ZEEP: Zone Encryption with Enhanced Privacy for Vehicular Communication

ACM Reference Format:
. 2026. ZEEP: Zone Encryption with Enhanced Privacy for Vehicular Communication. In *ACM Asia Conference on Computer and Communications Security (ASIA CCS '26), June 1–5, 2026, Bangalore, India.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3779208.3785391

C Security Analysis Of ZEEP

We extend the formal definitions and experiments of ZE [1] to establish PH-CCA security, anonymity, ciphertext integrity, and ticket traceability for ZEEP. We also define additional experiments for two properties specific to ZEEP: identity escrow-freeness (IEF), ensuring privacy against honest-but-curious authorities, and revocability, ensuring that revoked users cannot get authorized. We define the lists, notations, and oracles used in the security experiments below:

Lists. The challenger maintains several lists to store information gathered by adversary from its interactions with the oracles. Table 1 summarizes these lists. Here $\mathcal{V}$ represents a vehicle with id vid.

Table 1: Lists used in the security experiments

List	Description
$\mathcal{L}_{honest}$	Enrolled honest vehicles $\{(\mathcal{V})\}$
$\mathcal{L}_{corrupt}$	Enrolled vehicles $\{(\mathcal{V})\}$ that are corrupt
$\mathcal{L}_{auth}$	Authorized tickets of vehicles per epoch e, i.e. $\{(q_{\mathcal{V}}, \mathcal{V}, e)\}$
$\mathcal{L}_{enter}$	Messages $\{((q_{\mathcal{V}}, \mathcal{V})/(q_{\mathcal{A}}, \mathcal{A}), M = (z, t, m))\}$ exchanged during Enter by honest vehicles $\mathcal{V}$ or adversary $\mathcal{A}$ using their tickets $q_{\mathcal{V}}$ or $q_{\mathcal{A}}$
$\mathcal{L}_{sent}$	Ciphertexts $\{\gamma\}$ generated by honest vehicles
$\mathcal{L}_{received}$	Decrypted ciphertexts $\{\gamma = (t, Y, \gamma')\}$
$\mathcal{L}_{opened}$	Opened transcripts M
$\mathcal{L}_{revoked}$	Revoked authentication tickets $\{q_{\mathcal{V}}\}$
$\mathcal{L}_{keys}$	Zone-keys $\{(z, t, K_{t,z})\}$ obtained by adversary $\mathcal{A}$ after corrupting vehicles

Notations. We use $O(sk_{\mathcal{I}}, pk_{\mathcal{I}}, reg_{\mathcal{I}})$ to represent the set of oracles initialized with secret key $sk_{\mathcal{I}}$, public key $pk_{\mathcal{I}}$, and registry state $reg_{\mathcal{I}}$. We adopt the following notations from ZE:

- $O.protocol.\mathcal{P}$: It allows the adversary to interact with the honest party $\mathcal{P}$ running $protocol$. It verifies whether $\mathcal{V} \in \mathcal{L}_{honest}$ and aborts otherwise.
- $O.protocol.\mathcal{P}\&\mathcal{R}$: It enables the adversary to initiate interactive $protocol$ between honest parties $\mathcal{P}$ and $\mathcal{R}$. The oracle updates its internal states but does not learn any output. It ensures $\mathcal{V} \in \mathcal{L}_{honest}$ for honest vehicles.
- Honest parties maintain their state as $\mathcal{P}[reg\mathcal{P}]$. For example, $\mathcal{V}[L_K]$ denotes the zone keys L_K held by $\mathcal{V}$ in epoch e. The adversary's state is denoted by $state_{\mathcal{A}}$, which stores information obtained during its interactions.

ASIA CCS '26, Bangalore, India

ACM ISBN 979-8-4007-2356-8/26/06
https://doi.org/10.1145/3779208.3785391

Oracles. Fig. 1 details the oracles that the adversary can query.

C.1 PH-CCA Security

PH-CCA Security ensures that an adversary cannot learn any information about the ciphertext in any zone and time period without executing the zone entry protocol. The experiment $\mathrm{Exp}^{\text{ph-cca-b}}_{Z,E,T}(\mathcal{A})$ for proving PH-CCA Security is detailed in Fig. 2. The adversary $\mathcal{A}$ outputs two challenge payloads P_0 and P_1 for target zones Y^* and time period t^*. $\mathcal{A}$ then receives a challenge ciphertext encrypting P_b for a random $b \in \{0, 1\}$ and must determine b with probability better than random guessing, excluding trivial win cases.

Definition C.1. *ZEEP is PH-CCA secure if for any efficient adversary $\mathcal{A}$, zone-set Z, epoch set E and time-period set T,*

$$|Pr[Exp^{ph-cca-0}_{Z,E,T}(\mathcal{A}) = 1] - Pr[Exp^{ph-cca-1}_{Z,E,T}(\mathcal{A}) = 1]| \leq negl(\lambda)$$

Theorem C.1. *ZEEP is PH-CCA secure if TFRAC satisfies misauthentication resistance, DGSA satisfies traceability, SE and PKE are IND-CPA secure, and DAE satisfies privacy and authenticity.*

Proof. We prove the PH-CCA security by showing that no efficient adversary $\mathcal{A}$ can distinguish between the PH-CCA challenger C^{ph-cca}_0 (encrypting P_0) and C^{ph-cca}_1 (encrypting P_1). Following the ZE security, we construct a series of hybrids between C^{ph-cca}_0 and C^{ph-cca}_1. Let the challenge zone set be $Y^* = \{y_1, y_2, \ldots, y_n\}$. For $i = 0, \ldots, n$, we define the hybrid algorithms below.

1. Hybrid Δ_i: behaves exactly like C^{ph-cca}_0, except that to compute the challenge ciphertext, it encrypts the first i zones with a payload key K' and the remaining zones with another key K, and encrypts P_0 with K.
2. Hybrid Δ'_i: behaves exactly like C^{ph-cca}_1, except that to compute the challenge ciphertext, it encrypts the first i zones with a payload key K and the remaining zones with another key K' and encrypts P_1 with K.

By definition $\Delta_0 = C^{ph-cca}_0$ and $\Delta'_n = C^{ph-cca}_1$. To prove the theorem, we show that $\mathcal{A}$ has negligible advantage in distinguishing between any two consecutive hybrids Δ_i and Δ_{i+1}. If $\mathcal{A}$ can distinguish them, then we can use it to build a reduction algorithm $\mathcal{S}$ to break **DAE privacy**.

$\mathcal{S}$ answers $\mathcal{A}$'s queries and interacts with the DAE privacy challenger Cb^{priv} ($b \in 0, 1$) with the challenge key $Kpriv$, while maintaining the same lists as the PH-CCA challenger. $\mathcal{S}$ receives pp_{DAE} from C^{priv}_b and generates rest ZEEP parameters $sk\mathcal{I}, pk_{\mathcal{I}}, pp_{TFRAC}$, and pp_{DGSA} and sends pp_{TFRAC}, pp_{DGSA} and pk_I to $\mathcal{A}$. $\mathcal{S}$ randomly selects a time period $\tilde{t}$ and sets zone key as $K_{y_{i+1},\tilde{t}} = K_{priv}$ and answers the queries as follows.

- $Enroll.\mathcal{V}\&\mathcal{I}$: $\mathcal{S}$ runs the protocol and stores the generated TFRAC credential.
- $Enroll.\mathcal{I}$:, $\mathcal{S}$ runs the protocol with $sk_{\mathcal{I}}$.
- $Authorize.\mathcal{V}\&\mathcal{I}$: $\mathcal{S}$ runs the protocol and stores the generated $TFRAC$ and $DGSA$ credentials.

