K', and encrypts P_1 with K. By definition, $\Delta_0 = \mathcal{C}_0^{\text{ph-cca}}$, the challenger that encrypts P_0 and $\Delta'_{n_{Y^*}} = \mathcal{C}_1^{\text{ph-cca}}$, the challenger that encrypts P_1. To show that $\mathcal{A}$ has a negligible advantage in the PH-CCA distinction experiment, it suffices to show that the advantage of $\mathcal{A}$ in distinguishing two consecutive hybrids is negligible.

If adversary $\mathcal{A}$ can distinguish Δ_i from Δ_{i+1}, then it can be used to win the privacy game for DAE by having the reduction algorithm choose a time period $\tilde{t}$ uniformly at random and set the challenger key as the key for zone y_{i+1} in time $\tilde{t}$.

For an Enter query on $(\mathcal{V}, y_{i+1}, \tilde{t}, \mathit{role})$ for an honest vehicle $\mathcal{V}$ and role the role of a responding vehicle, upon receiving $(y_{i+1}, \tilde{t}, \mathit{ek}, \mathit{tok})$ from $\mathcal{A}$, the reduction algorithm first determines whether it comes from a non-corrupt vehicle identity in the same protocol execution, i.e., whether $\mathcal{A}$ is simply performing a passive attack by relaying a message from a non-corrupt vehicle identity.

If so, then the reduction algorithm, upon receiving $(y_{i+1}, \tilde{t}, \mathit{ek}, \mathit{tok})$ from $\mathcal{A}$, encrypts a random message instead of K with PKE. The IND-CPA security of PKE is important here to argue for indistinguishability between the two consecutive hybrids.

If $(y_{i+1}, \tilde{t}, \mathit{ek}, \mathit{tok})$ does not come from a non-corrupt vehicle in the same protocol execution, it is an active attack. The reduction algorithm then aborts the protocol execution.

In the event in which $\mathcal{A}$ wins the PH-CCA game, if $t^* = \tilde{t}$ (the reduction algorithm will abort if it is not the case), with t^* the challenge time period, the winning conditions imply that no vehicle $\mathcal{V}_j \in \mathcal{L}_{\mathsf{corrupt}}$ can be authorized in $e(\tilde{t})$, so that

- either there exists a vehicle identity $\mathcal{W}$ such that $(\mathcal{W}, e(\tilde{t})) \in \mathcal{L}_{\mathsf{auth}}$ but $\mathcal{W} \notin \mathcal{L}_{\mathsf{honest}}$ (and also not in $\mathcal{L}_{\mathsf{corrupt}}$), i.e., it has obtain a credential for $e(\tilde{t})$ but has never been enrolled neither as an honest vehicle nor a corrupt one; and it happens with negligible probability if SIG is EUF-CMA secure
- or no such vehicle exists and the token sent by the adversary can be valid w.r.t. to $pk_{\mathcal{I}}$ and $e(\tilde{t})$ with only negligible probability if DGSA satisfies traceability.

Therefore, by aborting the protocol once a token is received from $\mathcal{A}$ during the Enter protocol execution, The reduction algorithm is computationally indistinguishable from both Δ_i and Δ_{i+1}.

Furthermore, the privacy of DAE can be reduced to the computational indistinguishability of Δ'_i and Δ'_{i+1} in the very same manner.

Note also that the IND-CPA security of SE can be reduced to the computational indistinguishability of $\Delta_{n_{Y^*}}$ and Δ'_0. The reduction algorithm can set K as the challenger key, and forward the challenge tuple (P_0, P_1) at the challenge phase.

The advantage of $\mathcal{A}$ in the PH-CCA game is therefore at most, up to a negligible factor, $2n_{Y^*}|T|$ times the advantage of a reduction algorithm (running $\mathcal{A}$ as a subroutine) in the privacy game for DAE plus its advantage in the IND-CPA game for SE.

Therefore, if SE is IND-CPA secure and DAE satisfies privacy and authenticity, then $\mathcal{Z}$ must be PH-CCA secure. See Appendix D for a full proof.

Theorem 6 (Anonymity). *The ZE scheme $\mathcal{Z}$ satisfies anonymity if the DGS+A scheme DGSA satisfies anonymity and if SIG is EUF-CMA secure.*

Proof (Sketch). Under the assumptions of the theorem, the anonymity of $\mathcal{Z}$ can be proved via the following hybrid argument. Let $\mathcal{A}$ be an adversary for the ZE anonymity game that makes q Enter^* queries. One can assume that $q > 0$. Indeed, an adversary which wins the game with $q = 0$ can always be run as a sub-routine by an adversary which makes one arbitrary Enter^* query. For $i = 0, \ldots, q$, let Δ_i be an algorithm that proceeds exactly that the ZE anonymity game challenger, except that to answer the (*) queries with a bit d up to the ith Enter^*, it uses $\mathcal{V}_{1-d}$. For the remaining (*) queries (including the remaining $q - i$ Enter^*), it uses $\mathcal{V}_d$.

By definition, if $\mathcal{C}_b$ denotes the ZE anonymity challenger that uses $\mathcal{V}_b$, then $\Delta_0 = \mathcal{C}_0$ and $\Delta_q = \mathcal{C}_1$. The advantage of $\mathcal{A}$ in the ZE anonymity game is therefore at most q times its advantage in distinguishing Δ_i from Δ_{i+1} for any $0 \leq i \leq q-1$. However, if $\mathcal{A}$ can distinguish Δ_i from Δ_{i+1}, then it can be used to win the DGS+A anonymity game.

At the challenge phase, after the adversary outputs two challenge vehicle identities $\mathcal{V}_0$ and $\mathcal{V}_1$, the simulator, further denoted $\mathcal{S}$, randomly chooses a zone–time pair (z, t) such that both vehicles are authorized in $e(t)$. To answer the first i Enter^* queries for a bit d, the simulator asks for a fresh token on $\mathcal{V}_{1-d}$. To answer Exit^*, Send^* and $\mathsf{Receive}^*$ queries for a bit d, the simulator uses the state of $\mathcal{V}_d$ that it locally maintains. For the $i+1$th Enter^* query, if the vehicle is not in (z, t), the simulator aborts. If it is in (z, t), the simulator generates a key ek for PKE, sends $(\mathcal{V}_0, \mathcal{V}_1, e, (z, t, ek))$ to the DGS+A anonymity challenger and uses the challenge token tok^* to answer the query. For the remaining $q - i - 1$ (*) queries, the simulator uses $\mathcal{V}_d$. Note that the winning conditions enforce that neither $\mathcal{V}_0$ nor $\mathcal{V}_1$ can be corrupt throughout the game, and that $\mathcal{A}$ cannot make any opening query on any message exchanged during executions of protocol Enter with a challenge oracle.

At the end of the game, $\mathcal{A}$ outputs a decision bit b' to $\mathcal{S}$. If $\mathcal{A}$ has never queried the oracle to enter (z, t) with one of the challenge vehicles, then $\mathcal{S}$ returns $\perp$ to $\mathcal{C}$, otherwise $\mathcal{S}$ forwards b' to $\mathcal{C}_{\mathcal{Z},b}$. The advantage of $\mathcal{A}$ in distinguishing Δ_i from Δ_{i+1} is then at most $|Z||T|$ times the advantage of $\mathcal{S}$ in the anonymity game, running $\mathcal{A}$ as a subroutine. As $1/q$ is non-negligible (since $\mathcal{A}$ is efficient), the theorem follows. See Appendix D for a full proof.

Theorem 7 (Traceability). *The ZE scheme $\mathcal{Z}$ satisfies traceability if the DGS+A scheme DGSA satisfies traceability, if PKE is IND-CPA secure and if SIG is EUF-CMA secure.*

Proof (Sketch). The unforgeability of SIG ensures that only enrolled vehicles can be authorized in epochs. If the key K_{z^*,t^*} output by the adversary is in the list of keys of an honest vehicle $\mathcal{V}$, though the adversary has never obtained it by corrupting an honest vehicle, then either there exists a message exchanged during an execution of protocol Enter for (z^*, t^*) that can be traced back to the adversary or no such message exists.

If no such message exists, then the traceability of the scheme can be reduced to the IND-CPA security of PKE since the adversary only ever sees encryption of K_{z^*,t^*}.

If such a message exists, then the winning conditions imply that the vehicle identity to which it traces back to was never corrupt (whether from the beginning or later) or is corrupt but was never authorized in epoch $e(t^*)$.

Since such a message exists and that an honest vehicle $\mathcal{V}$ knows K_{z^*,t^*}, then the adversary must have computed an authentication token that was accepted by an honest vehicle (not necessarily $\mathcal{V}$, but at least one which shares K_{z^*,t^*} with $\mathcal{V}$).

If the token traces back to a vehicle that was never corrupt, then either it was enrolled or it was not.

(1a) If the vehicle was enrolled and authorized in $e(t^*)$, then either the adversary simply replayed a message between honest vehicles, in which case the traceability of $\mathcal{Z}$ can be reduced to the IND-CPA security of PKE, or the adversary forged a token that opens to an honest vehicle that never computed it, in which case the traceability of $\mathcal{Z}$ can be reduced to the traceability of DGSA.

(1b) If the vehicle was enrolled but not authorized in $e(t^*)$, then the traceability of $\mathcal{Z}$ can be reduced to the traceability of DGSA.

(1c) If the vehicle was not enrolled but was authorized in $e(t^*)$, then the adversary forged a certificate for it and traceability of $\mathcal{Z}$ can be reduced to the unforgeability of SIG.

(1d) If the vehicle was not enrolled and not authorized in $e(t^*)$, then the traceability of $\mathcal{Z}$ can be reduced to the traceability of DGSA.

If the message traces back to a vehicle that was corrupt but not authorized in $e(t^*)$, then the traceability of $\mathcal{Z}$ can once again be reduced to the traceability of DGSA. See Appendix D for a full proof.

Theorem 8 (Ciphertext Integrity). *The ZE scheme $\mathcal{Z}$ satisfies ciphertext integrity if* DAE *satisfies authenticity, if* SIG *is EUF-CMA secure, if* DGSA *satisfies traceability, and if* PKE *is IND-CPA secure.*

Proof (Sketch). Assuming that SIG is EUF-CMA secure and that DGSA is traceable, the ciphertext integrity of $\mathcal{Z}$ can be reduced to the authenticity of DAE as follows. The simulator guesses a zone–time pair that will be active for the honest receiving vehicle $\mathcal{V}$ with which the adversary wins the game, and implicitly sets the key of the privacy challenger as the key for that zone–time pair. The simulator generates keys for the other zones itself.

As in the proof of the PH-CCA security of $\mathcal{Z}$, executions of protocol Enter can be perfectly simulated for receiving vehicles under the assumptions that SIG is EUF-CMA secure and that DGSA is traceable.

Whenever the adversary queries the oracle to send a payload to the guessed zone–time pair, the simulator generates a payload key and queries the challenger to wrap with its key, and can thereby answer the query.

Ultimately, if the adversary can produce a new ciphertext for the guessed zone–time pair and that is accepted by $\mathcal{V}$, then the simulator wins the ciphertext-integrity game for DAE by outputting the part of the ciphertext that encrypts the payload key under the key of the guessed pair. See Appendix D for a full proof.

4.4 Efficiency & Comparison

We now describe how the building blocks of our ZE scheme can be instantiated such that the bandwidth constraint of 300 Bytes per message can be satisfied. We then discuss some design choices for the C-ITS deployment and compare it to the current C-ITS proposal.

4.4.1 Efficiency. To instantiate our ZE scheme at a 128-bit security level, we propose

- SIG as the BLS signature scheme [11] since no zero-knowledge proof must be computed during enrollment and authorization. On a Cocks–Pinch pairing curve [34] defined over a field of order 2^{544} and with embedding degree 8, group elements in $\mathbb{G}$ and $\tilde{\mathbb{G}}$ respectively take 68 Bytes and 136 Bytes (using their quadratic twists which have degree 4 [34]) for a group of 256-bit order. Therefore, vehicle certificates, each of which consist of a pair of keys and a signature, are 236 Bytes long.
- DGSA as the DGS+A scheme of Section 3.2. Authentication tokens sent during protocol Enter are then 246 Bytes.
- PKE as the Hash-ElGamal encryption scheme on the 256-bit first group of the previous Cocks–Pinch curve. A public key is a group element in $\mathbb{G}$, and a ciphertext consist of a group element and a bit string of same length as the plaintext (a 128-bit DAE zone key).
- SE as AES-CTR (Counter Mode) with 128-bit keys.
- DAE as AES-128-GCM-SIV [48, Section 5].

The complexity of the opening algorithm is the same as for DGSA, i.e., it grows linearly in the number of enrolled vehicles. This makes tracing expensive but allows for short authentication tokens, which is the appropriate trade-off for V2V communication in which CAMs should be short and tracing only be done in case of exceptional events, e.g., an accident or to revoke the key of a rogue device.

	p (bits)	Sec. Lvl. (bits)	Req. (B)	Resp. (B)
CP8-544 [34]	256	131 [34]	284	300
BLS12-446	299	132 [35]	263	279
FM12-446 [28]	296	136 [33]	261	277

Table 1. Sizes of Key Requests and Responses with various Curves and their associated field sizes and security levels.

4.4.2 C-ITS Deployment and Comparison. Suppose that the road network is divided into hexagonal zones, and that a vehicle broadcasts messages to the zone it currently is and its 6 neighboring zones, i.e., to 7 zones in total. With the parameters of Section 4.4.1, the ciphertexts of our ZE scheme (ignoring the time-period and zone indicators) consist of 7 AES-128-GCM-SIV ciphertexts (256 bits each) and an AES ciphertext (128 bits), amounting to 240 Bytes; well within the 300 Bytes bandwidth requirements for C-ITSs. With payloads of 128 bits, it corresponds to a cryptographic overhead of 224 Bytes.