Enrollment and Authorization

- $\mathcal{O}.Enroll.\mathcal{V}\&\mathcal{I}(sk_I,\cdot)$: On input $\mathcal{V}$, the adversary triggers the enrollment protocol between an honest vehicle $\mathcal{V}$ and the honest issuer $\mathcal{I}$. If $Enroll.\mathcal{V}$ returns a private output $\mathcal{V}[(cred_{TFRAC}, e)]$, then $\mathcal{V}$ is added to $\mathcal{L}_{honest}$.
- $\mathcal{O}.Enroll.\mathcal{I}(sk_I,\cdot)$: On input $\mathcal{V}$, the adversary, acting as a corrupt vehicle, triggers enrollment with the honest issuer $\mathcal{I}$ and adds $\mathcal{V}$ to $\mathcal{L}_{corrupt}$.
- $\mathcal{O}.Authorize.\mathcal{V}\&\mathcal{I}(sk_I,\cdot)$: On input $(q_{\mathcal{V}},\mathcal{V},e)$, the adversary initiates authorization of an honest vehicle $\mathcal{V}$ with the honest issuer $\mathcal{I}$. If the protocol outputs private credentials $\mathcal{V}[(cred_{TFRAC}, e')]$ and $\mathcal{V}[(cred_{\mathcal{V}}, e)]$, then $(q_{\mathcal{V}},\mathcal{V},e)$ is added to $\mathcal{L}_{auth}$.
- $\mathcal{O}.Authorize.\mathcal{I}(sk_I,\cdot)$: On input $(q_{\mathcal{V}},\mathcal{V},e)$, the adversary triggers authorization on behalf of a corrupt vehicle $\mathcal{V}$ and adds $(q_{\mathcal{V}},\mathcal{V},e)$ to $\mathcal{L}_{auth}$.

Entering and Exiting Zones

- $\mathcal{O}.Enter(\cdot)$: On input $(q_{\mathcal{V}},\mathcal{V},z,t,msg)$, the adversary triggers a zone-key request/response for an honest vehicle $\mathcal{V}$ in zone z at time t.
  - If $msg = request$, the oracle executes the Enter protocol with $\mathcal{V}$ as requester and honest vehicles $\mathcal{W}_i$ as responders.
  - If $msg = response$, the oracle allows $\mathcal{V}$ to respond to a request from a corrupt vehicle; the derived key $K_{z,t}$ is added to $\mathcal{V}[L_K]$.

  All messages $(q_{\mathcal{V}},\mathcal{V},(z,t,m))$ sent by honest vehicles are stored in $\mathcal{L}_{enter}$.
- $\mathcal{O}.Exit(\cdot)$: On input $(\mathcal{V},z,t)$, deletes the zone key $K_{z,t}$ from $\mathcal{V}[L_K]$.

Sending and Receiving Payloads

- $\mathcal{O}.Send(\cdot)$: On input $(\mathcal{V},P,Y,t)$, runs $\gamma/\bot \leftarrow Send(\mathcal{V}[L_K],P,Y,t)$ and adds γ to $\mathcal{L}_{sent}$.
- $\mathcal{O}.Receive(\cdot)$: On input $(\mathcal{V},\gamma)$, runs $m/\bot \leftarrow Receive(\mathcal{V}[L_K],\gamma)$ and adds γ to $\mathcal{L}_{received}$.

Opening, Corruption and Revocation

- $\mathcal{O}.Open(sk_I, reg_I,\cdot)$: On input M, returns $q_{\mathcal{V}} \leftarrow Open(sk_I, reg_I, M)$ and adds M to $\mathcal{L}_{opened}$.
- $\mathcal{O}.Corrupt(\cdot)$: On input $\mathcal{V}$, returns the full internal state of $\mathcal{V}$, adds $\mathcal{V}$ to $\mathcal{L}_{corrupt}$, and all keys to $\mathcal{L}_{keys}$.
- $\mathcal{O}.Revoke(sk_I,\cdot)$: On input authentication ticket $q_{\mathcal{V}}$, blocklists or penalizes $q_{\mathcal{V}}$ and updates $\mathcal{L}_{revoked}$.

Note. The notation "$protocol^*$" denotes a modified oracle with a specific response strategy. This applies to $Enter^*$, $Exit^*$, $Authorize^*$, $Send^*$, and $Receive^*$.

Figure 1: Adversarial Oracles and Capabilities

Experiment $\mathbf{Exp}^{\mathbf{ph\text{-}cca\text{-}b}}_{Z,E,T}(\mathcal{A})$:
$pp \leftarrow Setup(1^\lambda, Z, E, T)$
$(sk_{\mathcal{I}}, pk_{\mathcal{I}}, reg_{\mathcal{I}})) \leftarrow KeyGen.\mathcal{I}(pp)$

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

$\gamma^* \leftarrow Send(\mathcal{V}^*[L_K], P_b, Y^*, t^*)$ with $\gamma^* = (t^*, Y^*, \gamma^{*\prime})$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}$(guess, $\gamma^*, state_{\mathcal{A}}$)
return b' if $\mathcal{A}$ did not trivially win, i.e.:

1. $\forall (t^*, Y, \gamma) \in \mathcal{L}_{received}$: $\gamma \cap \gamma^{*'} = \emptyset$, i.e. $\mathcal{A}$ did not query decryption oracle on challenge ciphertext, and
2. $\forall y^* \in Y^*$: $(y^*, t^*, \cdot) \notin \mathcal{L}_{keys}$ i.e., $\mathcal{A}$ did not corrupt a vehicle with zone key for a challenge zone, and

3a. $\forall y^* \in Y^*$: $((q_{\mathcal{A}}, \mathcal{A}, y^*, t^*, \cdot) \notin \mathcal{L}_{enter})$, i.e. $\mathcal{A}$ did not enter a challenge zone in time t^* with corrupt vehicle or
3b. $\exists(\mathcal{A}, y^*, t^*, \cdot) \in \mathcal{L}_{enter}$ and $\forall \mathcal{V}_j \in \mathcal{L}_{corrupt}, \nexists(q_{\mathcal{V}_j}, \mathcal{V}_j, e(t^*)) \in \mathcal{L}_{auth}$ i.e. $\mathcal{A}$ entered a challenge zone without authorization.

Figure 2: Experiment for PH-CCA Security

- $Authorize.\mathcal{I}$: $\mathcal{S}$ runs the protocol with $pk_{\mathcal{I}}$.
- $Enter$: On input $(q_{\mathcal{V}},\mathcal{V},z,t,msg)$ on behalf of the honest vehicle $\mathcal{V}$, if $(z,t) = (y_{i+1,\tilde{t}})$, $\mathcal{S}$ executes $Enter$ with $\mathcal{A}$.
  - If $msg = request$, $\mathcal{S}$ generates $(ek, dk) \leftarrow PKE.KeyGen(1^\lambda)$, computes $tok_{\mathcal{V}} \leftarrow DGSA.GenTok(pk_{\mathcal{I}}, cred_{\mathcal{V}}, e, (z, t, ek))$ and sends $(M = (z,t,ek), tok_{\mathcal{V}})$ to $\mathcal{A}$.
  - If $msg = response$, then upon receiving $(y_{i+1}, \tilde{t}, ek, tok)$ from $\mathcal{A}$, $\mathcal{S}$ checks whether the request was originally sent by a non-corrupt vehicle in the same protocol execution—i.e., whether $\mathcal{A}$ is passively replaying some request. If so, $\mathcal{S}$ encrypts a random message instead of K with PKE. **IND-CPA security of PKE** ensures the indistinguishability of the hybrids here. If $\mathcal{V} \in \mathcal{L}_{corrupt}$ or if request is not from a non-corrupt vehicle in the same protocol execution (active attack), then $\mathcal{S}$ sends $\bot$ and aborts.

  If $\mathcal{A}$ wins the PH-CCA game, with $t^* = \tilde{t}$ as the challenge time period, the winning condition $3b$ ensures that no corrupt vehicle $\mathcal{V}_j$ is authorized in $e(\tilde{t})$. Therefore,
  - either there exists a vehicle with a valid credential for $e(\tilde{t})$ without enrollment, which occurs with negligible probability if **TFRAC is misauthentication-resistant**.
  - or no such vehicle exists, and the token sent by $\mathcal{A}$ is valid, which happens with negligible probability if **DGSA satisfies traceability**.

  Therefore after aborting, once $\mathcal{S}$ receives token from $\mathcal{A}$, $\mathcal{S}$ is computationally indistinguishable from Δ_i and Δ_{i+1}.
- $Exit$: On input $(\mathcal{V}, z, t)$, $\mathcal{S}$ deletes $(z, t, K_{z,t})$ from $\mathcal{V}[L_K]$.
- $Send$: On input $(\mathcal{V}, P, Y \ni y_{i+1}, \tilde{t})$, $\mathcal{S}$ checks if all zones in Y are active for $\mathcal{V}$ and if $(y_{i+1}, \tilde{t}, .) \in \mathcal{V}[L_K]$ and then generates payload key $DAE.KeyGen(pp_{DAE})$, computes $ct \leftarrow DAE.Enc(K, P)$, $\gamma_{y_{i+1},\tilde{t}}$ by sending K to C_b^{priv}. $\mathcal{S}$ then encrypts K with keys for other zone-time pairs in query and sets ciphertext as $Enter$ protocol.
- $Receive$: On input $\mathcal{V}$ and ciphertext $\gamma = (\tilde{t}, Y, ((y, \gamma_{y,\tilde{t}})_{y\in Y}, ct))$ such that $y_{i+1} \in Y$, $\mathcal{S}$ checks for any other active zone z for $\mathcal{V}$ by computing $K_P \leftarrow DAE.Dec(K_{z,\tilde{t}}, ct, \gamma_{z,\tilde{t}})$ and returns $P \leftarrow SE.Dec(K_P, ct)$. If $\gamma_{y_{i+1},\tilde{t}}$ and ct were included in a previous $Send$ query response, $\mathcal{S}$ returns the corresponding queried payload. Otherwise, it returns $\bot$ and aborts. **DAE authenticity** ensures that $\mathcal{A}$ can't generate a ciphertext that decrypts to a valid plaintext.
- $Open$: On query $M = (z, t, m), tok$ $\mathcal{S}$ runs $DGSA.OpenTok$ and sends output q to $\mathcal{A}$.
- $Revoke$: On input $q_{\mathcal{V}}$, blacklists $q_{\mathcal{V}}$ and informs $\mathcal{A}$.
- $Corrupt$: On input $\mathcal{V}$, sends the state of the vehicle, i.e. credentials and key list, to $\mathcal{A}$.

Note: For all queries where $z \neq y_{i+1}$ or $t \neq \tilde{t}$, $\mathcal{S}$ simply runs the corresponding protocol on the given inputs. In challenge phase, $\mathcal{A}$ outputs challenge tuples $(\mathcal{V}^*, P_0, P_1, Y^*, t^*)$.

- If $\mathcal{V}^* \in \mathcal{L}_{corrupt}$, $\mathcal{S}$ returns 0 same as PH-CCA challenger.
- If $\tilde{t} \neq t^*$, $\mathcal{S}$ sends $\perp$ and aborts.
- Otherwise, $\mathcal{S}$ generates payload keys K and K' and sends them to C and receives a ciphertext $\gamma^*_{y_{i+1},\tilde{t}}$ from C^{priv}_b. For $k = 1, \ldots, i$, $\mathcal{S}$ computes $\gamma_{y_k,\tilde{t}} \leftarrow DAE.Enc(K_{y_k,\tilde{t}}, K')$, for $k = i, \ldots, n$, computes $\gamma^*_{y_k,\tilde{t}} \leftarrow DAE.Enc(K_{y_k,\tilde{t}}, K)$, $ct \leftarrow DAE.Enc(K, P_0)$ and sends $\gamma^* = (\tilde{t}, Y^*, ((y_k, \gamma_{y_k,\tilde{t}})_{k\leq i}, (y_{i+1}, \gamma^*_{y_{i+1},\tilde{t}}), (y_k, \gamma_{y_k,\tilde{t}})_{k>i+1}, ct))$ to $\mathcal{A}$.

After challenge phase, to answer a *Receive* query on $(\mathcal{V}, \gamma)$, $\mathcal{S}$ checks that no part of γ is replayed from the challenge ciphertext γ' and then proceeds as before. Other queries are answered as before.

At the end of the game, $\mathcal{S}$ forwards the bit b of $\mathcal{A}$ to C. $\mathcal{S}$ perfectly simulates C^{ph-cca}_b to $\mathcal{A}$ under the stated assumptions. As $\tilde{t} = t^*$ with probability $1/|T|$, thus for $|Y^*| = n$ zones, the adversarial advantage is given by

$$Adv_{\Delta_i,\Delta_{i+1}(\mathcal{A})} - Q_1 Adv^{misauth}_{TFRAC}(\mathcal{S}(\mathcal{A})) - Q_2 Adv^{trace}_{DGSA}(\mathcal{S}(\mathcal{A})) - Q_3 Adv^{auth}_{DAE}(\mathcal{S}(\mathcal{A})) - Q_4 Adv^{ind-cpa}_{PKE}(\mathcal{S}(\mathcal{A})) \leq n|T| Adv^{priv}_{DAE}(\mathcal{S}(\mathcal{A}))$$

where Q_1, Q_2 are no. of active *Enter* queries, Q_3 is no. of *Recieve* queries with target zone as the only active zone, and Q_4 is the no. of passive *Enter* queries. Therefore, Δ_i and Δ_{i+1} are computationally indistinguishable.

We can use a similar reduction to **DAE privacy** for proving the indistinguishability of Δ'_i and Δ'_{i+1}. Furthermore, we can reduce the **IND-CPA security of SE** to the computational indistinguishability of Δ_n and Δ'_0. $\mathcal{S}$ can set K as the challenge key and forward (P_0, P_1) as the challenge tuples in the SE IND-CPA game. It can respond to all the *Send* queries (excluding the challenge) by generating fresh payload keys. Additionally, since $\mathcal{S}$ knows all the keys for active zones, it can address all the other queries as well. Therefore the advantage of $\mathcal{A}$ in the PH-CCA game is given by,

$$Adv^{ph-cca}_{ZEEP}(\mathcal{A}) \leq 2n|T| Adv^{priv}_{DAE}(\mathcal{S}(\mathcal{A})) + Adv^{ind-cpa}_{SE}(\mathcal{S}(\mathcal{A})).$$

Thus, if TFRAC is misauthentication resistant, SE and PKE are IND-CPA secure and DAE satisfies privacy and authenticity, then ZEEP is PH-CCA secure. □

C.2 Anonymity

ZEEP anonymity ensures that ciphertexts sent during the *Enter* protocol reveals no information about vehicle identity (anonymity) and remains unlinkable. The experiment $Exp^{ano-b}_{Z,E,T}(\mathcal{A})$ for anonymity is detailed in Fig. 3.