For messages during protocol Enter, the cryptographic overhead of our scheme is a PKE public key (83 Bytes) and an authentication token in a key request, and a PKE encryption of the DAE key (16 Bytes) and an authentication token in a key response. With tokens of size 216 Bytes, this yields a total of 284 Bytes for request and 300 Bytes for response messages.

Table 1 gives, for various curve choices, the security level, the size of certificates and the cryptographic overhead for key requests and key responses. Note that Cocks–Pinch curves are not vulnerable to TNFS attacks [2, 3, 33, 35, 38–40] which affects the security of some curves constructed from different methods, and these attacks may be improved in the future. The CP8-544 curve therefore seems to be the safest choice in terms of security at a 128-bit level or higher. On the other hand, although operations on CP8-544 are very efficient [33], the BLS12-446 and FM12-446 curves are the most efficient pairing curves [33] at that security level.

With our ZE scheme used in combination with our DGS+A scheme, a vehicle can create an unlimited number of unlinkable (even by other vehicles thanks to the anonymity of group signatures) signatures by downloading a single credential in every epoch. Compared to the current C-ITS proposals [25, 42], DGS+A combines the equivalent of an infinite pseudonym pool size with the negligible costs of downloading and storing a single constant-size credential per epoch. The latter aspect is a significant improvement not only in terms of storage, but also communication: in the current C-ITS proposals, vehicles have to spread out requests for individual pseudonyms over time, rather than downloading them in batches, to avoid that issuers are able to link the pseudonyms belonging to the same vehicle.

Moreover, with our scheme, each CAM carries only 64 Bytes more of cryptographic overhead than the current proposals with ECDSA signatures (160 Bytes), for all the additional security and privacy benefits. Besides, symmetric cryptography is typically significantly faster than elliptic-curve operations.

	Zone Encryption	C-ITS Proposal
Encrypted CAM	✓	✘
Anonymity	✓	✘
Pseudonyms per week	unlimited	100 (EU) / 20 (US)
CAM Authentication	DAE	ECDSA
Overhead per CAM	224 Bytes	160 Bytes
+ per entered Zones	284/300 Bytes	—

Table 2. Comparison of zone encryption to current C-ITS proposals at a 128-bit security level. "Pseudonyms" refers to the number of unlinkable authentication tokens a vehicle can generate per epoch.

Therefore, the verification of the authenticity of incoming CAMs is also faster with our scheme thanks to the use of a deterministic authenticated (symmetric) encryption scheme to encrypt payload keys.

Table 2 provides a brief overview of the core differences between zone encryption and the current C-ITS proposal.

4.5 Threat Model and Design Choices

By nature, V2X communication is an open system that enables all participating vehicles to communicate with each other; the security that one can hope to achieve is therefore also inherently limited. We discuss our threat model here in more detail and provide some insights into our design choices.

Passive vs. Active Eavesdropping. Because all vehicles must be able to decrypt messages from other close vehicles, no system can protect against eavesdropping attacks by insiders that have access to legitimate vehicle credentials and roam around to actively listen into nearby zones. However, our zone encryption scheme does force such an attacker to actively participate in zone key exchange protocols, thereby exposing its credentials to being traced and revoked by authorities. Doing so may not be straightforward in practice, but it is a considerable step up from the passive and covert eavesdropping attacks that are trivial to deploy in the current C-ITS proposals where all vehicles broadcast *plaintext* messages.

As discussed in the introduction, zone encryption enables authorities to considerably increase the manufacturing cost of black-market decryption devices, hopefully beyond the point of economical feasibility for ordinary criminals. Also as discussed, the threat of abusing traffic infrastructure for mass surveillance can be limited by giving infrastructure that has no need for privacy, nor to decrypt CAM traffic, a different type of credentials that cannot be used to obtain zone keys. Nevertheless, mass surveillance by a powerful adversary remains possible, e.g., through a network of (parked or moving) vehicles, or by road infrastructure that does have a legitimate need to read CAM traffic. Therefore, even when using zone encryption, the information in CAMs must still be minimized as much

as possible. Maintaining a pool of vehicles or infrastructure for performing such eavesdropping attacks becomes more expensive with our solution though, because by forcing the adversary to actively participate in zone key exchanges, suspicious behavior can be traced and the corresponding credentials revoked.

Cloning & Insider Attacks. An adversary that compromises and clones the keys of a vehicle, short-term credentials, or even its long-term certificate obviously allows the adversary to "impersonate" that vehicle. For zone keys and short-term (DGSA) credentials, the impact is limited by the timed aspect of zone encryption to, e.g., 15 minutes and a week, respectively. Corruption of long-term credentials is more damaging, but the certificate is also likely to enjoy stronger protection, e.g., from trusted hardware. Furthermore, the issuer of short-term credentials could monitor and detect suspicious use of long-term credentials, such as too frequent requests or requests from very distant locations, and block or revoke the long-term credential accordingly.

Apart from decrypting CAM traffic from other vehicles, the adversary can also use the compromised credentials to broadcast fake information. Indeed, the ciphertext integrity property of zone encryption protects against malicious information being inserted by outsiders, but not by inside attackers. Moreover, since zone keys are shared among vehicles, it is nearly impossible to exactly identify the culprit vehicle. This is indeed a drawback with respect to the current proposal for C-ITSs where each CAM message is signed. However, if an abnormal event occurs in a zone, the issuer, can with our scheme, reduce the list of suspects to just the devices the have entered the zone at the time of the event. Besides, with the current proposal, the issuer can only fully trace back malicious information, not prevent it, and information from CAMs will always be double-checked by other sensors such as cameras and LiDARs. All in all, we therefore argue that the privacy advantages of zone encryption outweigh not being able to directly identify malicious senders.

Alternative Authentication Mechanisms. Our zone-encryption design uses group signatures to authenticate messages of protocol Enter, which guarantees privacy for vehicles while enabling the issuer to trace and revoke compromised credentials. As priorly discussed, the capability to trace vehicles can be distributed over multiple authorities by using a group signature scheme with threshold opening (see Appendix C).

One could consider resorting to different authentication mechanisms, such as anonymous credentials [16]. Generic anonymous credential schemes have been proposed for use in V2X applications [20, 41, 51] but typically have much larger signature sizes.

In theory, limited-spending techniques [15] for such schemes might seem suitable to avoid the credential cloning and allow to trace compromised keys, instead of requiring a trusted opening authority as in our solution. However, doing so would require some authority to collect and cross-check *all* pseudonyms across *all* zones to identify overused credentials. Apart from being detrimental to privacy, such data collection is infeasible in a continent-scale V2X communication system with tens of millions of zones and billions of messages exchanged.

In comparison, our opening algorithm runs on a *single* authenticated message sent during a key request or a key response. There is no need for a synchronization between all zones. The anonymous messages exchanged in a zone can easily be recoverable if e.g., road infrastructures are required to maintain a record of messages exchanged in the zone they are in for a certain duration (e.g., a day), or if vehicles maintain a record of the messages they exchange when requesting or communicating zone keys for a short amount of time (e.g., an hour).

Some anonymous attestation mechanisms such as EPID [13] support signature-based revocation, meaning that the key behind a signature can be revoked by adding the signature to a blacklist. However, the size and/or the verification time of such non-revocation proofs grows linearly with the number of revoked users, which is impractical in a V2X system with hundreds of millions of vehicles.

Finally, note that our authentication mechanism could be replaced with a quantum-safe one if large-scale quantum computers were to be built in the future. However, no quantum-safe scheme known today is even close to fitting the 300-Byte limit at a 128-bit security level.

5 Conclusion

We presented two cryptographic constructions that are tailored to the needs of C-ITSs and can enhance their privacy properties. First, our compact DGS+A scheme has fully anonymous 216-Byte authentication tokens, while needing only a single constant-size credential per epoch. This is an important improvement over the limited pseudonym pool sizes and their expensive reloading protocols that are currently proposed for deployment. Secondly, our zone encryption scheme enables efficient encryption of position beacon messages, protecting their content from eavesdroppers. Our two techniques are best used in combination, but can be used independently as well: one can combine zone encryption with any other anonymous authentication scheme, and one can use our DGS+A scheme without using zone encryption.

Despite these improvements, our schemes should still be used with some care. Actively participating eavesdroppers can still receive all communication, so minimizing the information contained in CAMs for the particular envisaged applications remains crucial. Besides, authentication tokens leak the identity of the issuers, i.e., vehicles are only anonymous among other vehicles with the same issuers. This is easily circumvented at an organizational level by letting all vehicles use the same (e.g., country-wide) issuers. If that is not possible, technical solutions would involve delegatable credentials [4], but their tokens are too long to be used in C-ITS.

Moreover, there are further privacy limitations inherent to V2X communication. Vehicles can still be tracked by other means than CAMs. One could for instance fingerprint radio transmitters and antennas, use side-channel analysis, or even cameras and image processing to track vehicles [54]. However, the possibility of such attacks does not mean that privacy for vehicular communication should be entirely forgone. In fact, similar arguments can be made for most ap-

plications, e.g., users' online activities which can be fingerprinted through the hardware they use. Still, efficient privacy-preserving protocols are still sought after instead abandoning the idea of online privacy altogether.

Lastly, designing efficient and robust key-agreement strategies when vehicles enter zones, and determining appropriate time frames for reaching key consensus are interesting open engineering problems. We leave these issues to future work as the focus of this paper is on the cryptographic concerns of secure and privacy-friendly V2X communication.

Acknowledgements. This work supported by the CHIST-ERA USEIT project.

References

1. Article 29 Data Protection Working Party. Opinion 03/2017 on processing personal data in the context of cooperative intelligent transport systems (C-ITS) - wp252. `http://ec.europa.eu/newsroom/article29/item-detail.cfm?item_id=610171`, 2017.
2. R. Barbulescu and S. Duquesne. Updating key size estimations for pairings. *Journal of Cryptology*, 32(4):1298–1336, Oct 2019.
3. R. Barbulescu, P. Gaudry, and T. Kleinjung. The tower number field sieve. In T. Iwata and J. H. Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 31–55. Springer, Heidelberg, Nov. / Dec. 2015.
4. M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. Randomizable proofs and delegatable anonymous credentials. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 108–125. Springer, Heidelberg, Aug. 2009.
5. M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In A. Juels, R. N. Wright, and S. Vimercati, editors, *ACM CCS 06*, pages 390–399. ACM Press, Oct. / Nov. 2006.
6. M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In A. Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, Heidelberg, Feb. 2005.
7. F. Benhamouda, J. Camenisch, S. Krenn, V. Lyubashevsky, and G. Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In P. Sarkar and T. Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 551–572. Springer, Heidelberg, Dec. 2014.
8. P. Bichsel, J. Camenisch, G. Neven, N. P. Smart, and B. Warinschi. Get shorty via group signatures without encryption. In J. A. Garay and R. D. Prisco, editors, *SCN 10*, volume 6280 of *LNCS*, pages 381–398. Springer, Heidelberg, Sept. 2010.
9. N. Bißmeyer, S. Mauthofer, J. Petit, M. Lange, M. Moser, D. Estor, M. Sall, M. Feiri, R. Moalla, M. Lagana, and F. Kargl. V2X security architecture v2. PRESERVE Project, Deliverable D1.3, 2014.
10. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, Aug. 2004.
11. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, Dec. 2001.

12. D. Boneh and M. Naor. Traitor tracing with constant size ciphertext. In P. Ning, P. F. Syverson, and S. Jha, editors, *ACM CCS 08*, pages 501–510. ACM Press, Oct. 2008.
13. E. Brickell and J. Li. Enhanced privacy ID from bilinear pairing. Cryptology ePrint Archive, Report 2009/095, 2009. `http://eprint.iacr.org/2009/095`.
14. J. Camenisch, M. Drijvers, A. Lehmann, G. Neven, and P. Towa. Short threshold dynamic group signatures. Cryptology ePrint Archive, Report 2020/016, 2020. `https://eprint.iacr.org/2020/016`.
15. J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich. How to win the clonewars: Efficient periodic n-times anonymous authentication. In A. Juels, R. N. Wright, and S. Vimercati, editors, *ACM CCS 06*, pages 201–210. ACM Press, Oct. / Nov. 2006.
16. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In S. Cimato, C. Galdi, and G. Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 268–289. Springer, Heidelberg, Sept. 2003.
17. J. Camenisch, G. Neven, and M. Rückert. Fully anonymous attribute tokens from lattices. In I. Visconti and R. D. Prisco, editors, *SCN 12*, volume 7485 of *LNCS*, pages 57–75. Springer, Heidelberg, Sept. 2012.
18. J. Y. Choi, M. Jakobsson, and S. Wetzel. Balancing auditability and privacy in vehicular networks. In *Q2SWinet'05*, pages 79–87, 2005.
19. B. Chor, A. Fiat, and M. Naor. Tracing traitors. In Y. Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 257–270. Springer, Heidelberg, Aug. 1994.
20. J. M. de Fuentes, L. González-Manzano, J. Serna-Olvera, and F. Veseli. Assessment of attribute-based credentials for privacy-preserving road traffic services in smart cities. *Personal and Ubiquitous Computing*, 21(5):869–891, 2017.
21. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In G. Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 307–315. Springer, Heidelberg, Aug. 1990.
22. M. Dworkin. Request for review of key wrap algorithms. Cryptology ePrint Archive, Report 2004/340, 2004. `http://eprint.iacr.org/2004/340`.
23. M. J. Dworkin. Recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac. `https://www.nist.gov/`, 2007.
24. M. J. Dworkin. Recommendation for block cipher modes of operation: The cmac mode for authentication. `https://www.nist.gov/`, 2016.
25. Intelligent transport systems (ITS); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service. Technical Report ETSI EN 302 637-2 V1.3.1, ETSI, 2014.
26. Intelligent transport systems (ITS); security; ITS communications security architecture and security management. Technical Report ETSI TS 102 940 v1.3.1, ETSI, 2018.
27. Intelligent transport systems (ITS); security; trust and privacy management. Technical Report ETSI TS 102 941 V1.3.1, ETSI, 2019.
28. G. Fotiadis and C. Martindale. Optimal tnfs-secure pairings on elliptic curves with composite embedding degree. Cryptology ePrint Archive, Report 2019/555, 2019. `https://eprint.iacr.org/2019/555`.
29. C. Freitag, J. Katz, and N. Klein. Symmetric-key broadcast encryption: The multi-sender case. In S. Dolev and S. Lodha, editors, *Cyber Security Cryptography and Machine Learning - First International Conference, CSCML 2017*, volume 10332 of *Lecture Notes in Computer Science*, pages 200–214. Springer, 2017.