Definition C.2. *ZEEP satisfies anonymity if for any efficient adversary $\mathcal{A}$, zone-set Z, epoch set E and time-period set T,*

$$|Pr[Exp^{ano-0}_{Z,E,T}(\mathcal{A}) = 1] - Pr[Exp^{ano-1}_{Z,E,T}(\mathcal{A}) = 1]| \leq negl(\lambda)$$

Theorem C.2. *ZEEP satisfies anonymity if TFRAC is misauthentication resistant and DGSA satisfies anonymity.*

Proof. Let $\mathcal{A}$ be an adversary in the ZEEP anonymity game, making Q queries to the *Enter** oracle. We define Δ_i as an algorithm that simulates the ZEEP anonymity game challenger, only differing in its handling of *Enter** queries. For the first i *Enter** queries made with bit d, Δ_i responds using $(q_{\mathcal{V}_{1-d}}, \mathcal{V}_{1-d})$, while for the remaining queries, it responds using $(q_{\mathcal{V}_d}, \mathcal{V}_d)$. Let C^{ano}_b represents the ZEEP anonymity challenger using $(q_{\mathcal{V}_b}, \mathcal{V}_b)$, then by definition $\Delta_0 = C^{ano}_0$ and $\Delta_Q = C^{ano}_1$.

Experiment $\mathbf{Exp}^{ano-b}_{Z,E,T}(\mathcal{A})$
$pp \leftarrow Setup(1^\lambda, Z, E, T)$
$(sk_\mathcal{I}, pk_\mathcal{I}, reg_\mathcal{I}) \leftarrow KeyGen.\mathcal{I}(pp)$

Initialize all oracles $O(sk_\mathcal{I}, pk_\mathcal{I}, reg_\mathcal{I})$
$((q_{\mathcal{V}_0}\mathcal{V}_0), (q_{\mathcal{V}_1}\mathcal{V}_1), state_\mathcal{A}) \leftarrow \mathcal{A}^O(\text{choose}, pp, pk_\mathcal{I})$
Abort if $\mathcal{V}_0$ and $\mathcal{V}_1$ have different states, i.e. for $d \in \{0, 1\}$: $\nexists e_i$ s.t. $(q_{\mathcal{V}_d}, \mathcal{V}_d, e_i) \in \mathcal{L}_{auth}$ and $(q_{\mathcal{V}_{1-d}}, \mathcal{V}_{1-d}, e_i) \notin \mathcal{L}_{auth}$ and $\mathcal{V}_0[L_K] = \mathcal{V}_1[L_K]$. It ensures that $Enter, Exit$ is not queried for the challenge vehicles unless both vehicles have been authorized in the same epoch.

Use the challenge oracles $O^*(b)$ for the queries on $Enter^*$, $Exit^*$, $Send^*$, and $Receive^*$
$b' \leftarrow \mathcal{A}^{O^*}(\text{guess}, state_\mathcal{A})$
Return b' if $\mathcal{A}$ did not trivially win, i.e., for $d \in \{0, 1\}$: $\mathcal{V}_d \in \mathcal{L}_{honest}$ and $\forall M \in \mathcal{L}^*_{enter} : M \notin \mathcal{L}_{opened}$, i.e. adversary did not open a message sent during $Enter$ by a challenge vehicle.

Figure 3: Experiment for Anonymity

We prove that if $\mathcal{A}$ can distinguish between any two consecutive hybrids Δ_i and Δ_{i+1}, then we can build a reduction $\mathcal{S}$ which can use $\mathcal{A}$ as a subroutine to win the **DGSA anonymity** game.

$\mathcal{A}$'s advantage in the ZEEP anonymity game is thus bounded by $Q \times Adv_{\Delta_i,\Delta_{i+1}}$, where $Adv_{\Delta_i,\Delta_{i+1}}$ is the advantage of $\mathcal{A}$ in distinguishing Δ_i from Δ_{i+1} for any $0 \leq i \leq Q - 1$.

$\mathcal{S}$ receives pp_{DGSA} and pk_{DGSA} from the DGSA challenger, generates $sk_{TFRAC}, pk_{TFRAC} \leftarrow TFRAC.KeyGen(pp_{TFRAC})$ and parameters for other ZEEP protocols. $\mathcal{S}$ sends the public parameters and pk_{DGSA}, pk_{TFRAC} to $\mathcal{A}$. $\mathcal{S}$ answers the queries as follows.

- *Enroll.*$\mathcal{V}\&\mathcal{I}$: $\mathcal{S}$ runs the protocol and stores the generated TFRAC credentials.
- *Enroll.*$\mathcal{I}$: $\mathcal{S}$ runs the protocol using sk_{TFRAC}.
- *Authorize.*$\mathcal{V}\&\mathcal{I}$: On $(q_\mathcal{V}, \mathcal{V}, e)$ for vehicle $\mathcal{V}$ in epoch e, simulator $\mathcal{S}$ runs *TFRAC.Auth* and queries *DGSA.Issue* oracle of $C_{DGSA,b}$ and stores the output (updated TFRAC and DGSA) credentials.
- *Authorize.*$\mathcal{I}$: On $(\mathcal{V}, e)$, $\mathcal{S}$ executes *Authorize* with $\mathcal{A}$. Let $\mathcal{V}_0, \mathcal{V}_1$ be the challenge vehicles used by $\mathcal{A}$ in the challenge phase to win the game.
  - If $\mathcal{A}$ queries on $\mathcal{V} = \mathcal{V}_0$ or $\mathcal{V} = \mathcal{V}_1$, with $(cred_{TFRAC}, cred_\mathcal{V})$, $\mathcal{S}$ check $\mathcal{V} \in \mathcal{L}_{honest}$; If not it aborts. $\mathcal{S}$ is indistinguishable from ZEEP anonymity challenger because of **TFRAC misuthentication-resistance**.
  - If $\mathcal{V} \neq \mathcal{V}_0$ or $\mathcal{V}_1$, then $\mathcal{S}$ runs *TFRAC.Auth* using sk_{TFRAC}, pk_{TFRAC}. If the authentication succeeds, it outputs an updated TFRAC credential $cred'_{TFRAC}$ and queries *DGSA.Issue* oracle of $C_{DGSA,b}$ for DGSA credential and forwards the responses to $\mathcal{A}$. Otherwise, it .
  - *Enter* : $\mathcal{S}$ uses *DGSA.AuthTok* and *DGSA. VerifyTok* on the input to generate and verify authentication tokens.
  - *Exit* : On $(\mathcal{V}, z, t)$, $\mathcal{S}$ deletes $(z, t, K(z, t))$ from $\mathcal{V}[L_K]$ that it locally maintains for $\mathcal{V}$.
  - For *Send* and *Recieve*, $\mathcal{S}$ runs the corresponding protocols
  - *Open* : On input M, $\mathcal{S}$ queries the *DGSA.OpenTok* oracle of $C_{DGSA,b}$ and returns q to $\mathcal{A}$.
  - *Revoke* : On input $q_\mathcal{V}$, blacklists $q_\mathcal{V}$ and informs $\mathcal{A}$.

– $Corrupt$: On input $\mathcal{V}$, sends the state of the vehicle, i.e. credentials and key list, to $\mathcal{A}$.

In the challenge phase, $\mathcal{A}$ outputs challenges $(q_{\mathcal{V}_0}, V_0)$ and $(q_{\mathcal{V}_1}, \mathcal{V}_1)$. $\mathcal{S}$ checks if both tickets are authorized in same epochs and $\mathcal{V}_0[L_K] = \mathcal{V}_1[L_K]$. If not it aborts. $\mathcal{S}$ randomly selects a zone-time pair $(\tilde{z}, \tilde{t})$ where both the challenge vehicle tickets $q_{\mathcal{V}_0}$ and $q_{\mathcal{V}_1}$ are authorized. After the challenge phase, the oracles $Enter$, $Exit$, $Send$, and $Receive$ are replaced by $Enter^*$, $Exit^*$, $Send^*$, and $Receive^*$, respectively. For all other oracle queries involving a bit d, $\mathcal{S}$ responds as it did before the challenge phase.

- For the $(*)$ queries up to the i^{th} $Enter^*$ query with a bit d, $\mathcal{S}$ uses $(q_{\mathcal{V}_{1-d}}, \mathcal{V}_{1-d})$ to respond. It verfies $(q_{\mathcal{V}_{1-d}}, \mathcal{V}_{1-d}, e) \in \mathcal{L}_{auth}$ and executes the ZEEP sub-protocols during $Enter$, by querying $C_{DGSA,b}$ for $DGSA.AuthTok$, and executing the remaining protocols itself.
- To answer $Exit^*$, $Send^*$ and $Receive^*$ queries for a bit d the simulator uses the state of $\mathcal{V}_{1-d}$ that it locally maintains.
- For the $(i+1)^{th}$ $Enter^*$ query on input (d, z, t, msg) $\mathcal{S}$ aborts if $(z, t) \neq (\tilde{z}, \tilde{t})$. Otherwise, if $msg = request$, $\mathcal{S}$ generates ek for PKE and sends $(q_{\mathcal{V}_0}, q_{\mathcal{V}_1}, e(\tilde{t}), M = (z, t, ek))$ as a challenge tuple to $C_{DGSA,b}$. If $msg = response$, $\mathcal{S}$ sends $(q_{\mathcal{V}_0}, q_{\mathcal{V}_1}, e(\tilde{t}), M = (z, t, ct))$ as challenge to $C_{DGSA,b}$.
- To answer the remaining $(*)$ queries, $\mathcal{S}$ uses $(q_{\mathcal{V}_d}, \mathcal{V}_d)$.

The winning condition implies that $\mathcal{A}$ never queried the $Open$ oracle on any message exchanged during $Enter^*$. Therefore the distribution of $\mathcal{S}'s$ responses except for the $(i+1)^{th}$ $Enter^*$ query is identical to those in Δ_i and Δ_{i+1}.

At the end of the game, $\mathcal{A}$ outputs a decision bit b', which $\mathcal{S}$ forwards to $C_{DGSA,b}$. The advantage of $\mathcal{A}$ in distinguishing Δ_i from Δ_{i+1} is therefore given as

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

where $negl(\lambda)$ is the negligible advantage due to the misauthentication resistance of TFRAC and $Adv^{ano}_{DGSA}(S(\mathcal{A}))$ is the advantage of $\mathcal{S}$ running $\mathcal{A}$ as a subroutine in the DGSA anonymity game. Thus,

$$Adv^{ano}_{ZEEP}(\mathcal{A}) - negl(\lambda) \leq Q|Z||T|Adv^{ano}_{DGSA}(S(\mathcal{A})).$$

Thus, a non-negligible advantage for $\mathcal{A}$ in ZEEP anonymity game implies a non-negligible advantage for $\mathcal{S}$ in DGSA anonymity game. □

C.3 Ciphertext Integrity

Ciphertext integrity ensures that an adversary cannot generate a valid ciphertext for any zone z and time period t without possessing the corresponding zone key $K_{z,t}$. The experiment $\mathrm{Exp}^{integrity}_{Z,Epoch,T}(\mathcal{A})$ to prove ciphertext integrity is detailed in Fig. 4.

Definition C.3. *ZEEP satisfies ciphertext integrity if for any efficient adversary $\mathcal{A}$, zone-set Z, epoch set E and time-period set T,*

$$Pr[\boldsymbol{Exp}^{integrity}_{Z,E,T}(\mathcal{A}) = 1] \leq negl(\lambda).$$

To win, the adversary must output a fresh and valid ciphertext γ^* for zones whose keys it does not know, and also output an honest vehicle $\mathcal{V}$ that decrypts γ^* to $P \neq \perp$.

Theorem C.3. *ZEEP satisfies ciphertext integrity if DAE satisfies authenticity, TFRAC is misauthentication resistant if DGSA satisfies traceability, and PKE is IND-CPA secure.*

Experiment $\mathbf{Exp}^{integrity}_{Z,E,T}(\mathcal{A})$
$pp \leftarrow Setup(1^\lambda, Z, E, T)$
$(sk_{\mathcal{I}}, pk_{\mathcal{I}}, reg_{\mathcal{I}}) \leftarrow KeyGen.\mathcal{I}(pp)$

Initialize all the oracles as $O(sk_{\mathcal{I}}, pk_{\mathcal{I}}, reg_{\mathcal{I}})$
$(\mathcal{V}, \gamma^*) \leftarrow \mathcal{A}^O(\text{forge}, pp, pk_{\mathcal{I}})$
Parse $\gamma^* = (t^*, Y^*, \gamma^{*'})$, abort if $\mathcal{V} \in \mathcal{L}_{corrupt}$

Return 1 if $Receive(\mathcal{V}[L_K], \gamma^*) \neq \perp$ and $\mathcal{A}$ didn't trivially win, i.e.

(1) $\forall(t^*, Y, \gamma) \in \mathcal{L}_{sent} : \gamma \cap \gamma^{*'} = \emptyset$, i.e. γ^* does not include any ciphertext previously queried by $\mathcal{A}$ to the $Send$ oracle, and
(2) $\forall y^* \in Y^* : (y^*, t^*, \cdot) \notin \mathcal{L}_{keys}$ i.e., $\mathcal{A}$ has not corrupted any vehicle that knows the key of challenge zone, and
3a. $\forall y^* \in Y^* : ((\cdot, \mathcal{A}, y^*, t^*, \cdot) \notin \mathcal{L}_{enter})$, i.e. either $\mathcal{A}$ has not entered challenge zone in time t^*, or
3b. $\exists(\cdot, \mathcal{A}, y^*, t^*) \in \mathcal{L}_{enter}$ and $\forall \mathcal{V}_j \in \mathcal{L}_{corrupt} : \nexists(\cdot, \mathcal{V}_j, e(t^*)) \in \mathcal{L}_{auth}$, i.e. $\mathcal{A}$ has entered challenge zone in time t^* without authorization.

Figure 4: Experiment for Ciphertext Integrity

Proof. Assuming that TFRAC is misauthentication-resistant, DGSA satisfies traceability, and PKE is IND-CPA secure, we prove that ZEEP achieves ciphertext integrity by reducing it to the **authenticity of the DAE scheme.**

Let $\mathcal{A}$ be an adversary that wins the ZEEP ciphertext integrity game with probability at least ϵ. Let $\mathcal{S}$ be the simulator that runs $\mathcal{A}$ as a subroutine and interacts with the DAE authenticity challenger C^{priv}_b. The challenger generates secret key $K \leftarrow DAE.KeyGen(pp)$ and provides pp to $\mathcal{S}$. $\mathcal{S}$ generates the remaining protocol parameters. It runs $(sk_{TFRAC}, pk_{TFRAC}) \leftarrow TFRAC.KeyGen(pp_{TFRAC})$, $(sk_{DGSA}, pk_{DGSA}) \leftarrow DGSA.KeyGen\ (pp_{DGSA})$ and sends all the public parameters and $pk_{\mathcal{I}} = (pk_{TFRAC}, pk_{DGSA})$ to $\mathcal{A}$.

$\mathcal{S}$ randomly selects a zone, time period $(\tilde{z}, \tilde{t})$ and sets the zone key $K_{\tilde{z},\tilde{t}} = K$. It then responds to the adversary's queries as follows.