30. J. Freudiger, M. Raya, M. Félegyházi, P. Papadimitratos, and J.-P. Hubaux. Mix-zones for location privacy in vehicular networks. *ACM Workshop on Wireless Networking for Intelligent Transportation Systems (WiN-ITS)*, 2007.
31. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In J. Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 295–310. Springer, Heidelberg, May 1999.
32. L. Gollan and C. Meinel. Digital signatures for automobiles. In *Systemics, Cybernetics and Informatics (SCI) 2002*, pages 1–5, 2002.
33. A. Guillevic. A short-list of stnfs-secure pairing-friendly curves at the 128-bit security level. Cryptology ePrint Archive, Report 2019/1371, 2019. `https://eprint.iacr.org/2019/1371`.
34. A. Guillevic, S. Masson, and E. Thomé. Cocks–pinch curves of embedding degrees five to eight and optimal ate pairing computation. 2019.
35. A. Guillevic and S. Singh. On the alpha value of polynomials in the tower number field sieve algorithm. Cryptology ePrint Archive, Report 2019/885, 2019. `https://eprint.iacr.org/2019/885`.
36. J. Hubaux, S. Capkun, and J. Luo. The security and privacy of smart vehicles. *IEEE Security & Privacy*, 2(3):49–55, 2004.
37. P. Kamat, A. Baliga, and W. Trappe. An identity-based security framework for VANETs. In *Proceedings of the 3rd International Workshop on Vehicular Ad Hoc Networks*, VANET '06, pages 94–95, New York, NY, USA, 2006. ACM.
38. T. Kim and R. Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 543–571. Springer, Heidelberg, Aug. 2016.
39. T. Kim and J. Jeong. Extended tower number field sieve with application to finite fields of arbitrary composite extension degree. In S. Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 388–408. Springer, Heidelberg, Mar. 2017.
40. A. Menezes, P. Sarkar, and S. Singh. Challenges with assessing the impact of nfs advances on the security of pairing-based cryptography. Cryptology ePrint Archive, Report 2016/1102, 2016. `https://eprint.iacr.org/2016/1102`.
41. G. Neven, G. Baldini, J. Camenisch, and R. Neisse. Privacy-preserving attribute-based credentials in cooperative intelligent transport systems. In *2017 IEEE Vehicular Networking Conference, VNC 2017*, pages 131–138. IEEE, 2017.
42. U. D. of Transportation; National Highway Traffic Safety Administration. Notice of proposed rulemaking for federal motor vehicle safety standards; V2V communications. *Federal Register*, 82(8), 2017.
43. J. Petit, F. Schaub, M. Feiri, and F. Kargl. Pseudonym schemes in vehicular networks: A survey. *IEEE Communications Surveys and Tutorials*, 17(1):228–255, 2015.
44. D. Pointcheval and O. Sanders. Short randomizable signatures. In K. Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 111–126. Springer, Heidelberg, Feb. / Mar. 2016.
45. D. Pointcheval and O. Sanders. Reassessing security of randomizable signatures. In N. P. Smart, editor, *CT-RSA 2018*, volume 10808 of *LNCS*, pages 319–338. Springer, Heidelberg, Apr. 2018.
46. L. Reyzin, A. Lysyanskaya, V. Shmatikov, A. D. Smith, and Center for Democracy & Technology. Comments on NHTSA notice of proposed rule for FMVSS no. 150, V2V communications. `https://cdt.org/files/2017/04/FMVSS150CommentsOnPrivacy-as-submitted.pdf`, 2017.

47. M. Riley, K. Akkaya, and K. Fong. Group-based hybrid authentication scheme for cooperative collision warnings in VANETs. *Security and Communication Networks*, 4(12):1469–1482, 2011.
48. P. Rogaway and T. Shrimpton. A provable-security treatment of the key-wrap problem. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, Heidelberg, May / June 2006.
49. F. Schaub, Z. Ma, and F. Kargl. Privacy requirements in vehicular communication systems. In *2009 International Conference on Computational Science and Engineering*, volume 3, pages 139–145, Aug 2009.
50. A. Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, Nov. 1979.
51. A. Singh and H. S. Fhom. Restricted usage of anonymous credentials in vehicular ad hoc networks for misbehavior detection. *Int. J. Inf. Sec.*, 16(2):195–211, 2017.
52. K. Sjöberg, P. Andres, T. Buburuzan, and A. Brakemeier. C-ITS deployment in europe - current status and outlook. *CoRR*, abs/1609.03876, 2016.
53. H. Toulni, M. Boudhane, B. Nsiri, and M. Miyara. An adaptive key exchange procedure for VANET. *International Journal of Advanced Computer Science and Applications*, 7(4), 2016.
54. C. Troncoso, E. Costa-Montenegro, C. Diaz, and S. Schiffner. On the difficulty of achieving anonymity for vehicle-2-x communication. *Computer Networks*, 55(14), 2011.
55. W. Whyte, A. Weimerskirch, V. Kumar, and T. Hehn. A security credential management system for V2V communications. In *2013 IEEE Vehicular Networking Conference*, pages 1–8. IEEE, 2013.
56. K. Zeng. Pseudonymous PKI for ubiquitous computing. In *EuroPKI 2006*, volume 4043 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2006.

A Deterministic Authenticated Encryption

Deterministic Authenticated Encryption (DAE) [48] is mainly used in the context of key wrapping, i.e., transmitting a secret key from one party to another. It is suitable for our ZE scheme as fresh keys are usually encrypted only once.

A.1 Security Properties.

(Deterministic) Privacy [48, Appendix B] is modeled via a distinction experiment. In the real game, a key is chosen uniformly at random and the adversary can make (, without loss of generality, non-repeated) encryption queries. In the ideal game, encryption queries are answered with uniformly random bit strings. A DAE scheme satisfies privacy if no efficient adversary has a non-negligible advantage in distinguishing the real game from the ideal game. (Deterministic) Authenticity is formalized via a key at the beginning of which a key is chosen uniformly at random. The adversary can make (non-repeated) encryption queries, and can submit ciphertexts none of which is the output of a previous encryption query; that is to say, it can make forgery attempts. The adversary wins the game as soon as one of those forgery attempts does not fail. A DAE scheme satisfies authenticity if no efficient adversary has non-negligible advantage in winning this game.

A.2 SIV Construction.

Rogaway and Shrimpton constructed a DAE scheme from IV-based encryption schemes and Pseudo-Random Functions (PRFs). They called it the Synthetic-IV (SIV) construction. Their construction requires the ciphertexts of the encryption scheme to be unpredictable if the initialization vector is a uniformly random n-bit string, with n the IV length of the encryption scheme. (The latter property is referred to as privacy of IV-based encryption schemes.) They therefore use a PRF in the SIV construction to compute the initialization vector from the message and the header, so as to make the IV unpredictable.

Note that the PRF must thereby support vectors of bit strings even though most PRFs in the literature are designed to be computed on a single bit string. Of course, in case a PRF must be computed on a vector of bit strings, the string could be concatenated, but it would incur an important efficiency loss [48, Section 5]. Rogaway and Shrimpton consequently proposed a String-to-Vector (S2V) transformation [48, Section 5] of string PRFs to PRFs that directly support string vectors without the efficiency loss of trivial solutions.

Formally, the SIV construction is the following. Let $\mathsf{PRF}\colon \mathcal{K}_1 \times \{0,1\}^{**} \to \{0,1\}^n$ be a pseudo-random function with key space $\mathcal{K}_1$ and the set of vectors of bit strings $\{0,1\}^{**}$ as message space. Consider also $(\mathsf{Enc}, \mathsf{Dec})$ (the setup algorithm is omitted) an IV-based encryption scheme with key space $\mathcal{K}_2$. To generate keys for the DAE scheme, generate independently and uniformly at random two keys K_1 and K_2 from $\mathcal{K}_1$ and $\mathcal{K}_2$ respectively. To encrypt, on the input of K_1, K_2, a header H and a message M, compute $IV \leftarrow \mathsf{PRF}(K_1, (H, M))$ then $C \leftarrow \mathsf{Enc}(K_2, IV, M)$, and output $IV \| C$. To decrypt a ciphertext C, if it is less than n-bit long, abort. Otherwise parse it as $IV \| C'$ with IV the first n bits, and compute $\mathsf{Dec}(K_2, IV, C')$ then $IV' \leftarrow \mathsf{PRF}(K_1, H, M)$. If $IV = IV'$, then output M, otherwise output $\perp$.

Rogaway and Shrimpton proved [48, Theorem 2] that if the IV-based scheme $(\mathsf{Enc}, \mathsf{Dec})$ satisfies privacy and if PRF is a pseudo-random function (i.e., such that its outputs are computationally indistinguishable from uniformly random n-bit strings), then the SIV construction satisfies privacy and authenticity.

To instantiate their construction, one can use the S2V transform of a block cipher such as AES in CMAC mode [24] as PRF and a block cipher in counter (CTR) mode [23] as IV-based encryption scheme.

B Dynamic Group Signatures with Attributes

In this section, we first give formal definitions for the properties that a DGS+A scheme should satisfy. We then prove that our scheme from Section 3.1 satisfies these properties.

B.1 Definition of DGS+A

A DGS+A scheme should satisfy correctness, anonymity and traceability, which we define following the security notions for conventional dynamic group signatures adapted to our setting with attributes.

Correctness. Correctness captures the idea that a truthfully generated authentication token should be accepted by the verification algorithm. Furthermore, if all the algorithms are honestly executed and the opening algorithm is run on a token, then the opening algorithm should return the identity of the user who computed the token. These properties should hold independently of the order in which credentials are issued for user–attribute pairs and with overwhelming probability.

Anonymity. Anonymity ensures that an authentication token *tok* does not reveal any information about the identity of the user that generated if the issuer is honest and *tok* has not been opened. In the game (Figure 8), the adversary is given oracle access to the honest issuer and to the honest users which it can trigger to obtain membership credentials for identities and attributes of its choice, let honest users sign messages, corrupt users and open authentication tokens. At the end of the game, the adversary outputs two honest-user identities id_0^*, id_1^* together with a message m and attributes A^*. The challenge computes a token *tok* for either of the identities and sends it to the adversary. The task of the adversary is to determine the origin of the token.

Definition 7 (Anonymity). *A DGS+A scheme* DGSA *satisfies anonymity if for all efficient adversary $\mathcal{A}$, for all $\lambda \in \mathbb{N}$, the advantage of the adversary*

$$\left| \Pr\left[\mathbf{Exp}^{\text{ano}-0}_{\mathsf{DGSA},\lambda}(\mathcal{A}) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ano}-1}_{\mathsf{DGSA},\lambda}(\mathcal{A}) = 1\right] \right|$$

is negligible in λ.

Experiment $\mathbf{Exp}^{\text{ano}-b}_{\mathsf{DGSA},\lambda}(\mathcal{A})$:
$pp \leftarrow \mathsf{Setup}(1^\lambda)$, $(pk, (sk, st)) \leftarrow \mathsf{KG}(pp)$
$(id_0^*, id_1^*, A^*, m^*, state_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{O}(sk,st,\cdot)}(\mathsf{choose}, pp, pk)$
abort if $\exists d \in \{0,1\}\colon (id_d^*, cred_d, A^*) \notin \mathcal{L}_{\mathsf{joined}}$
$b \in_R \{0,1\}$, $tok^* \leftarrow \mathsf{Auth}(pk, cred_b, m^*)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}(st,sk,\cdot)}(\mathsf{guess}, tok^*, state_{\mathcal{A}})$
return b' if
$(m^*, A^*, tok^*) \notin \mathcal{L}_{\mathsf{opened}}$ and $(id_{0/1}^*, A^*) \notin \mathcal{L}_{\mathsf{corrupt}}$
else abort

Fig. 8. Anonymity Security Experiment for DGS+A schemes. $\mathcal{O} = \{\mathsf{Issue}, \mathsf{Issue.I}, \mathsf{Auth}, \mathsf{Corrupt}, \mathsf{Open}\}$ as defined in Section B.1.

Traceability. Traceability captures the expected unforgeability guarantees of our group signatures. It guarantees that as long as the issuer is honest, for any valid token tok^*, message m^* and attribute set A^*, opening can neither fail nor reveal an incorrect honest identity id. The latter means that the user id either never joined the group w.r.t. A^*, or has joined but never signed the message m^*. The traceability game is defined in Figure 9.

Definition 8 (Traceability). *A DGS+A scheme* DGSA *satisfies traceability if for every efficient adversary* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$*,* $\Pr\left[\mathbf{Exp}^{\text{trace}}_{\mathsf{DGSA},\lambda}(\mathcal{A}) = 1\right] \leq \text{negl}(\lambda)$.