- $Enroll.\mathcal{V}\&\mathcal{I}$: $\mathcal{S}$ runs the protocol and stores the generated TFRAC credential.
- $Enroll.\mathcal{I}$: $\mathcal{S}$ runs the protocol with $sk_{\mathcal{I}}$.
- $Authorize.\mathcal{V}\&\mathcal{I}$: $\mathcal{S}$ runs the protocol and stores generated $TFRAC$ and $DGSA$ credentials.
- $Authorize.\mathcal{I}$: $\mathcal{S}$ runs the protocol with $pk_{\mathcal{I}}$.
- $Enter$: On input $(q_{\mathcal{V}}, \mathcal{V}, z, t, msg)$ on behalf of honest vehicle $\mathcal{V}$, if $(z, t) = (\tilde{z}, \tilde{t})$ and $\mathcal{V} \notin \mathcal{L}_{corrupt}$, $\mathcal{S}$ executes $Enter$ protocol with $\mathcal{A}$.
  - If $msg = request$, then simulator $\mathcal{S}$ generates $(ek, dk) \leftarrow PKE.KeyGen(1^\lambda)$, computes authentication token $tok \leftarrow DGSA.GenTok(pk_{\mathcal{I}}, cred_{\mathcal{V}}, e, (z, t, ek))$, and sends $(z, t, ek, tok_{\mathcal{V}})$ to $\mathcal{A}$.
  - If $msg = response$, then on input $(\tilde{z}, \tilde{t}, ek, tok)$ from $\mathcal{A}$, $\mathcal{S}$ checks whether the request is from a non-corrupt vehicle identity in the same protocol execution (i.e. $\mathcal{A}$ is replaying a request of a non-corrupt vehicle (passive attack). If so, $\mathcal{S}$ encrypts a random message instead of K with PKE. **IND-CPA security of PKE** ensures the indistinguishability from the actual protocol. If $\mathcal{V} \in \mathcal{L}_{corrupt}$, or if the request is not from a non-corrupt vehicle in the same protocol execution (active attack) then $\mathcal{S}$ sends $\perp$ and aborts.

If $\mathcal{A}$ wins the ciphertext integrity game, with $t^* = \tilde{t}$ as the challenge time period, the winning condition 3b ensures that no corrupt vehicle $\mathcal{V}_j$ is authorized in $e(\tilde{t})$. Therefore, either
- there exists a vehicle with a valid credential for $e(\tilde{t})$ without enrollment, which occurs with negligible probability if **TFRAC is misauthentication-resistant**, or
- no such vehicle exists, and the token sent by the adversary is valid, which happens with negligible probability if **DGSA satisfies traceability**.

Therefore, by aborting the protocol upon receiving a token from $\mathcal{A}$, simulator's response is computationally indistinguishable from the real protocol execution.

- *Exit* : On input $(\mathcal{V}, z, t)$, $\mathcal{S}$ simply deletes $(z, t, K_{z,t})$ from $\mathcal{V}[L_K]$ that it locally maintains for $\mathcal{V}$.
- *Send* : On input $(\mathcal{V}, P, Y \ni \tilde{z}, \tilde{t})$, $\mathcal{S}$ checks whether all zones in Y are active for $\mathcal{V}$ and $(\tilde{z}, \tilde{t}, .) \in \mathcal{V}[L_K]$. If not it aborts, otherwise, $\mathcal{S}$ generates payload key $K \leftarrow SE.KeyGen(pp_{SE})$, computes $ct \leftarrow SE.Enc(K, P)$ and $\gamma_{\tilde{z},\tilde{t}}$ by sending K to C_b^{priv}. $\mathcal{S}$ encrypts K with keys of remaining zone-time pairs in the query and sets ciphertext similar to *Enter*.
- *Receive* : On ciphertext $\gamma = (t, Y \ni \tilde{z}, ((y, \gamma_y, t))_{y \in Y}, ct)$ such that $t = \tilde{t}$, $\mathcal{S}$ checks if there exists a zone $y \neq \tilde{z}$ in Y such that $(y, t, \cdot) \in \mathcal{V}[L_K]$. If so, it decrypts the ciphertext using $K_{y,t}$ and replies as the real *Receive* protocol. Otherwise, it checks whether $(\tilde{z}, t, \cdot) \in \mathcal{V}[L_K]$. If not, it returns $\perp$; if yes, it sends $(\gamma_{\tilde{z}}, \tilde{t}, ct)$ as a header to the challenger C and forwards its response to $\mathcal{A}$.
- *Open*, *Revoke*, *Corrupt* queries are handled exactly as in the previous games.

Note: For all queries where $z \neq \tilde{z}$ or $t \neq \tilde{t}$, $\mathcal{S}$ simply runs the corresponding protocol on the given inputs.

Finally $\mathcal{A}$ outputs a challenge $(\mathcal{V}, \gamma^*)$, where γ^* is parsed as $(t^*, Y^*, \gamma^{*\prime})$. If $t^* \neq \tilde{t}$ or $\tilde{z} \notin Y^*$, $\mathcal{S}$ aborts. Otherwise in the case if $\mathcal{A}$ wins, $Receive(\mathcal{V}[L_K], \gamma^*) \neq \perp$, then some $y^* \in Y^*$ exists which satisfies $DAE.Dec(K_{y^*,\tilde{t}}, \gamma_{y^*,\tilde{t}}) \neq \perp$. The probability of such $y^* = \tilde{z}$ is $1/|Z|$, and such $t^* = \tilde{t}$ is $1/|T|$. Additionally, since γ^* was not in the set of sent ciphertexts, $\gamma_{\tilde{z},\tilde{t}}$ must be a fresh one. $\mathcal{S}$ then forwards it to $C_{DAE,b}^{auth}$, breaking DAE authenticity. Hence, if $\mathcal{A}$ succeeds with probability ϵ, $\mathcal{S}$ breaks DAE authenticity with probability at least $\epsilon/(|Z||T|) - negl(\lambda)$. As **DAE satisfies authenticity** and $|Z|, |T|$ are polynomial, ϵ must be negligible.

□

C.4 Ticket traceability

ZEEP ensures that any vehicle holding a zone key $K_{z,t}$ must have entered zone z at time t by sending a message that allows the corresponding authentication ticket to be traced and blocklisted. The experiment $\text{Exp}_{Z,E,T}^{trace}(\mathcal{A})$ is detailed in Fig. 5.

Definition C.4. *ZEEP satisfies ticket traceability if for any PPT adversary $\mathcal{A}$, zone-set Z, epoch set E, and time-period set T,*

$$Pr[Exp_{Z,E,T}^{trace}(\mathcal{A}) = 1] \leq negl(\lambda).$$

Theorem C.4. *ZEEP satisfies ticket traceability if DGSA satisfies unforgeability and traceability, PKE is IND-CPA secure, and TFRAC is misauthentication resistant.*

Experiment $\text{Exp}_{Z,E,T}^{trace}(\mathcal{A})$:
$pp \leftarrow Setup(1^\lambda, Z, E, T)$
$(sk_{\mathcal{I}}, pk_{\mathcal{I}}, reg_{\mathcal{I}}) \leftarrow KeyGen.\mathcal{I}(pp)$

Initialize all the oracles as $\mathcal{O}(sk_{\mathcal{I}}, reg_{\mathcal{I}})$
$(z^*, t^*, K_{z^*,t^*}) \leftarrow \mathcal{A}^{\mathcal{O}}(\text{forge}, pp, pk_{\mathcal{I}})$
Check if $\exists \mathcal{V} \in \mathcal{L}_{honest}$ with $K_{z^*,t^*} \in \mathcal{V}[L_K]$, otherwise abort.

Return 1 if knowledge of K_{z^*,t^*} cannot be traced with *Open* to the authentication ticket of a corrupt vehicle, i.e.

(1) $K_{z^*,t^*} \notin \mathcal{L}_{keys}$, i.e., $\mathcal{A}$ has not corrupted an honest vehicle which already had key K_{z^*,t^*} and
(2a) $\forall(\cdot, z^*, t^*, m_j) \in \mathcal{L}_{enter}$ s.t. $q_{\mathcal{V}_j} \leftarrow Open(sk_{\mathcal{I}}, reg_{\mathcal{I}}, m_j)$, then $(\nexists(q_{\mathcal{V}_j}, \mathcal{V}_j, e(t^*)) \in \mathcal{L}_{auth}))$:$(\mathcal{V}_j \in \mathcal{L}_{corrupt})$, i.e. if an enter message opens to ticket $q_{\mathcal{V}_j}$, then no corrupt vehicle authorized using it in epoch $e(t^*)$.

Figure 5: Experiment for Traceability to Tickets

Proof. Let $\mathcal{A}$ be an adversary that wins the ticket traceability game with ϵ probability. To win the game $\mathcal{A}$ outputs a tuple (z^*, t^*, K_{z^*,t^*}) such that there exists at least one honest vehicle $\mathcal{V} \in \mathcal{L}_{honest}$ which accepted the key. The winning conditions implies that $K_{z^*,t^*} \notin \mathcal{L}_{keys}$, which requires one of the following condition to hold.

(1) Case 1: No enter message involving $m_j = (z^*, t^*, \cdot)$ is sent, i.e. $m_j = (z^*, t^*, \cdot)$ does not exist in $\mathcal{L}_{enter}$
(2) Case 2: There exists atleast one enter message involving $m_j = (z^*, t^*, \cdot)$ in $\mathcal{L}_{enter}$

Case 1: Since no message of the form $m_j = (z^*, t^*, \cdot)$ exists in $\mathcal{L}_{\text{enter}}$, $\mathcal{A}$ does not interact actively with the system and cannot influence the key list. $\mathcal{A}$ only sees ciphertexts generated via PKE during Enter. Hence, traceability reduces to the **IND-CPA security of PKE**.
Case 2: In this case atleast one such message m_j exists, then the winning condition ensures that if $q_{\mathcal{V}_j} \leftarrow Open(sk_I, st_I, m_j)$ is the ticket recovered from m_j then the following scenarios can occur:

(1) There is no tuple $(q_{\mathcal{V}_j}, \mathcal{V}_j, e(t)^*) \in \mathcal{L}_{\text{auth}}$ such that $\mathcal{V}_j \in \mathcal{L}_{\text{corrupt}}$. It shows some corrupt vehicle $\mathcal{V}_j$ did not get authorized in epoch $e(t)^*$ using $q_{\mathcal{V}}$. In this case, the traceability of ZEEP can be reduced to the **traceability of DGSA**. We construct a simulator $\mathcal{S}$ that runs the adversary $\mathcal{A}$ as a subroutine. The simulator $\mathcal{S}$ uses the message m_j and token *tok* output by $\mathcal{A}$ as a forgery for the vehicle $\mathcal{V}_j$ in epoch $e(t)^*$.
(2) There exist $(q_{\mathcal{V}_j}, \mathcal{V}_j, e(t)^*) \in \mathcal{L}_{\text{auth}}$ such that $\mathcal{V}_j \in \mathcal{L}_{honest}$. It shows either *i*) $\mathcal{A}$ replayed a valid message from the honest vehicle or *ii*) $\mathcal{A}$ forged a token that opens to an authentication ticket $q_{\mathcal{V}}$ of an honest vehicle that never computed it. In the first case, ZEEP traceability can be reduced to the **IND-CPA security of PKE** similar to Case 1. In the second case, traceability can be reduced to **DGSA traceability** similar to Case 2.1.
(3) There exists $(q_{\mathcal{V}_j}, \mathcal{V}_j, e(t)^*) \in \mathcal{L}_{\text{auth}}$ such that $\mathcal{V}_j \notin \mathcal{L}_{honest}$ and $\mathcal{V}_j \notin \mathcal{L}_{corrupt}$. This implies $\mathcal{V}_j$ was never enrolled i.e. $\mathcal{A}$ forged a TFRAC credential. In this case, ZEEP traceability can be reduced to **TFRAC misauthentication resistance**.

Since all primitives are assumed to be secure, the adversary's advantage in breaking the traceability of ZEEP is negligible. □

Experiment $\mathbf{Exp}^{revoke}_{Z,E,T}(\mathcal{A})$:
$pp \leftarrow Setup(1^\lambda, Z, E, T)$
$(sk_{\mathcal{I}}, pk_{\mathcal{I}}, reg_{\mathcal{I}}) \leftarrow KeyGen.\mathcal{I}(pp)$
Initialize all the oracles as $O(sk_{\mathcal{I}}, reg_{\mathcal{I}})$

$(\mathcal{V}_j, m_j, e(t)) \leftarrow \mathcal{A}^O(\text{choose}, pp, pk_{\mathcal{I}})$

Revocation: C opens the ticket of Enter message $q \leftarrow Open(sk_{\mathcal{I}}, reg_{\mathcal{I}}, m_j)$ and blocklists it.
$\mathcal{A}$ outputs $(q', V_j, m'_j, e'(t'))$ for an epoch $e'(t') > e(t)$.

Return 1 if $q' \leftarrow Open(sk_{\mathcal{I}}, reg_{\mathcal{I}}, m'_j)$, i.e. $\mathcal{A}$ is able to successfully send enter messages using the ticket q' in epoch $e'(t')$.

Figure 6: Experiment for Revocability

C.5 Revocability

ZEEP ensures revocability: if a vehicle's authentication ticket is blocklisted in an epoch, it cannot authorize or participate in zone communications in later epochs. The experiment $\mathsf{Exp}^{revoke}_{Z,E,T}(\mathcal{A})$ in Fig. 6 formalizes this notion.

Definition C.5. *ZEEP satisfies revocability if for any efficient adversary $\mathcal{A}$, zone-set Z, epoch set E, and time-period set T,*

$$Pr[Exp^{revoke}_{Z,E,T}(\mathcal{A}) = 1] \leq negl(\lambda).$$

Theorem C.5. *ZEEP satisfies revocability if TFRAC satisfies revocability and DGSA satisfies traceability.*

Proof. We structure the proof in three phases:

Pre-revocation. The adversary $\mathcal{A}$ can interact with the revocability challenger C by querying *Enroll*, *Authorize*, *Enter*, *Exit*, *Send*, and *Receive*, all of which are handled as in prior security games.

Revocation. $\mathcal{A}$ outputs a challenge vehicle $\mathcal{V}_j$ along with a zone-entry message m_j in some epoch $e(t)$. The challenger opens this message to obtain the corresponding authentication ticket, $q \leftarrow Open(sk_{\mathcal{I}}, reg_{\mathcal{I}}, m_j)$ and blocklists it using the accumulator.

Post-revocation. To win the revocability game, $\mathcal{A}$ must output $(q', V_j, m'_j, e'(t'))$ for an epoch $e'(t') > e(t)$ such that zone entry message opens to authentication ticket $q' \leftarrow Open(sk_{\mathcal{I}}, reg_{\mathcal{I}}, m'_j)$.

We now consider the two possible scenarios.