Experiment $\mathbf{Exp}^{\text{trace}}_{\mathsf{DGSA},\lambda}(\mathcal{A})$:
$pp \leftarrow \mathsf{Setup}(1^\lambda)$, $(pk, (sk, st)) \leftarrow \mathsf{KG}(pp)$
$(A^*, m^*, tok^*) \leftarrow \mathcal{A}^{\mathcal{O}(sk,st,\cdot)}(\mathsf{forge}, pp, pk)$
$id^* \leftarrow \mathsf{Open}(sk, st, m^*, A^*, tok^*)$
return 1 if $\mathsf{Vf}(pk, m^*, A^*, tok^*) = 1$ and
1) opening failed, i.e., $id^* = \perp$ or 2) opening is "incorrect", i.e.,
$(id^*, A^*, \cdot) \notin \mathcal{L}_{\mathsf{joined}}$ or
$((id^*, A^*, \cdot) \in \mathcal{L}_{\mathsf{joined}}$ and $(id^*, A^*) \notin \mathcal{L}_{\mathsf{corrupt}}$ and $(id^*, A^*, m^*) \notin \mathcal{L}_{\mathsf{signed}})$
else return 0

Fig. 9. Traceability Security Experiment for DGS+A schemes. $\mathcal{O} = \{\mathsf{Issue}, \mathsf{Issue.I}, \mathsf{Auth}, \mathsf{Corrupt}, \mathsf{Open}\}$ as defined in Section B.1.

Oracles for DGS+A Security Experiments. To define anonymity and traceability for DGS+A schemes, consider the experiments on Figures 8 and 9 respectively. In those experiments, the adversary has access to the oracles defined below.

- $\mathsf{Issue}(sk, st, \cdot)$, on input (id, A), generates a credential for an honest (id, A) with the issuer key sk, i.e., it executes $\langle \mathsf{Issue.U}(pk, id, A) \rightleftharpoons \mathsf{Issue.I}(sk, st, id, A)\rangle \rightarrow \langle cred, st'\rangle$ and adds $(id, A, cred)$ to the list $\mathcal{L}_{\mathsf{joined}}$. If there already is a credential $cred$ for (id, A) in $\mathcal{L}_{\mathsf{joined}}$, oracle Issue simply retrieves it. It eventually returns $cred$
- $\mathsf{Issue.I}(sk, st, \cdot)$, on input (id, A), lets the adversary, in the role of a corrupt user, run an issuance protocol with an honest issuer. That is, it starts an execution of protocol for $\mathsf{Issue.I}(sk, st, id, A)$, and adds $(id, A, \perp)$ to $\mathcal{L}_{\mathsf{joined}}$ and (id, A) to $\mathcal{L}_{\mathsf{corrupt}}$
- $\mathsf{Corrupt}(\cdot)$, on input id, returns all $(id, A, cred) \in \mathcal{L}_{\mathsf{joined}}$, i.e., all the credentials that have been generated for id (with potentially different attribute sets) by the challenger at the time of the query. It also adds all revealed (id, A) to $\mathcal{L}_{\mathsf{corrupt}}$. Notice that this allows an identity that was corrupted to later join the system with a new attribute set, and the pair identity–attribute-set will be considered honest
- $\mathsf{Auth}(pk, \cdot)$, on input (id, A, m), fetches $(id, A, cred) \in \mathcal{L}_{\mathsf{joined}}$ and computes $at \leftarrow \mathsf{Auth}(pk, cred, m)$. It adds (id, A, m) to $\mathcal{L}_{\mathsf{signed}}$ and returns at. If no such credential exists, the oracle replies with $\perp$
- $\mathsf{Open}(sk, st, \cdot)$, on input (m, A, tok), returns to the adversary the opened identity of an attribute token of his choice. That is, it runs the opening algorithm on the inputs, returns the opened identity and adds the tuple (m, A, tok) to the list $\mathcal{L}_{\mathsf{opened}}$.

B.2 Our DGS+A scheme

We here provide a more detailed discussion about the efficiency of our scheme and how it compares to previous dynamic group signature schemes. We also give proofs that it satisfies the correctness and security requirements of Section 3.1.

B.2.1 Correctness and Security of our DGS+A Scheme.

Proof (of Theorem 1). First, a truthfully generated token tok for a message m with a publid key pk and a credential $cred = (id, A, \sigma, e(\sigma_1, \tilde{Y}_{id}), e(\sigma_1, \tilde{Y}_{k+1}))$ is always accepted by the verification algorithm on input (pk, m, A, tok) since $(\sigma_1' = \sigma_1^r, \sigma_2' = \sigma_2^r, a')$ (for $r \in_R \mathbb{Z}_p^*$) is a valid PS signature on (id, A) and the Schnorr signature of knowledge on m of a PS signature on (id, A) is correct. Secondly, since the pair (id, A) is added to the issuer state st during the issuance protocol, id is returned by the opening algorithm which simply tests the validity of $(\sigma_1', \sigma_2', a')$ as a PS signature agains each entry in its state; unless there exists a different pair (id', A') in st, with $id' < id$, such that if $b' = \mathcal{H}_0(id', A')$, then $(\sigma_1', \sigma_2', b')$ is a valid PS signature on (id', A'). This event occurs if and only if $e\big(\sigma_1', \tilde{X}\tilde{Y}_{id}^{id'} \prod_{i=1}^{k} \tilde{Y}_i^{a_i'} \tilde{Y}_{k+1}^{b'}\big) = e(\sigma_2', \tilde{g}) = \big(\sigma_1', \tilde{X}\tilde{Y}_{id}^{id} \prod_{i=1}^{k} \tilde{Y}_i^{a_i} \tilde{Y}_{k+1}^{a'}\big)$. That is, if and only if $y_{id}(id' - id) + \sum_{i=1}^{k} y_i(a_i' - a_i) = y_{k+1}(a' - b')$. However, the issuer chooses y_{k+1}, a' and b' uniformly at random, and independently of the user identities, the attributes and the other secret keys. The equalitiy therefore holds with probability at most $1/p$, which is negligible.

Proof (of Theorem 2). The anonymity of DGSA is proved via a hybrid argument. Denote by $\mathcal{C}_b$ the challenger that uses the credential associated to id_b^* and A^* for $b \in \{0, 1\}$. Let Δ be an algorithm that proceeds exactly like $\mathcal{C}_0$ except for the challenge phase. At the challenge phase, Δ sends two random $\mathbb{G}$ elements as re-randomized group elements of the PS signature on (id_0^*, A^*) and programs the random oracle accordingly.

More precisely, consider an efficient adversary $\mathcal{A}$ for the anonymity experiment. Δ interacts with $\mathcal{A}$ and Whenever $\mathcal{A}$ issues its challenge query, Δ first checks whether it has generated a credential $cred_b$ for both $b \in \{0, 1\}$. If not, it aborts, otherwise it generates $\sigma_1, \sigma_2 \in_R \mathbb{G}^*$ and $s_{id}, s_{a'}, c \in_R \mathbb{Z}_p$, computes

$$u \leftarrow e\left(\sigma_1, \tilde{Y}_{id}\right)^{s_{id}} e\left(\sigma_1, \tilde{Y}_{k+1}\right)^{s_{a'}} \left(e\big(\sigma_2, \tilde{g}\big) e\left(\sigma_1, \tilde{X}^{-1} \prod_{j=1}^{k} \tilde{Y}_j^{-a_j^*}\right)\right)^{c},$$

and programs $\mathcal{H}(u, A^*, m^*, \sigma_1, \sigma_2, pk) \leftarrow c$. Hybrid Δ then sets $\pi \leftarrow (c, s_{id}, s_{a'})$ and returns $(\sigma_1, \sigma_2, \pi)$ to $\mathcal{A}$.

The answer of Δ to the challenge query is computationally indistinguishable from that of $\mathcal{C}_0$ as the distribution of σ_1 and σ_2 in the challenge authentication token computed by $\mathcal{C}$ and Δ are indistinguishable under the DDH assumption in $\mathbb{G}$. However, it remains to argue that the joint distribution of the answers of Δ to the oracle queries are indistinguishable from those of $\mathcal{C}_0$.

In more detail, the indistinguishability of $\mathcal{C}_0$ from Δ can be argued as follows. Consider an algorithm which runs $\mathcal{A}$ as a subroutine and interacts with a challenger that either outputs a Diffie–Hellman tuple or a random tuple. Upon receiving a tuple (g_0, g_1, g_2, g_3), the reduction algorithm generates the other scheme parameters itself and chooses two random challenges identities (id_0^*, id_1^*). To generate a credential for (id_0^*, A) for any attribute set A, the reduction algorithm generates $a', r \in_R \mathbb{Z}_p^*$, sets $\sigma_1 \leftarrow g_0^r$ and $\sigma_2 \leftarrow \sigma_1^{x+\sum_{j=1}^{k} y_j a_j + y_{k+1} a'} g_1^{r id_0^* y_{id}}$, and stores (a', σ_1, σ_2), i.e., it is a signature on $(\mathrm{dlog}_{g_0}(g_1) id_0^*, A)$. Note that it has the same distribution as in the real scheme.

To answer an Auth query on (id_0^*, A, m), the reduction algorithm programs the random oracle to generate the proofs of knowledge.

To answer an Open query (m, A, tok), the reduction algorithm first verifies its validity. If the token is valid, it checks whether it was the answer to a prior Auth query on (id_0^*, A, m). If so, it returns id_0^*. If it was not the answer to such a query, under the SDL assumption, tok cannot have been forged for an identity id such that (id, A) is honest. Indeed, to use an adversary that would make such a forgery to win the SDL challenge, upon receiving an SDL tuple $(\Gamma, g_0, g_1, \tilde{g}_0, \tilde{g}_1)$ with $\mathrm{dlog}_{g_0} g_1 = \mathrm{dlog}_{\tilde{g}_0} \tilde{g}_1$ (instead of a challenge DDH tuple), the reduction algorithm chooses uniformly at random an identity id^* for which it sets the credential as before for id_0^*. After that, it proceeds exactly like before, and to test whether a token $tok = (\sigma_1, \sigma_2, \pi)$ should open to (id^*, A), it can verify the equality $e\left(\sigma_1, \tilde{g}_0^x \tilde{g}_1^{id^* y_{id} + \sum_j a_j y_j + y_{k+1} a'}\right) = e(\sigma_2, \tilde{g}_0)$. If it holds, it can run $\mathcal{A}$ anew and reprogram oracle $\mathcal{H}$ to extract $\mathrm{dlog}_{g_0} g_1$ with probability at least $\varepsilon^2/q_{\mathcal{H}} - 1/p$ ($q_{\mathcal{H}}$ the maximum number of $\mathcal{H}$ queries that $\mathcal{A}$ makes) by the forking lemma [5]. At the challenge phase, if $(id_0^*, A_0^*) = (id^*, A^*)$, then the reduction algorithm first generates $\nu \in \mathbb{Z}_p^*$, computes $g_2 \leftarrow g_0^\nu$, $g_3 \leftarrow g_1^\nu$, $\sigma_1 \leftarrow g_3$ and $\sigma_2 \leftarrow g_2^{x+\sum_{j=1}^{k} y_j a_j^* + y_{k+1} a'^*} g_3^{y_{id} id_0^*}$, and programs $\mathcal{H}$ to simulate a proof of knowledge of $\mathrm{dlog}_{g_0}(g_1) id^*$. After the challenge phase, it answers the queries as before. In the event in which $\mathcal{A}$ forges a token tok on m and A for an identity id such that (id, A) is honest, that identity is id^* with probability $1/|ID|$. The reduction algorithm can therefore win the SDL game with probability at least $1/|ID|(\varepsilon^2/q_{\mathcal{H}} - 1/p)$. Under the SDL assumption, tok can therefore not be a forgery for an identity id such that (id, A) is honest. The reduction algorithm (to the DDH game) can then runs the Open algorithm and returns its output.

The reduction algorithm answers the other queries as specified in the security experiment.

At the challenge phase, the reduction algorithm aborts if its guess is not correct, otherwise computes $\left(g_2, g_2^{x+\sum_{j=1}^{k} y_j a_j + y_{k+1} a'} g_3^{r id_0^* y_{id}}\right)$, simulates a proof of knowledge that involves m^* in the hash by programming the random oracle accordingly, and replies the couple and the proof as challenge token. If (g_0, g_1, g_2, g_3) is a DDH tuple, it is distributed as the response of $\mathcal{C}_0$, otherwise as that of Δ.

The reduction algorithm answer the other queries as before the challenge.

Recall that conditioned on the event in which the adversary wins the game,

1. $(id_0^*, A^*) \notin \mathcal{L}_{\mathsf{corrupt}}$ and $(id_1^*, A^*) \notin \mathcal{L}_{\mathsf{corrupt}}$, and
2. $(m^*, A^*, tok^*) \notin \mathcal{L}_{\mathsf{opened}}$.

Consequently, the joint distribution of the answers of the reduction algorithm to Issue, Issue.I, Auth, Corrupt, Open queries and the challenge queries are identically distributed to those of either $\mathcal{C}_0$ or Δ depending on whether (g_0, g_1, g_2, g_3) is a DDH tuple or not.

$\mathcal{C}_0$ and Δ are thus computationally indistinguishable under the DDH assumption. (The distinguishing advantage of any adversary is at most $|ID|^2$ times the DDH advantage of the prior reduction algorithm.)

Likewise, Δ and $\mathcal{C}_1$ are computationally indistinguishable. $\mathcal{C}_0$ and $\mathcal{C}_1$ are thereby computationally indistinguishable, and DGSA satisfies anonymity under the DDH assumption. Denote by DDH the DDH advantage of the reduction algorithm and SDL its SDL advantage, The anonymity advantage of $\mathcal{A}$ is at most

$$2\left(|ID|^2 DDH + \sqrt{q_{\mathcal{H}}(|ID|SDL + 1/p)}\right),$$

and the theorem follows.

Proof (of Theorem 3). The proof consists in reducing the traceability of the DGS+A scheme to the existential unforgeability of the modified PS k+1-message signature scheme. As its unforgeability relies on the q-MSDH-1 assumption, the theorem follows.

Let $\mathcal{A}$ be an efficient adversary that wins the traceability game with probability at least ε. Consider a simulator $\mathcal{S}$ which runs $\mathcal{A}$ as a subroutine and interacts with a forgery-game challenger $\mathcal{C}$ for the modified PS $k+1$-message multi-signature scheme. Upon receiving public parameters pp and of a verification vk from $\mathcal{C}$, simulator $\mathcal{S}$ sets $pk \leftarrow vk$ and forwards pp and pk to $\mathcal{A}$.

Simulator $\mathcal{S}$ answers $\mathcal{H}$ queries on new inputs by choosing a uniformly random $\mathbb{Z}_p$ element, and later replies with the same answer when queried on the same inputs. To answer Issue queries on (id, A), simulator $\mathcal{S}$ queries $\mathcal{C}$ on (id, A) and follows the rest of the protocol (, and stores the generated credential). Whenever $\mathcal{A}$ makes an Issue.I query on (id, A), simulator $\mathcal{S}$ queries $\mathcal{C}$ on (id, A) and obtains a PS signature $\sigma = (a', \sigma_1, \sigma_2)$. It then records and sends $cred \leftarrow (id, A, \sigma, e(\sigma_1, \tilde{Y}_{id}), e(\sigma_1, \tilde{Y}_{k+1}))$ to $\mathcal{A}$, and adds (id, A, a') to a list $L_{\mathcal{I}}$.

$\mathcal{S}$ answers Auth by calling on $\mathcal{C}$ to compute the challenge of the proofs and can complete the answer on its own.

To answer Corrupt a corrupt query on (id, A), simulator $\mathcal{S}$ simply returns the corresponding credential that it has recorded if it exists, otherwise returns $\perp$.

Ultimately, adversary $\mathcal{A}$ outputs an attribute set A^*, a message m^*, an authentication token $tok^* = (\sigma_1^*, \sigma_2^*, \pi^*)$. Conditioned on the event in which $\mathcal{A}$ wins the game, $\mathsf{DGSA.Vf}(pk, m^*, A^*, tok^*) = 1$ Simulator $\mathcal{S}$ runs $id^* \leftarrow \mathsf{Open}(\perp, L_{\mathcal{I}}, m^*, A^*, tok^*)$. (Note that algorithm Open only needs $L_{\mathcal{I}}$.) The winning condition ensures that 1) $id = \perp$ or 2.1) adversary $\mathcal{A}$ has never made a Issue or Issue.I query on (id^*, A^*) (due to the condition $(id^*, A^*) \notin \mathcal{L}_{\mathsf{joined}}$), so that

simulator $\mathcal{S}$ has never made a signing query to $\mathcal{C}$ on the message (id, A^*), or that 2.2) if the adversary has generated a credential for the pair (id^*, A^*) and never corrupted it, it has never obtained a token for the message m^* with the credential associated to (id^*, A^*) (due to the condition $(id^*, A^*) \in \mathcal{L}_{\mathsf{joined}} \wedge (id^*, A^*) \notin \mathcal{L}_{\mathsf{corrupt}} \wedge (id^*, A^*, m^*) \notin \mathcal{L}_{\mathsf{signed}}$).

Since $\mathsf{DGSA.Vf}(pk, m^*, A^*, tok^*) = 1$, the proof π^* is valid with respect to the random oracle $\mathcal{H}$ run by $\mathcal{S}$. Simulator $\mathcal{S}$ runs $\mathcal{A}$ anew on the same inputs and the same randomness. By the forking lemma [5], with probability at least $\varepsilon(\varepsilon/q - 1/p)$, during its second run, $\mathcal{A}$ outputs a second challenge tuple (A'^*, m'^*, tok'^*) and also queries $\mathcal{H}$ on $(u', A'^*, m'^*, \sigma_1'^*, \sigma_2'^*, pk)$ at the same computation step at which it queried $\mathcal{H}$ on $(u, A^*, m^*, \sigma_1^*, \sigma_2^*, pk)$ for

$$u := e\left(\sigma_1^{v_{id}^*}, \tilde{Y}_{id}\right) e\left(\sigma_1^{v_{a'}^*}, \tilde{Y}_{k+1}\right) e\left(\sigma_2^{*c}, \tilde{g}\right) e\left(\sigma_1^{*c}, \tilde{X}^{-1} \prod_{j=1}^{k} \tilde{Y}_j^{-a_j}\right),$$

and u' defined similarly. Moreover, the answers c^* and c'^* to these queries are distinct modulo p (as the output space of $\mathcal{H}$ is $\mathbb{Z}_p$). Simulator $\mathcal{S}$ can then extract a'^* such that

$$e\left(\sigma_1^*, \tilde{X}\tilde{Y}_{id}^{id} \prod_{i=1}^{k} \tilde{Y}_i^{a_i} \tilde{Y}_{k+1}^{a'^*}\right) = e(\sigma_2^*, \tilde{g}).$$

In case 1), simulator $\mathcal{S}$ sends (id^*, A^*) and $(a'^*, \sigma_1^*, \sigma_2^*)$ to $\mathcal{C}$ and wins the forgery game win probability at least ε. Case 2.1) cannot occur as the opening algorithm would only return id^* such that $(id^*, A^*, *)$ is in the state of the issuer and therefore also in $\mathcal{L}_{\mathsf{joined}}$. In case 2.2), simulator $\mathcal{S}$ returns $\perp$ to $\mathcal{C}$. Case 2.2) can only occurs if $\mathcal{A}$ has forged a token for an honest identity–attribute pair. As in the proof of anonymity, $\mathcal{A}$ can forge a token for an honest identity–attribute pair only if it can win the SDL game. It follows that $\mathcal{S}$ wins the forgery game with probability at least $\varepsilon(\varepsilon/q - 1/p) - |ID|SDL$. If ε were non-negligible, then $\mathcal{S}$ would be an algorithm that wins the forgery game of the modified PS $k+1$-message signature scheme with non-negligible probability, which is impossible under the q-MSDH-1 assumption.

C DGS+A with Threshold Opening

In definition and construction of DGS+A in Sections 3.1 and 3.2, the issuer can single-handedly open all tokens, making him a single point of failure for privacy. However, one can modify the scheme in Section 3.2 to distribute the authority to open tokens over a group of n authorities, so that at least a threshold $\tau + 1$ of them have to collaborate to open a token.

The main idea is to link the user's identity to an element $g^z \in \mathbb{G}$ which the issuer blindly signs as part of the credential. We then use the folklore encrypt-and-sign construction of group signatures [7, Section 5] so that every token contains an ElGamal ciphertext encrypting g^z under a public key, of which the decryption key is secret-shared among all opening authorities.

In more detail, with respect to the DGS+A scheme in Section 3.2, the public parameters contain additional elements $g, h_1 \in \mathbb{G}^*$. The n opening authorities perform a distributed key generation protocol [31] to generate a public key $h_0 \in \mathbb{G}^*$ so that there exists a polynomial $P(X)$ of degree τ such that $h_0 = g^{P(0)}$, and such that the i-th authority obtains secret key share $x_i = P(i) \bmod p$. The issuer generates his keys as before, except that y_{id} and $\tilde{Y}_{id}$ are respectively renamed y_z and $\tilde{Y}_z$.

In the issuance protocol, the user chooses a random $z \in \mathbb{Z}_p$ and sends $Z = g^z$ together with a proof of knowledge of z to the issuer. The issuer then generates a PS signature on $(z, a_1, \ldots, a_k)$ by choosing $a' \in \mathbb{Z}_p$ and $r \in \mathbb{Z}_p^*$, and computing a PS signature as $\left(a', g^r, g^{x+\sum_{j=1}^{k} y_j a_j + y_{k+1} a'} \cdot Z^{y_z}\right)$. The issuer sends (id, Z) to all opening authorities and sends the PS signature back to the user. The user stores the PS signature as well as her secret exponent z.

To authenticate a message m with a credential $cred = (z, A, \sigma,\ e(\sigma_1, \tilde{Y}_z),\ e(\sigma_1,\ \tilde{Y}_{k+1}))$, the user first generates $r \in \mathbb{Z}_p^*$ and compute $(\sigma_1', \sigma_2') \leftarrow (\sigma_1^r, \sigma_2^r)$. Next, the user computes two ElGamal ciphertexts C_0 and C_1 of g^z under keys h_0 and h_1, respectively. Namely, she chooses $r_0, r_1 \in \mathbb{Z}_p$ and sets $C_0 \leftarrow (g^{r_0}, g^z h_0^{r_0})$ and $C_1 \leftarrow (g^{r_1}, g^z h_1^{r_1})$. Finally, the user computes a Schnorr signature of knowledge on m of z, a', r_0 and r_1 such that $(\sigma_1', \sigma_2', a')$ is a PS signature of the issuer on (z, A), and such that r_0 and r_1 are the randomness used to respectively compute the first and second encryptions of g^z. A token is thus of the form $(\sigma_1', \sigma_2', C_0, C_1, \pi := (c, v_z, v_{a'}, v_{r_0}, v_{r_1}))$. Verification simply consists in verifying the signature of knowledge.

To open a token $(\sigma_1', \sigma_2', C_0, C_1, \pi)$ for a message m between $\tau+1$ openers in $I \in \binom{[n]}{\tau+1}$, each opener $i \in I$ first verifies the token. The ith opener broadcasts a decryption share $\mathscr{C}_i \leftarrow C_{0,0}^{x_{0,i}}$ of $C_0 =: (C_{0,0}, C_{0,1})$ to all the other openers $j \in I \setminus \{i\}$. Each opener can then decrypt C_0 by computing $C_{0,1} / \prod_i \mathscr{C}_i^{w_i}$, where w_i is the Lagrange interpolation coefficient of i in I, and check whether the resulting plaintext Z matches a pair (id, Z) that he received from the issuer. If so, the opener returns id, otherwise he returns $\perp$.

With the same parameters as in Section 3.2.1, tokens are now 552 Bytes long, or more than twice as long as the single-opener scheme of Section 3.2. This is acceptable for use in Zone Encryption, however, because the DGS+A scheme is only used when a vehicle enters a new zone, not for every CAM.

Alternatively, one could use the threshold group signatures of Camenisch et al. [14] which do not burden signatures which an additional encryption but keeps them as short as in the single-authority setting. Tokens are then still 216 Bytes long.

D Our Zone-Encryption Scheme

We here give the full security proofs of our zone-encryption scheme.

Proof (of Theorem 5). Under the assumptions of the theorem, the PH-CCA security of $\mathcal{Z}$ can be proved via the following hybrid argument. Let $\mathcal{A}$ be an adversary

for the PH-CCA security distinction experiment (i.e., $\mathcal{A}$ tries to tell apart the PH-CCA challenger $\mathcal{C}_0^{\mathrm{ph-cca}}$ that encrypts P_0 and the challenger $\mathcal{C}_1^{\mathrm{ph-cca}}$ that encrypts P_1). Suppose that $\mathcal{A}$ wins the game with $(\mathcal{V}^*, P_0, P_1, Y^*, t^*)$ as a challenge tuple. Number the zones in Y^* from 1 to $n_{Y^*} \coloneqq |Y^*|$, i.e., $Y^* = \{y_1, \ldots, y_{n_{Y^*}}\}$. For $i = 0, \ldots, n_{Y^*}$, consider the hybrid algorithm Δ_i that proceeds exactly like the PH-CCA challenger $\mathcal{C}_0^{\mathrm{ph-cca}}$, except that to compute the challenge ciphertext, it encrypts the first i zones with a payload key K' and the remaining $z-i$ zones with another key K, and encrypts P_0 with K. Namely, the challenge ciphertext of Δ_i is of the form

$$\begin{aligned}&\mathsf{DAE.Enc}(K_{y_1,t}, K'), \ldots, \mathsf{DAE.Enc}(K_{y_i,t}, K'),\\ &\mathsf{DAE.Enc}(K_{y_{i+1},t}, K), \ldots, \mathsf{DAE.Enc}(K_{n_{Y^*},t}, K),\\ &\mathsf{SE.Enc}(K, P_0).\end{aligned}$$

Consider also, for $i = 0, \ldots, n_{Y^*}$, the hybrid algorithm Δ_i' that proceeds exactly like the PH-CCA challenger $\mathcal{C}_1^{\mathrm{ph-cca}}$, except that to compute the challenge ciphertext, it encrypts the first i zones with a payload key K and the remaining $z-i$ zones with another key K', and encrypts P_1 with K. Namely, the challenge ciphertext of Δ_i' is of the form

$$\begin{aligned}&\mathsf{DAE.Enc}(K_{y_1,t}, K), \ldots, \mathsf{DAE.Enc}(K_{y_i,t}, K),\\ &\mathsf{DAE.Enc}(K_{y_{i+1},t}, K'), \ldots, \mathsf{DAE.Enc}(K_{n_{Y^*},t}, K'),\\ &\mathsf{SE.Enc}(K, P_1).\end{aligned}$$

By definition, $\Delta_0 = \mathcal{C}_0^{\mathrm{ph-cca}}$, the challenger that encrypts P_0 and $\Delta_{n_{Y^*}}' = \mathcal{C}_1^{\mathrm{ph-cca}}$, the challenger that encrypts P_1. To show that $\mathcal{A}$ has a negligible advantage in the PH-CCA distinction experiment, it suffices to show that the advantage of $\mathcal{A}$ in distinguishing two consecutive hybrids is negligible.

If adversary $\mathcal{A}$ can distinguish Δ_i from Δ_{i+1}, then it can be used to win the DAE privacy game for DAE as follows. Assume SIG to be existentially unforgeable, DGSA to satisfy traceability, SE to be IND-CPA secure, PKE to be IND-CPA secure, and DAE to satisfy authenticity. Let $\mathcal{S}$ be a simulator that features $\mathcal{A}$ and interacts with a DAE privacy game $\mathcal{C}_b^{\mathrm{priv}}$ for $b \in \{0,1\}$, which generates a secret key K_{priv}. At the beginning of the game, $\mathcal{S}$ receives parameters pp_{DAE} for DAE from $\mathcal{C}_b^{\mathrm{priv}}$ and generates the other parameters itself. It generates $(pk_{\mathcal{E}} = vk, sk_{\mathcal{E}} = sk) \leftarrow \mathsf{SIG.KG}(pp_{\mathsf{SIG}})$ and $(pk_{\mathcal{I}} = pk, sk_{\mathcal{I}} = sk) \leftarrow \mathsf{DGSA.KG}(pp_{\mathsf{DGSA}})$. It then sends all the parameters and $pk_{\mathcal{E}}$ and pk_I to $\mathcal{A}$.

$\mathcal{S}$ chooses a time period $\tilde{t}$ uniformly at random and implicitly sets $K_{y_{i+1},\tilde{t}} \coloneqq K_{\mathrm{priv}}$ (i.e., it will query $\mathcal{C}_b^{\mathrm{priv}}$ to answer queries Send queries involving zone y_{i+1} and time $\tilde{t}$).

Throughout the game, $\mathcal{S}$ locally maintains the same lists as the PH-CCA challenger does.

For $\mathsf{Enroll.V\&Enroll.E}$ queries, $\mathcal{S}$ runs the protocol and stores the generated certificates.

For $\mathsf{Enroll.E}$ queries, simulator runs the corresponding algorithms with $sk_{\mathcal{E}}$.

For Authorize.V&I queries, $\mathcal{S}$ runs the protocol and stores the generated credentials.

For Authorize.I queries, $\mathcal{S}$ runs the corresponding algorithm with pk_I.

For an Enter query on input $(\mathcal{V}, z, t, \mathit{role})$, ($\mathcal{V} \notin \mathcal{L}_{\mathsf{corrupt}}$ by definition of this oracle) for any *role*, if $(z,t) = (y_{i+1}, \tilde{t})$ then $\mathcal{S}$ starts an execution of protocol Enter with $\mathcal{A}$.

If $\mathit{role} = \mathit{requester}$, then simulator $\mathcal{S}$ generates $(ek, dk) \leftarrow \mathsf{PKE.KG}(1^\lambda)$, computes a token $tok \leftarrow \mathsf{DGSA.Auth}(pk_{\mathcal{I}}, \mathit{cred}_{\mathcal{V},e(t)}, (z, t, ek))$ and sends $(z, t, ek, tok_{\mathcal{V}})$ to $\mathcal{A}$.

If $\mathit{role} = \mathit{responder}_i$, and that $\mathcal{V}$ should respond according to the strategy of $\mathit{responder}_i$, then $\mathcal{S}$, upon receiving $(y_{i+1}, \tilde{t}, ek, tok)$ from $\mathcal{A}$, determines whether it comes from a non-corrupt vehicle identity in the same protocol execution, i.e., whether $\mathcal{A}$ is simply performing a passive attack by relaying a message from a non-corrupt vehicle identity.

If so, then $\mathcal{S}$ encrypts a random message instead of K with PKE. The IND-CPA security of PKE is important here to argue for indistinguishability between the two consecutive hybrids. If $\mathcal{V} \in \mathcal{L}_{\mathsf{corrupt}}$, simulator $\mathcal{S}$ returns $\perp$.

If $(y_{i+1}, \tilde{t}, ek, tok)$ does not come from a non-corrupt vehicle in the same protocol execution, it is an active attack. $\mathcal{S}$ then aborts the protocol execution. In the event in which $\mathcal{A}$ wins the PH-CCA game, if $t^* = \tilde{t}$ ($\mathcal{S}$ will abort if it is not the case), the winning condition 3b) implies that no vehicle $\mathcal{V}_j \in \mathcal{L}_{\mathsf{corrupt}}$ can be authorized in $e(\tilde{t})$, so that

- either there exists a vehicle identity $\mathcal{W}$ such that $(\mathcal{W}, e(\tilde{t})) \in \mathcal{L}_{\mathsf{auth}}$ but $\mathcal{W} \notin \mathcal{L}_{\mathsf{honest}}$ (and also not in $\mathcal{L}_{\mathsf{corrupt}}$), i.e., it has obtain a credential for $e(\tilde{t})$ but has never been enrolled neither as an honest vehicle nor a corrupt one; and it happens with negligible probability if SIG is EUF-CMA secure
- or no such vehicle exists and the token sent the adversary can be valid w.r.t. to $pk_{\mathcal{I}}$ and $e(\tilde{t})$ with only negligible probability if DGSA satisfies traceability.

Therefore, by aborting the protocol once a token is received from $\mathcal{A}$ during the Enter protocol execution, $\mathcal{S}$ is computationally indistinguishable from both Δ_i and Δ_{i+1}.

If $(z,t) \neq (y_{i+1}, \tilde{t})$, then $\mathcal{S}$ simply runs the Enter.V algorithm on input $(\mathcal{V}, z, t, \mathit{role})$, and never has to send K_{priv}.

For an Exit on input $(\mathcal{V}, z, t)$, simulator $\mathcal{S}$ simply deletes $(z, t, K_{z,t})$ from $\mathcal{V}[L_K]$ that it locally maintains for $\mathcal{V}$.

For a Send query on $(\mathcal{V}, P, Y \ni y_{i+1}, \tilde{t})$, simulator $\mathcal{S}$ checks whether all the zones in Y are active for $\mathcal{V}$, and in particular if $(y_{i+1}, \tilde{t}, \cdot) \in \mathcal{V}[L_K]$. If not, it returns $\perp$, otherwise $\mathcal{S}$ generates a payload key $K \leftarrow \mathsf{DAE.KG}(pp_{\mathsf{DAE}})$, computes $ct \leftarrow \mathsf{DAE.Enc}(K, P)$, and computes $\gamma_{y_{i+1},\tilde{t}}$ for the zone–time pair $(y_{i+1}, \tilde{t})$ by sending K to $\mathcal{C}_b^{\mathsf{priv}}$. Simulator $\mathcal{S}$ then encrypts K with the keys for the other zone–time pairs in the query, and sets the ciphertext as the Enter algorithm does. For Send queries such that $y_{i+1} \notin Y$ or $t \neq \tilde{t}$, simulator $\mathcal{S}$ runs algorithm Send on the inputs.

For a Receive query on a vehicle identity $\mathcal{V}$ and a ciphertext $\gamma = (\tilde{t}, Y, ((y, \gamma_{y,\tilde{t}})_{y \in Y}, ct))$ such that $y_{i+1} \in Y$, algorithm $\mathcal{S}$ first determines whether y_{i+1} is

the only zone in Y that is active for $\mathcal{V}$ in time $\tilde{t}$. If not, then $\mathcal{S}$ can answer the query using any other zone z that is active for $\mathcal{V}$, i.e., by computing $K_P \leftarrow \mathsf{DAE.Dec}(K_{z,\tilde{t}}, ct, \gamma_{z,\tilde{t}})$, and returning $P \leftarrow \mathsf{SE.Dec}(K_P, ct)$. If y_{i+1} is the only zone in γ that is active for $\mathcal{V}$ in time t, then $\mathcal{S}$ checks whether $\gamma_{y_{i+1},\tilde{t}}$ and ct are the output of a same previous Send query that it answered. If not, then $\mathcal{S}$ replies with $\perp$. This is where the authenticity of DAE comes into play. Indeed, if DAE satisfies authenticity, $\mathcal{A}$ can submit a ciphertext that does not decrypt to $\perp$ with only negligible probability. If $\gamma_{y_{i+1},\tilde{t}}$ and ct are part of the answer to a previous Send query, then $\mathcal{S}$ replies with the payload on which it was queried. For Receive oracle involving other zone–time pairs, $\mathcal{S}$ simply runs algorithm Receive on the queried input.

For an Open query on input m, simulator $\mathcal{S}$ parses m as (m', t, tok) and runs $\mathsf{DGSA.Open}(sk_{\mathcal{I}}, st_{\mathcal{I}}, (m, e(t), tok))$. It returns the output to $\mathcal{A}$.

For a Corrupt query on an identity $\mathcal{V}$, simulator $\mathcal{S}$ replies by sending to $\mathcal{A}$ the certificate $\mathcal{V}[cert_{\mathcal{V}}]$, all the credentials $\mathcal{V}[e_j, cred_{V,j}]$ and the key list $\mathcal{V}[L_K]$. Note that in the event in which $\mathcal{A}$ wins the game, if $t = \tilde{t}$, no challenge zone $y^* \in Y^*$ can be active for $\mathcal{V}$ in time t^* (condition 2). In particular, $\mathcal{S}$ never has to send $K_{y_{i+1},\tilde{t}}$ to $\mathcal{A}$.

At the challenge phase, adversary $\mathcal{A}$ outputs the challenge tuple $(\mathcal{V}^*, P_0, P_1, Y^*, t^*)$. If $\mathcal{V}^* \in \mathcal{L}_{\mathsf{corrupt}}$, then $\mathcal{S}$ returns 0 as the PH-CCA challenger does. If $\tilde{t} \neq t^*$, then $\mathcal{S}$ aborts and outputs $\perp$; otherwise, $\mathcal{S}$ generates two payload keys K and K', sends them to $\mathcal{C}$, and receives a ciphertext $\gamma^*_{y_{i+1},\tilde{t}}$ from $\mathcal{C}_b^{\text{priv}}$. For $k = 1, \ldots, i$, Simulator $\mathcal{S}$ computes $\gamma_{y_k,\tilde{t}} \leftarrow \mathsf{DAE.Enc}(K_{y_k,\tilde{t}}, K')$, and for $i < k \leq n_{Y^*}$, it computes $\gamma_{y_k,\tilde{t}} \leftarrow \mathsf{DAE.Enc}(K_{y_k,\tilde{t}}, K)$. It also compute $ct \leftarrow \mathsf{DAE}(K, P_0)$, and then sends the ciphertext $\gamma^* \leftarrow \big(\tilde{t}, Y^*, ((y_k, \gamma_{y_k,\tilde{t}})_{k\leq i}, (y_{i+1}, \gamma^*_{y_{i+1},\tilde{t}}), (y_k, \gamma_{y_k,\tilde{t}})_{k>i+1}, ct)\big)$ to adversary $\mathcal{A}$.

After the challenge phase, for Receive on $(\mathcal{V}, \gamma)$ queries from $\mathcal{A}$, simulator $\mathcal{S}$ first parses γ as (t, Y, γ'). Conditioned on the event in which $\mathcal{A}$ wins the PH-CCA game, no part of γ' is replayed from the challenge ciphertext γ^*, i.e., $\gamma' \cap \gamma^{*'} \neq \emptyset$. $\mathcal{S}$ can then proceeds as before the challenge phase.

For the other queries, $\mathcal{S}$ replies as before the challenge phase.

At the end of the experiment, $\mathcal{S}$ forwards the decision bit of $\mathcal{A}$. Up to the existential unforgeability of SIG and the traceability of DGSA, $\mathcal{S}$ perfectly simulates $\mathcal{C}_b^{\text{ph-cca}}$ to adversary $\mathcal{A}$ except for Receive queries involving $(y_{i+1}, \tilde{t})$ as the only zone in the ciphertext that is active for the queried vehicle and for passive Enter queries. Consequently, as $\tilde{t} = t^*$ with probability $1/|T|$,

$$\begin{aligned}\mathbf{Adv}_{\Delta_i,\Delta_{i+1}}(\mathcal{A}) - q_{\mathsf{Enter.act}}(y_{i+1}, \tilde{t})\mathbf{Adv}^{\text{euf-cma}}_{\mathsf{SIG}}(\mathcal{S}(\mathcal{A})) \\ - q_{\mathsf{Enter.act}}(y_{i+1}, \tilde{t})\mathbf{Adv}^{\text{trace}}_{\mathsf{DGSA}}(\mathcal{S}(\mathcal{A})) \\ - q_{\mathsf{Receive}}(y_{i+1}, \tilde{t})\mathbf{Adv}^{\text{auth}}_{\mathsf{DAE}}(\mathcal{S}(\mathcal{A})) \\ - q_{\mathsf{Enter.pass}}(y_{i+1}, \tilde{t})\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE}}(\mathcal{S}(\mathcal{A})) \\ \leq |T|\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{SE}}(\mathcal{S}(\mathcal{A}))\end{aligned}$$

with

– $q_{\mathsf{Enter.act}}(y_{i+1}, \tilde{t})$ the number of active enter queries in $(y_{i+1}, \tilde{t})$
– $q_{\mathsf{Receive}}(y_{i+1}, \tilde{t})$ the number of Receive queries involving $(y_{i+1}, \tilde{t})$ as the only zone in the ciphertext that is active for the queried vehicle
– $q_{\mathsf{Enter.pass}}(y_{i+1}, \tilde{t})$ the numer of passive Enter queries made by for zone y_{i+1} in time $\tilde{t}$.

Therefore, if SIG is EUF-CMA secure, if DGSA satisfies traceability, if PKE is IND-CPA secure and if DAE satisfies authenticity, then Δ_i and Δ_{i+1} are computationally indistinguishable.

The IND-CPA security of SE can be reduced to the computational indistinguishability of Δ'_i and Δ'_{i+1} in the very same manner.

Note also that the IND-CPA security of SE can be reduced to the computational indistinguishability of $\Delta_{n_{Y^*}}$ and Δ'_0. The reduction algorithm can implicitly set K as the challenger key, and forward the challenge tuple (P_0, P_1) at the challenge phase. It can answer of all the other Send queries (i.e., other that the challenge one) by generating fresh payload keys. Moreover, as all the key for active zones are known to $\mathcal{S}$, it can answer all the other queries.

Overall,

$$\mathbf{Adv}^{\mathrm{ph-cca}}_{\mathcal{Z}}(\mathcal{A}) - \mathrm{negl}\,(\lambda) \leq 2n_{Y^*}|T|\mathbf{Adv}^{\mathrm{priv}}_{\mathsf{DAE}}(\mathcal{S}(\mathcal{A})) + \mathbf{Adv}^{\mathrm{ind-cpa}}_{\mathsf{SE}}(\mathcal{S}(\mathcal{A})),$$

hence the statement of the theorem.

Proof (of Theorem 6). Assuming that DGSA satisfies anonymity, and that SIG is EUF-CMA secure, the anonymity of $\mathcal{Z}$ can be proved via the following hybrid argument. Let $\mathcal{A}$ be an adversary for the ZE anonymity game that makes q Enter* queries. One can assume that $q > 0$. Indeed, an adversary that wins the game with $q = 0$ can always be run as a sub-routine by an adversary that makes one arbitrary Enter* query. For $i = 0, \ldots, q$, let Δ_i be an algorithm that proceeds exactly that the ZE anonymity game challenger, except that to answer the (*) queries with a bit d up to the ith Enter*, it uses $\mathcal{V}_{1-d}$. For the remaining (*) queries (including the remaining $q - i$ Enter*), it uses $\mathcal{V}_d$. By definition, if $\mathcal{C}_b$ denotes the ZE anonymity challenger that uses $\mathcal{V}_b$, then $\Delta_0 = \mathcal{C}_0$ and $\Delta_q = \mathcal{C}_1$. The advantage of $\mathcal{A}$ in the ZE anonymity game is therefore at most q times its advantage in distinguishing Δ_i from Δ_{i+1} for some $0 \leq i \leq q-1$. However, if $\mathcal{A}$ can distinguish Δ_i from Δ_{i+1}, then it can be used to win the DGS+A anonymity game as follows.

Consider a simulator that runs $\mathcal{A}$ as a subroutine and interacts with the DGS+A anonymity challenger $\mathcal{C}_{\mathsf{DGSA},b}$ for $b \in \{0, 1\}$. Throughout the game, $\mathcal{S}$ locally maintains the same lists as the PH-CCA challenger does.

Upon receiving parameters pp_{DGSA} for DGSA and a public pk_I, simulator $\mathcal{S}$ generates the parameters for the other schemes itself, generates $(pk_{\mathcal{E}} = vk, sk_{\mathcal{E}} = sk) \leftarrow \mathsf{SIG.KG}(pp_{\mathsf{SIG}})$ and sends all the parameters to $\mathcal{A}$ as well as $pk_{\mathcal{E}}$ and $pk_{\mathcal{I}}$.

For Enroll.V&Enroll.E queries, $\mathcal{S}$ runs the protocol and stores the generated certificates.

For Enroll.E queries, simulator runs the corresponding algorithms with $sk_{\mathcal{E}}$.

For an Authorize.V&I on $(\mathcal{V}, e)$, simulator $\mathcal{S}$ queries the Auth oracle of $\mathcal{C}_{\mathsf{DGSA},b}$ for $(\mathcal{V}, e)$ and stores the output credential.

For an Authorize.I query on $(\mathcal{V}, e)$, simulator $\mathcal{S}$ starts an execution of protocol Authorize with $\mathcal{A}$. Conditioned on the event in which $\mathcal{A}$ wins the game with a pair of identities $\mathcal{V}_0$ and V_1, if $\mathcal{V} = \mathcal{V}_d$ for $d \in \{0, 1\}$, upon receiving $(vk_{\mathcal{V}}, \sigma_{\mathcal{E}}, \sigma_{\mathcal{V}})$ from $\mathcal{A}$, simulator $\mathcal{S}$ checks whether $\mathcal{V}_d \in \mathcal{L}_{\mathsf{honest}}$ (i.e., it $\mathcal{V}_d$ is enrolled). If not, then $\mathcal{S}$ aborts and is indistinguishable from the ZE anonymity challenger under the assumption that SIG is EUF-CMA secure. If $\mathcal{V} \neq \mathcal{V}_0, \mathcal{V}_1$, then simulator $\mathcal{S}$ first checks that $\mathsf{SIG.Vf}(pk_{\mathcal{E}}, (\mathcal{V}, vk_{\mathcal{V}}), \sigma_{\mathcal{E}}) = 1$ and that . If not, $\mathcal{S}$ aborts; otherwise it starts an execution of protocol DGSA.Issue with $\mathcal{A}$ on $(\mathcal{V}, e)$ and simply forwarding every message from $\mathcal{A}$ to the Issue.I oracle provided by $\mathcal{C}_{\mathsf{DGSA},b}$ and vice versa.

$\mathcal{S}$ answers Enter queries by calling on oracle Auth to generate and verify authentication tokens.

For an Exit on input $(\mathcal{V}, z, t)$, simulator $\mathcal{S}$ simply deletes $(z, t, K_{z,t})$ from $\mathcal{V}[L_K]$ that it locally maintains for $\mathcal{V}$.

For Send and Receive queries, $\mathcal{S}$ runs the corresponding algorithms on the inputs.

For an Open query on input m, simulator $\mathcal{S}$ queries the Open oracle provided by the DGS+A-anonymity challenger on m .

To answer Corrupt queries on a vehicle identity $\mathcal{V}$, simulator $\mathcal{S}$ sends to the Corrupt oracle provided by $\mathcal{C}$ all the pairs $(\mathcal{V}, e_j)$ such that $(\mathcal{V}, e_j) \in \mathcal{L}_{\mathsf{auth}}$, gets a credential $cred_j$ for each e_j, and sets $cred_{\mathcal{V},j} \leftarrow cred_j$. It can then answer the query by returning the certificate $\mathcal{V}[cert_{\mathcal{V}}]$ (that it maintains locally), $\mathcal{V}[(e_j, cred_{\mathcal{V},j}]$ and the list $\mathcal{V}[L_K]$ of keys that it maintains for $\mathcal{V}$.

At the challenge phase, after the adversary outputs two challenge vehicle identities $\mathcal{V}_0$ and $\mathcal{V}_1$. Simulator $\mathcal{S}$ checks that they are authorized in exactly the same epochs and that $\mathcal{V}_0[L_K] = \mathcal{V}_1[L_K]$. If it is not the case, $\mathcal{S}$ aborts.

$\mathcal{S}$ simulator randomly chooses a zone–time pair $(\tilde{z}, \tilde{t})$ such that both vehicles are authorized in $e(\tilde{t})$.

After the challenge phase, the oracles Enter, Exit, Send and Receive are respectively replaced with the Enter*, Exit*, Send* and Receive* oracles.

For all oracles queries except the queries to these oracles with a bit d, simulator $\mathcal{S}$ replies as before the challenge phase (and recall that conditioned on the event in which $\mathcal{A}$ wins the game, neither $\mathcal{V}_0$ nor V_1 can be corrupt).

For the (*) queries up to the ith Enter* query,

1. if Enter* is queried on $(d, z, t, requester)$, $\mathcal{S}$
- checks that $(e(t), \mathcal{V}_{1-d}) \in \mathcal{L}_{\mathsf{auth}}$ (aborts if not)
- checks if $\exists (z, t, K_{z,t}) \in \mathcal{V}_{1-d}[L_K]$ (does nothing if it is the case)
- makes an authenticated key request: it generates a $(ek, dk) \leftarrow \mathsf{PKE.KG}(1^\lambda)$, queries the Auth oracle of $\mathcal{C}_{\mathsf{DGSA},b}$ on the tuple $(\mathcal{V}_{1-d}, e(t), (z, t, ek))$, receives an authentication token tok and sends (z, t, ek, tok) to $\mathcal{A}$
- upon receiving (z, t, ct, tok') from $\mathcal{A}$, checks that $\mathsf{DGSA.Vf}(pk_{\mathcal{I}},$ $(z, t, ct), e(t), tok') = 1$. If not, $\mathcal{S}$ aborts, otherwise it decrypts $K_{z,t} \leftarrow \mathsf{PKE.Enc}(ek, ct)$ and adds $(z, t, K_{z,t})$ to $\mathcal{V}_{1-d}[L_K]$

2. if Enter^* is queried on $(d, z, t, \mathit{responder}_i)$, $\mathcal{S}$ upon receiving (z, t, ek, tok) from $\mathcal{A}$, if $\exists(z, t, K_{z,t}) \in \mathcal{V}_{1-d}[L_K]$ and if it should reply according to the strategy of $\mathit{responder}_i$,
– checks that $\mathsf{DGSA.Vf}(pk_{\mathcal{I}}, (z, t, ek), e(t), tok) = 1$ (aborts if not)
– computes $ct \leftarrow \mathsf{PKE.Enc}(ek, K_{z,t})$
– queries the Auth oracle of $\mathcal{C}_{\mathsf{DGSA},b}$ on $(\mathcal{V}_{1-d}, e(t), (z, t, ct))$ and receives a token tok'
– sends (z, t, ct, tok') to $\mathcal{A}$
3. if Exit^* is queried on (d, z, t), simulator $\mathcal{S}$ deletes $(z, t, K_{z,t})$ from both $\mathcal{V}_{1-d}[L_K]$
4. if Send^* is queried on (d, P, Y, t), simulator $\mathcal{S}$
– computes $\gamma \leftarrow \mathcal{Z}.\mathsf{Enc}(V_{1-d}[L_K], P, Y, t)$ and sends it to $\mathcal{A}$

For the $i+1$th Enter^* query on input $(d, z, t, role)$, if $(z, t) \neq (\tilde{z}, \tilde{t})$, simulator $\mathcal{S}$ aborts, otherwise if it is an Enter^* with $role = requester$, to compute an authenticated key request, it sends $(\mathcal{V}_0, \mathcal{V}_1, e(\tilde{t}), (z, t, ek))$ as a challenge tuple to $\mathcal{C}_{\mathsf{DGSA},b}$. If it is an Enter^* query and that $role = \mathit{responder}_i$ and that it should reply according to the strategy to the strategy of $\mathit{responder}_i$, to compute its authenticated key response, simulator $\mathcal{S}$ sends $(\mathcal{V}_0, \mathcal{V}_1, e(\tilde{t}), (z, t, ct))$ as a challenge tuple to $\mathcal{C}_{\mathsf{DGSA},b}$.

For the remaining (*) queries $\mathcal{S}$ uses the state of $\mathcal{V}_d$ instead of $\mathcal{V}_{1-d}$.

Note that the winning condition enforces that neither $\mathcal{V}_0$ nor $\mathcal{V}_1$ can be corrupt throughout the game so $\mathcal{S}$ never has to return their states.

Moreover, the winning condition also implies that $\mathcal{A}$ never made an Open query on any message exchanged during the executions protocol Enter^*. As a consequence, the distribution of the answers of $\mathcal{S}$ to oracle queries except for the $i+1$th Enter^* query are identically to those of Δ_i and Δ_{i+1}.

At the end of the game, $\mathcal{S}$ forwards the decision bit b' of $\mathcal{A}$ to $\mathcal{C}_{\mathsf{DGSA},b}$. If $\mathcal{A}$ has made no Enter^* query on z and t, then $\mathcal{S}$ returns $\perp$ to $\mathcal{C}$, otherwise $\mathcal{S}$ forwards b' to $\mathcal{C}_{\mathcal{Z},b}$, and

$$\mathbf{Adv}_{\Delta_i,\Delta_{i+1}}(\mathcal{A}) - \mathrm{negl}(\lambda) \leq |Z||T|\mathbf{Adv}^{\mathrm{ano}}_{\mathsf{DGSA}}(\mathcal{S}(\mathcal{A}))$$

with the negligible factor coming from the unforgeability of SIG.

Therefore,

$$\mathbf{Adv}^{\mathrm{ano}}_{\mathcal{Z}}(\mathcal{A}) - \mathrm{negl}(\lambda) \leq q|Z||T|\mathbf{Adv}^{\mathrm{ano}}_{\mathsf{DGSA}}(\mathcal{S}(\mathcal{A})).$$

As $1/q|Z||T|$ is non-negligible, if $\mathcal{A}$ has a non-negligible advantage in the ZE anonymity game, then so does $\mathcal{S}$ in the DGS+A anonymity game; hence the theorem.

Proof (of Theorem 7). Consider an adversary $\mathcal{A}$ that wins the ZE traceabilty game with a non-negligible probability. To win the game, at the challenge phase, $\mathcal{A}$ outputs a tuple (z^*, t^*, K_{z^*,t^*}) such that there exists an honest vehicle identity $\mathcal{V}$ such that $(z^*, t^*, K_{z^*,t^*}) \in \mathcal{V}[L_K]$.

The winning condition implies that $K_{z^*,t^*} \notin \mathcal{L}_{\mathsf{keys}}$, so for K_{z^*,t^*} to be active for $\mathcal{V}$, either there exists at least one message m_j (with $m_j = (z^*, t^*, ct, tok)$ or

(z^*, t^*, ek, tok)) such that $(\mathcal{A}, z^*, t^*, m_j) \in \mathcal{L}_{\mathsf{enter}}$ or $(\mathcal{W}, z^*, t^*, m_j) \in \mathcal{L}_{\mathsf{enter}}$ for $\mathcal{W} \in \mathcal{L}_{\mathsf{corrupt}}$, or there does not exist such a message.

If there is no such message (case 0), then the traceability of $\mathcal{Z}$ can be reduced to the IND-CPA security of PKE since $\mathcal{A}$ only sees transcripts of Enter protocol executions.

If there exists at least one such message m_j, then the winning condition ensures that for $\mathcal{V}_j \leftarrow \mathsf{Open}(sk_{\mathcal{I}}, st_{\mathcal{I}}, m_j)$,

1. $\mathcal{V}_j \notin \mathcal{L}_{\mathsf{corrupt}}$ or
2. $(\mathcal{V}_j \in \mathcal{L}_{\mathsf{corrupt}}) \ \wedge \ (\mathcal{V}_j, e(t^*)) \notin \mathcal{L}_{\mathsf{auth}})$.

In case 1), the traceability of $\mathcal{Z}$ can be reduced to the unforgeability of SIG or the IND-CPA security of PKE. Indeed, the fact that $K_{z^*,t^*} \in \mathcal{V}[L_K]$ means that $\mathcal{A}$ has sent an authentication token tok for a message m_j that was accepted by an honest vehicle in (z^*, t^*), be it $\mathcal{V}$, (since $K_{z^*,t^*} \in \mathcal{V}[L_K]$) during an Enter protocol execution, i.e., $\mathsf{DGSA.Vf}(pk_{\mathcal{I}}, m_j, e(t^*), tok) = 1$. However, as $\mathcal{V}_j \notin \mathcal{L}_{\mathsf{corrupt}}$, if

1.1) $\mathcal{V}_j \in \mathcal{L}_{\mathsf{honest}}$, then $\mathcal{A}$ either 1.1.1) simply relayed a message between $\mathcal{V}_j$ and that honest vehicle, and the traceability of $\mathcal{Z}$ can be reduced to the IND-CPA security of PKE, or 1.1.2) the adversary forged a token that opens to an honest vehicle that never computed it, in which case the traceability of $\mathcal{Z}$ can be reduced to the traceability of DGSA

1.2) $\mathcal{V}_j \notin \mathcal{L}_{\mathsf{honest}}$ (and also not in $\mathcal{L}_{\mathsf{corrupt}}$), then the traceability of $\mathcal{Z}$ can be reduced to the unforgeability of SIG if 1.2.1) $(\mathcal{V}_j, e(t^*)) \in \mathcal{L}_{\mathsf{auth}}$ ($\mathcal{V}_j$ was never enrolled, i.e., $\mathcal{A}$ forged a certificate for it) or the traceability of DGSA if 1.2.2) $(\mathcal{V}_j, e(t^*)) \notin \mathcal{L}_{\mathsf{auth}}$.

In case 2), the traceability of $\mathcal{Z}$ can be reduced to the traceability of DGSA as $\mathcal{V}_j$ is corrupt, but was not authorized in $e(t^*)$.

The reduction to the IND-CPA security of PKE in case 0) and 1.1.1), is done by encrypting random a random key instead of the challenger in passive Enter queries.

To reduce to the traceability of DGSA in cases 1.1.2), 1.2.2) and 2), a simulator $\mathcal{S}$ running $\mathcal{A}$ as a subroutine uses message m_j and token tok sent by $\mathcal{A}$ as a forgery for the vehicle identity $\mathcal{V}_j$ in epoch $e(t^*)$.

To reduce to the unforgeability of SIG in case 1.2.1), $\mathcal{S}$ uses the signature of the certificate of $\mathcal{V}_j$ as a forgery.

Proof (of Theorem 8). Assume SIG to be EUF-CMA secure, DGSA to satisfy traceability and PKE to be IND-CPA secure. The ciphertext integrity of $\mathcal{Z}$ can be reduced to the authenticity of DAE as follows. Let $\mathcal{A}$ be an adversary that wins the authenticity game of $\mathcal{Z}$ with probability at least ε. Let $\mathcal{S}$ be a simulator which runs $\mathcal{A}$ as a subroutine and interacts with the challenger $\mathcal{C}$ of the authenticity game of DAE (which generates a secret key K). At the beginning of the game, $\mathcal{S}$ receives parameters pp_{DAE} for DAE from $\mathcal{C}_b^{\mathrm{priv}}$ and generates the other parameters itself. It generates $(pk_{\mathcal{E}} = vk, sk_{\mathcal{E}} = sk) \leftarrow \mathsf{SIG.KG}(pp_{\mathsf{SIG}})$ and $(pk_{\mathcal{I}} = pk, sk_{\mathcal{I}} = sk) \leftarrow \mathsf{DGSA.KG}(pp_{\mathsf{DGSA}})$. It then sends all the parameters and $pk_{\mathcal{E}}$ and pk_I to $\mathcal{A}$.

Simulator $\mathcal{S}$ chooses a zone–time pair $(\tilde{z}, \tilde{t})$ uniformly at random and implicitly sets $K_{\tilde{z},\tilde{t}} \coloneqq K$ (i.e., it will query $\mathcal{C}$ to answer queries involving $\tilde{z}$ and $\tilde{t}$).

Throughout the game, $\mathcal{S}$ locally maintains the same lists as the PH-CCA challenger does.

For Enroll.V&Enroll.E queries, $\mathcal{S}$ runs the protocol and stores the generated certificates.

For Enroll.E queries, simulator runs the corresponding algorithms with $sk_{\mathcal{E}}$.

For Authorize.V&I queries, $\mathcal{S}$ runs the protocol and stores the generated credentials.

For Authorize.I queries, $\mathcal{S}$ runs the corresponding algorithm with pk_I.

For an Enter query on input $(\mathcal{V}, z, t, role)$, ($\mathcal{V} \notin \mathcal{L}_{\mathsf{corrupt}}$ by definition of this oracle) for any $role$, if $(z,t) = (\tilde{z}, \tilde{t})$ then $\mathcal{S}$ starts an execution of protocol Enter with $\mathcal{A}$.

If $role = requester$, then $\mathcal{S}$ generates $(ek, dk) \leftarrow \mathsf{PKE.KG}(1^\lambda)$, computes $tok \leftarrow \mathsf{DGSA.Auth}(pk_{\mathcal{I}}, cred_{\mathcal{V},e(t)}, (z, t, ek))$ and sends $(z, t, ek, tok_{\mathcal{V}})$ to $\mathcal{A}$.

If $role = responder_i$, and that $\mathcal{V}$ should respond according to the strategy of $responder_i$, then $\mathcal{S}$, upon receiving $(\tilde{z}, \tilde{t}, ek, tok)$ from $\mathcal{A}$, determines whether it comes from a non-corrupt vehicle identity in the same protocol execution, i.e., whether $\mathcal{A}$ is simply performing a passive attack by relaying a message from a non-corrupt vehicle identity.

Simulator $\mathcal{S}$ then encrypts a random message instead of K with PKE. The IND-CPA security of PKE is important here to argue for indistinguishability between the two consecutive hybrids. If $\mathcal{V} \in \mathcal{L}_{\mathsf{corrupt}}$, simulator $\mathcal{S}$ returns $\perp$.

If $(\tilde{z}, \tilde{t}, ek, tok)$ does not come from a non-corrupt vehicle in the same protocol execution, it is an active attack. $\mathcal{S}$ then aborts the protocol execution. Conditioned on the event in which $\mathcal{A}$ wins the PH-CCA game, if $t^* = \tilde{t}$ ($\mathcal{S}$ will abort if it is not the case), the winning condition 3b) implies that no vehicle $\mathcal{V}_j \in \mathcal{L}_{\mathsf{corrupt}}$ can be authorized in $e(\tilde{t})$, so that

- either there exists a vehicle identity $\mathcal{W}$ such that $(\mathcal{W}, e(\tilde{t})) \in \mathcal{L}_{\mathsf{auth}}$ but $\mathcal{W} \notin \mathcal{L}_{\mathsf{honest}}$ (and also not in $\mathcal{L}_{\mathsf{corrupt}}$), i.e., it has obtain a credential for $e(\tilde{t})$ but has never been enrolled whether as an honest vehicle or not; and it happens with negligible probability if SIG is EUF-CMA secure
- or no such vehicle exists and the token sent the adversary can be valid w.r.t. to $pk_{\mathcal{I}}$ and $e(\tilde{t})$ with only negligible probability if DGSA satisfies traceability.

Therefore, by aborting the protocol once a token is received from $\mathcal{A}$ during the Enter protocol execution, $\mathcal{S}$ is computationally indistinguishable from both Δ_i and Δ_{i+1}.

If $(z,t) \neq (\tilde{z}, \tilde{t})$, then $\mathcal{S}$ simply runs the Enter.V algorithm on input $(\mathcal{V}, z, t, role)$, and never has to send K_{priv}.

For an Exit on input $(\mathcal{V}, z, t)$, simulator $\mathcal{S}$ simply deletes $(z, t, K_{z,t})$ from $\mathcal{V}[L_K]$ that it locally maintains for $\mathcal{V}$.

For a Send query on $(\mathcal{V}, P, Y \ni \tilde{z}, \tilde{t})$, simulator $\mathcal{S}$ checks whether all the zones in Y are active for $\mathcal{V}$, and in particular if $(\tilde{z}, \tilde{t}, \cdot) \in \mathcal{V}[L_K]$. If not, it

returns $\perp$, otherwise $\mathcal{S}$ generates a payload key $K \leftarrow \mathsf{SE.KG}(pp_{\mathsf{SE}})$, computes $ct \leftarrow \mathsf{SE.Enc}(K, P)$, and computes $\gamma_{\tilde{z},\tilde{t}}$ for the zone–time pair $(\tilde{z}, \tilde{t})$ by sending K to $\mathcal{C}_b^{\mathsf{priv}}$. Simulator $\mathcal{S}$ then encrypts K with the keys for the other zone–time pairs in the query, and sets the ciphertext as the Enter algorithm does. For Send queries such that $\tilde{z} \notin Y$ or $t \neq \tilde{t}$, simulator $\mathcal{S}$ runs algorithm Send on the inputs.

For every Receive query on vehicle identity $\mathcal{V}$ and a ciphertext $\gamma = (t, Y \ni \tilde{z}, ((y, \gamma_{y,t})_{y \in Y}, ct))$ such that $t = \tilde{t}$, simulator $\mathcal{S}$ first checks whether there exists a zone $y \neq \tilde{z}$ in Y such that $(y, \tilde{t}, \cdot) \in \mathcal{V}[L_K]$. If so, it decrypts using $K_{y,\tilde{t}}$ and replies as algorithm Receive does, otherwise it checks whether $(\tilde{z}, \tilde{t}, \cdot) \in \mathcal{V}[L_K]$. If not, it returns $\perp$; otherwise it sends $\gamma_{\tilde{z},\tilde{t}}$ together with ct as a header to $\mathcal{C}$ and forwards its answert to $\mathcal{A}$. For Receive queries such that $\tilde{z} \notin Y$ or $t \neq \tilde{t}$, simulator $\mathcal{S}$ runs the corresponding algorithm on the inputs.

For an Open query on input m, simulator $\mathcal{S}$ parses m as (m', t, tok) and runs $\mathsf{DGSA.Open}(sk_{\mathcal{I}}, st_{\mathcal{I}}, (m, e(t), tok))$. It forwards the output to $\mathcal{A}$.

For a Corrupt query on an indentity $\mathcal{V}$, simulator $\mathcal{S}$ replies by sending to $\mathcal{A}$ the certificate $\mathcal{V}[cert_{\mathcal{V}}]$, all the credentials $\mathcal{V}[e_j, cred_{V,j}]$ and the key list $\mathcal{V}[L_K]$. Note that in the event in which $\mathcal{A}$ wins the game, no challenge zone $y^* \in Y^*$ can be active for $\mathcal{V}$ in time t^* (condition 2). In particular, if $t = \tilde{t}$ (simulator $\mathcal{S}$ will abort otherwise), $\mathcal{S}$ never has to send $K_{\tilde{z},t}$ to $\mathcal{A}$.

$\mathcal{A}$ ultimately outputs a challenge tuple $(\mathcal{V}, \gamma^*)$. Simulator $\mathcal{S}$ parses γ^* as $(t^*, Y^*, \gamma^{*'})$. If $\tilde{t} \neq t^*$ or $\tilde{z} \notin Y^*$, simulator $\mathcal{S}$ aborts; otherwise, in the event in which $\mathcal{A}$ wins, $\mathsf{Receive}(\mathcal{V}[L_k], \gamma^*) \neq \perp$, meaning that there exists $y^* \in Y^*$ such that $\mathsf{DAE.Dec}(K_{y^*,\tilde{t}}, \gamma_{y^*,\tilde{t}}) \neq \perp$. Such a y^* is equal to $\tilde{z}$ with probability at least $1/|Z|$ and $t = \tilde{t}$ with probability $1/|T|$. Moreover, since $\forall (t^*, Y, \gamma) \notin \mathcal{L}_{\mathsf{sent}}, \gamma \cap \gamma^* = \emptyset$, it follows that $\gamma_{\tilde{z},\tilde{t}}$ was never output by $\mathcal{C}_b^{\mathrm{priv}}$. Simulator $\mathcal{S}$ then sends $\gamma_{\tilde{z},\tilde{t}}$ to $\mathcal{A}$.

As SIG is assumed to be EUF-CMA secure, DGSA to satisfy traceabilty and PKE to be IND-CPA secure, simulator $\mathcal{S}$ is computationally indistinguishable from the ZE-scheme integrity challenger. Adversary $\mathcal{A}$ then wins the authenticity game with probability at least $(\varepsilon - \mathrm{negl}(\lambda))/|Z||T|$. As DAE satisfies authenticity and as $1/|Z||T|$ is non-negligible, ε must be negligible.