- There exists $(q', \mathcal{V}, e'(t')) \in \mathcal{L}_{auth}$: It shows $\mathcal{A}$ was able to successfully authorize using a fresh ticket q'. In this case, ZEEP revocability reduces to **TFRAC revocability**.
- There exists no tuple $(q', \mathcal{V}, e'(t')) \in \mathcal{L}_{auth}$: It shows that $\mathcal{A}$ was able to forge a DGSA credential and generate a zone-entry token for q' in epoch $e'(t')$ without authorization. Hence, ZEEP revocability reduces to the **DGSA traceability.**

Since the adversarial advantages in breaking TFRAC revocability and DGSA traceability are negligible, it follows that $\mathcal{A}$ also has a negligible advantage in winning the ZEEP revocability game.

□

C.6 Identity escrow freeness (IEF)

IEF ensures that no TTP (including the issuer) can determine the identity or ticket of a vehicle that generated a given authorization or entry transcript. In the IEF security game, the adversary is modeled as an honest-but-curious issuer $\mathcal{I}^*$. It honestly follows the protocol, i.e. vehicle *enrollment* and *authorization*, as well as *opening* the enter messages to reveal the corresponding tickets.

Given two honestly enrolled vehicles and an authorization or entry transcript generated by one of them, the adversary's goal is to identify which vehicle generated the transcript with a non-negligible advantage over random guessing. The IEF game for chosen bit b is formalized in Figure 7.

Experiment $\mathbf{Exp}^{IEF-b}_{Z,E,T}(\mathcal{I}^*)$

$pp \leftarrow Setup(1^\lambda, Z, E, T)$, $(sk_{\mathcal{I}}, pk_{\mathcal{I}}, reg_{\mathcal{I}}) \leftarrow KeyGen_{\mathcal{I}}(pp)$

Initialize all oracles $O(sk_{\mathcal{I}}, pk_{\mathcal{I}}, reg_{\mathcal{I}})$
$(\mathcal{V}_0, \mathcal{V}_1) \leftarrow \mathcal{I}^{*O}(\text{choose}, pp, sk_{\mathcal{I}}, pk_{\mathcal{I}})$

Enroll both the vehicles with $\mathcal{I}^*$
Use the challenge oracles $O^*(b)$ to interact with adversary $\mathcal{I}^*$ using vehicle $\mathcal{V}_b$
$b' \leftarrow \mathcal{I}^{*O^*}(\text{guess}, state_{\mathcal{I}^*})$
Return 1 if $b' = b$

Figure 7: Experiment for Identity Escrow Freeness

Definition C.6. *ZEEP satisfies identity escrow freeness if for any efficient adversary $\mathcal{I}^*$, zone-set Z, epoch set E and time-period set T,*

$$|Pr[Exp^{IEF-0}_{Z,E,T}(\mathcal{I}^*) = 1] - Pr[Exp^{IEF-1}_{Z,E,T}(\mathcal{I}^*) = 1]| \leq negl(\lambda).$$

Theorem C.6. *ZEEP satisfies identity escrow freeness if TFRAC is misauthentication resistant, and the ZKPoKs of TFRAC credential proof π_{auth} and DGSA token tok satisfy zero-knowledge.*

Proof. The challenger C runs the system setup and provides the public parameters pp and the issuer's key pair $(sk_{\mathcal{I}}, pk_{\mathcal{I}})$ to the adversary $\mathcal{I}^*$. In the challenge phase, $\mathcal{I}^*$ outputs two honest vehicle identities $\{\mathcal{V}_0, \mathcal{V}_1\}$ and sends them to C. The challenger C engages in the enrollment protocol with $\mathcal{I}^*$ on behalf of both the honest vehicles $\mathcal{V}_0$ and $\mathcal{V}_1$. For $b \in \{0, 1\}$, C sends the commitment $Comm_b$ of the vehicle's secret attributes and ticket queue containing one-time usable authentication ticket q_b and a ZKPoK $\pi_{enroll-b}$ to $\mathcal{I}^*$. C receives a TFRAC signature $\sigma_{TFRAC-b}$ from $\mathcal{I}^*$ and sets the corresponding TFRAC credential $cred_{TFRAC-b}$ for each vehicle.

Note: The **misauthentication-resistance of TFRAC** ensures only honestly enrolled vehicles can receive the TFRAC credential and participate in the ZEEP protocol.

The challenger C picks a random bit $b \in \{0, 1\}$ and interacts with $\mathcal{I}^*$ in the remaining protocols using vehicle $\mathcal{V}_b$ as follows.

- *Authorize:* C runs the authorization protocol for $\mathcal{V}_b$ in epoch e' using the ticket q_b and $cred_{TFRAC-b}$. It generates the TFRAC credential proof π_{auth-b} and sends it to $\mathcal{I}^*$. The adversary $\mathcal{I}^*$ updates the TFRAC signature to $\sigma'_{TFRAC-b}$, generates a DGSA signature σ_{DGSA} on (q_b, e'), and sends both back to C. Challenger C then updates the TFRAC credential to $cred'_{TFRAC-b}$ and DGSA credential to $cred_{DGSA-b}$.
- *Enter:* The challenger C uses q_b and $cred_{DGSA-b}$ to generate a token tok_b and executes the enter protocol for epoch e' with message m_b on behalf of the vehicle $\mathcal{V}_b$.
- *Open:* The adversary $\mathcal{I}^*$ may invoke the opening oracle to open the enter transcript corresponding to message m_b and obtain the authentication ticket q_b used to generate it.

Although $\mathcal{I}^*$ may request for transcripts of multiple rounds across different epochs, each round uses fresh one-time tickets and zero-knowledge proofs, making them unlinkable. Thus, indistinguishability within single epoch suffices for the IEF. We now construct a series of hybrid games to argue the IEF property of ZEEP for one epoch.

Game 0 (Real IEF Experiment): This is the original identity escrow freeness game where $\mathcal{I}^*$ attempts to guess b based on the observed transcripts during *Authorize*, *Enter*, and *Open*.

Game 1 (Simulate TFRAC credential proof): This game is similar to the previous game, except it replaces π_{auth} during the *Authorize* phase with a simulated transcript π^*_{auth} using a zero-knowledge simulator. *Game 1* is indistinguishable from *Game 0* due to the **zero-knowledge** property of π_{auth}.

Game 2 (Randomize the authentication ticket): This game is similar to the previous game, except it replaces q used during the *Authorize* protocol with a random ticket q^*. As the simulated proof π^*_{auth} in *Game 1* is independent of the actual secret, replacing q_b with q^* in *Game 2* still preserves the indistinguishability.

Game 3 (Simulate DGSA Token) : This game is similar to the previous game, except that during the *Enter* protocol, instead of actually generating a DGSA token tok_b, the challenger uses the zero-knowledge simulator to generate a simulated token tok^* with a random message m^*. *Game 3* is indistinguishable from *Game 2* due to the **zero-knowledge** property of the ZKPoK tok^*.

The adversary's view in *Game 3* consists of a random authentication ticket q^*, a simulated authorization proof π^*_{auth}, a simulated enter transcript with DGSA token and random message tok^*, m^*, and an output of the Open oracle which reveals q^* (already random and independent of b). Each of them is independent of the identity $\mathcal{V}_b$. Hence, $\mathcal{I}^*$ gains no advantage in guessing the bit b. By transitivity of indistinguishability, the adversary's advantage in Game 0 is also negligible. Since each authorization session involves a fresh random authentication ticket, IEF also ensures backward unlinkability across epochs.

□

References

[1] Jan Camenisch, Manu Drijvers, Anja Lehmann, Gregory Neven, and Patrick Towa. 2020. Zone encryption with anonymous authentication for V2V communication. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE.