Road-to-Vehicle Communications with Time-Dependent Anonymity: A Light Weight Construction and its Experimental Results[*]

Keita Emura[†] Takuya Hayashi[‡]

July 24, 2017

Abstract

This paper describes techniques that enable vehicles to collect local information (such as road conditions and traffic information) and report it via road-to-vehicle communications. To exclude malicious data, the collected information is signed by each vehicle. In this communications system, the location privacy of vehicles must be maintained. However, simultaneously linkable information (such as travel routes) is also important. That is, no such linkable information can be collected when full anonymity is guaranteed using cryptographic tools such as group signatures. Similarly, continuous linkability (via pseudonyms, for example) may also cause problem from the viewpoint of privacy.

In this paper, we propose a road-to-vehicle communication system with relaxed anonymity via group signatures with time-token dependent linking (GS-TDL). Briefly, a vehicle is unlinkable unless it generates multiple signatures in the same time period. We provide our experimental results (using the RELIC library on a cheap and constrained computational power device, Raspberry Pi), and simulate our system by using a traffic simulator (PTV), a radio wave propagation analysis tool (RapLab), and a network simulator (QualNet). Though a similar functionality of time-token dependent linking was proposed by Wu, Domingo-Ferrer and González-Nicolás (IEEE T. Vehicular Technology 2010), we can show an attack against the scheme where anyone can forge a valid group signature without using a secret key. In contrast, our GS-TDL scheme is provably secure.

In addition to the time-dependent linking property, our GS-TDL scheme supports verifier-local revocation (VLR), where a signer (vehicle) is not involved in the revocation procedure. It is particularly worth noting that no secret key or certificate of a signer (vehicle) must be updated whereas the security credential management system (SCMS) must update certificates frequently for vehicle privacy. Moreover, our technique maintains constant signing and verification costs by using the linkable part of signatures. This might be of independent interest.

1 Introduction

Location privacy is widely recognized as an important issue, especially in the case of motor vehicles. For example, Rouf et al. [48] mentioned that location privacy could be compromised via a tire pressure monitoring system, and Xu et al. [56] proposed a secure communication protocol as a countermeasure against this vulnerability. However, simultaneously real-time local information is important and useful, e.g., information about traffic and road conditions is indispensable for maintaining urban operations. Therefore, collecting local information without infringing on privacy is an important issue that requires a resolution.

Let us consider a situation in which vehicles collect such local information and report it via road-to-vehicle communications. To exclude malicious data, this collected information must be signed by each

[*]An extended abstract appears in the 4th International Workshop on Lightweight Cryptography for Security and Privacy (LightSec 2015) [27]. This is the full version. In the conference version, we provided the GS-TDL part, and used the TEPLA library [4] for estimating the efficiency of our GD-TDL scheme. In this full version, we use the RELIC library and reconsider the pairing equations in our algorithm. Moreover, we simulate our system using PTV, RapLab, and QualNet.

[†]National Institute of Information and Communications Technology (NICT), 4-2-1, Nukui-kitamachi, Koganei, Tokyo, 184-8795, Japan. `k-emura@nict.go.jp`

[‡]Kobe University, 1-1, Rokko-Dai, Nada-ku, Kobe, Hyogo, 657-8501, Japan. `t-hayashi@eedept.kobe-u.ac.jp`

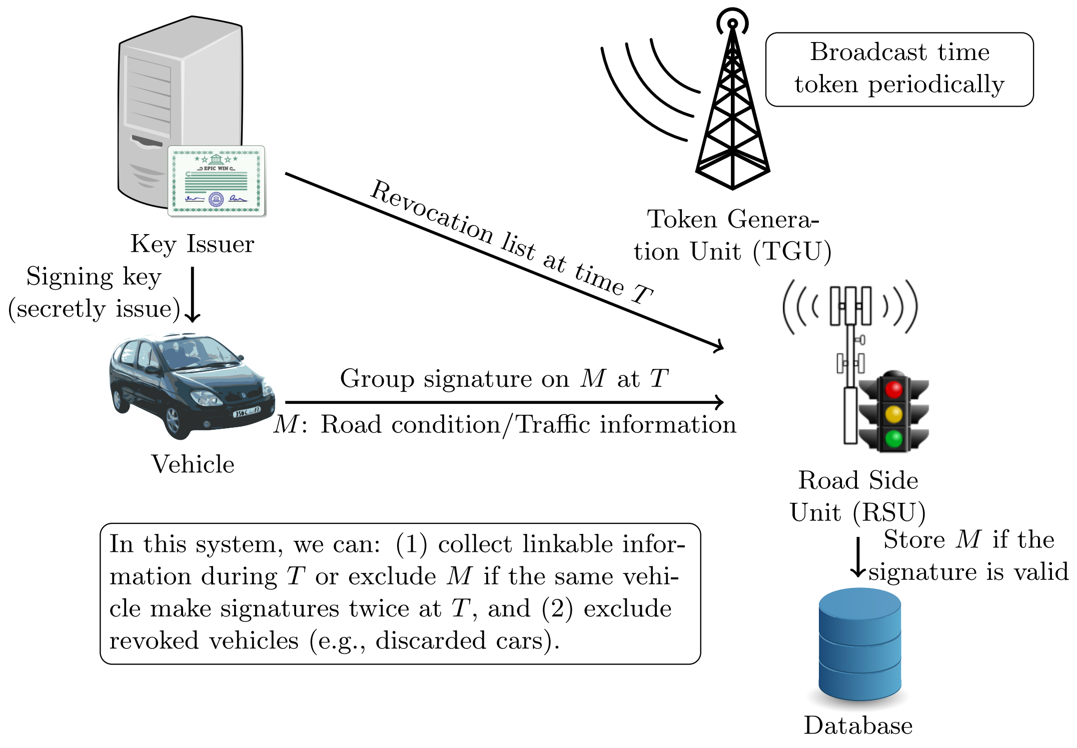
Broadcast time token periodically
Key Issuer
Revocation list at time T
Token Generation Unit (TGU)
Signing key (secretly issue)
Group signature on M at T
M: Road condition/Traffic information
Vehicle
Road Side Unit (RSU)
Store M if the signature is valid
In this system, we can: (1) collect linkable information during T or exclude M if the same vehicle make signatures twice at T, and (2) exclude revoked vehicles (e.g., discarded cars).
Database

Figure 1: Brief Description of Our Road-to-Vehicle Communication System Based on GS-TDL

vehicle. In this case, it seems undesirable, from the viewpoint of privacy, to link location information (obtained from collected information and signatures) to an individual. As a countermeasure, Whyte et al. provided the security credential management system (SCMS) [53], in which a pseudonym certificate authority issues certificates to vehicles frequently to prevent linking vehicles. However, because vehicles must to update their certificates frequently, it is difficult to manage securely in practice. Applying group signatures [21] could be a solution where a verifier can anonymously verify a signer (vehicle), about which there have been numerous studies, e.g., [22, 54, 42, 49, 55]. In particular, Wu, Domingo-Ferrer and González-Nicolás [54] proposed an anonymous threshold authentication scheme, in which messages are accepted when more than t vehicles send the same message (signed by the vehicles). The core cryptographic technique in their system utilizes a message-linkable group signature (MLGS), where two group signatures become linkable if a signer generates a group signature for the same message twice. Thus, we can protect the case in which a vehicle sends two signatures of the same massage even in an anonymous environment.

Our Target: In road-to-vehicle systems, collecting linkable information (such as travel routes) is important. However, if cryptographic tools with full anonymity (or more precisely, unlinkability), such as group signatures, are simply employed, no such linkable information can be collected. Conversely, continuous linkability (via pseudonyms, for example) is problematic from the viewpoint of privacy. For example, a vehicle with continuous linkability is tracked from the time the vehicle is acquired up until the time it is sold, and parking spaces, which may include the driver's home and work place, are revealed. Therefore, suitably defining "moderate" anonymity with practical efficiency in a road-to-vehicle communications context is an important issue that must be resolved.

One may think that the Wu et al. MLGS scheme [54] could be applicable for constructing a privacy-preserving system, however, we note that no formal security definition for MLGS is provided in [54] and the security proofs are informal. In fact, we can show an attack against the MLGS scheme of Wu et al., where anyone can forge a valid group signature without using a secret key (see Section 3). Moreover, we must consider revocation to exclude discarded cars, cars with malicious behaviors, and so on.

Our Contribution: In this paper, we propose a road-to-vehicle communication system with relaxed anonymity by considering time-dependent linking properties, where a vehicle is unlinkable unless it generates multiple signatures in the same time period T. We assume that a token generation unit (TGU) generates a time-dependent token t_T at a time T, and broadcasts t_T. Each signer (vehicle) with a unique identity ID, generates a signature σ on a message M (local information) by using its own signing key $\mathsf{sigk}_{\mathsf{ID}}$ and t_T. A verifier (we assume a road side unit (RSU) in the road-to-vehicle communications context) checks whether σ is a valid signature. As a remark, no secure channel is required for issuing a token t_T e.g., via GPS systems or more simply via RSU in the road-to-vehicle communications context. We give a brief description of our system in Fig 1. Time-dependent linking appears to be more suitable for our system than message-dependent linking [54] where a vehicle is always linkable if it generates group signatures on the same message twice, and this situation might occur when a vehicle is used for work trips and uses the same road each day.[1]

Our system's core cryptographic tools utilize group signatures with time-token dependent linking (GS-TDL), which we propose in this paper. In GS-TDL, a signer is unlinkable unless it generates multiple signatures in the same time period T. If T is set as short as possible, then GS-TDL comes close to usual group signatures i.e., a signer is always unlinkable whereas if T is set as long as possible, then GS-TDL comes close to usual digital signatures i.e., a signer is always linkable. That is, GS-TDL can control how long linkable information can be corrected by setting T (in our network simulation (Section 5.2), we set T as 600 seconds). Our GS-TDL supports verifier-local revocation (VLR) [46, 45, 17], where no signer is involved in the revocation procedure. In particular, our GS-TDL achieves backward unlinkability [45, 46, 40], which prevents adversaries from breaking anonymity, even after vehicles are revoked. It is particularly worth noting that no secret key or certificate of a signer (vehicle) must be updated whereas SCMS [53] must update certificates frequently for vehicle privacy. Moreover, our time-dependent linking properties enable us to achieve *constant verification costs*, whereas those of previous schemes are $O(r)$, where r is the number of revoked users. This is apparently related to independent interest.

We also provide the experimental results of our road-to-vehicle communication system, and show that our system is feasible in practice. To implement GS-TDL, we use the RELIC library [7] [2]. We also simulate our system by using a traffic simulator (PTV) [3], a radio wave propagation analysis tool (RapLab) [4], and a network simulator (QualNet) [5].

Our GS-TDL is secure in the random oracle model under the q-strong Diffie-Hellman (q-SDH) assumption [15] and the strong decisional Diffie-Hellman inversion (SDDHI) assumption [20, 30]. The group signature size in our scheme is shorter than that of previous schemes owing to time-dependent linkability. Specifically, a signature contains only 6 group elements, whereas that of the short group signature scheme [16] contains 9 group elements, that of the short controllable linkable group signature scheme [35] contains 8 group elements, and that of the controllable linkable group signature scheme (for dynamic group setting) [36] contains 12 group elements. Recently, though a short dynamic controllable linkable group signature scheme is proposed [34], a signature contains 8 group elements.[6] In addition to the signature size, our linking algorithm does not require cryptographic computations. The algorithm simply compares whether two elements are the same.

Related Work: Because car security has been recognized as a real threat, several organizations have been launched, e.g., Preserve (EU) [3], ITS Info-communications Forum (Japan) [2], and IntelliDrive (USA) [1], and car security is researched in several papers e.g., [52, 19, 51, 43].

[1]Even if a random nonce is included as a part of signed message, no linking algorithm works and this leads to a wag-the-dog situation. Even if a time T is included, e.g., sign $M||T$ by using a MLGS scheme, anyone can manipulate T and such a signer-driven anonymous system must be avoided because vehicles have incentive to hide identity. On the contrary, in GS-TDL, time T is authorized by TGU and no vehicle can manipulate T.

[2]We used the TEPLA library [4] in the previous paper [27]. We reconsider the pairing equations in our algorithm according to the RELIC library.

[3]http://vision-traffic.ptvgroup.com/

[4]http://www.kke.co.jp/en/solution/theme/raplab.html

[5]http://web.scalable-networks.com/content/qualnet

[6]We remark that these schemes [16, 34, 35, 36] also achieve only CPA-anonymity (i.e., no opening oracle access is allowed in anonymity game) as in ours.

Nakanishi et al. proposed a linkable group signature [44], where anyone can determine whether two signatures were made by the same signer. In contrast to GS-TDL, no time-dependent token is required for linking. That is, two group signatures made by the same signer are always linkable. A group signature with a relaxed anonymity for VANET has been considered in [42]. However the link algorithm is not publicly executable, and an authority called Link Manager is introduced. That is, two group signatures made by the same signer are always linkable from the viewpoint of Link Manager. Moreover, pairing computations are required for linking. Abe et al. [5] proposed double-trapdoor anonymous tags which can generally construct traceable signatures [38]. Because a signer is always linkable after the corresponding token is broadcasted, we cannot use traceable signatures instead of GS-TDL. As a special case of traceable signatures, group signatures with controllable linkability have been proposed [14, 36, 35], where a link key is defined for the linking procedure. However, pairing computations are required for linking, which lead to inefficiency.

Kumar et al. proposed a group signature scheme with probabilistic revocation [39]. In this scheme, a token (which they call an alias token) is contained in a group signature, and the same token is used when group signatures are generated in the same time period, i.e., these are linkable as in our GS-TDL. Though their scheme could be a candidate for constructing an anonymous time-dependent authentication system, there are many problems as follows. First, their security model does not consider revocation. That is, there is a possibility that, for example, anonymity might be broken if some other users are revoked. Moreover, their security model also does not consider time-dependent linking.

Liu et al. [41] proposed an anonymous authentication roaming protocol supporting time-bound credentials, where a user secret key is bounded to an expiry time. Though this time-bound property might be applicable to time-dependent linking, their protocol has a disadvantage where the size of the common group public key, which is necessary for generating signatures, depends on the bit-length for representing a time period. In the road-to-vehicle communication context, each vehicle is required to have such a long-size pubic key. In the meantime, the size of the group pubic key in our scheme is constant.

Chow, Susilo, and Yuen [24] proposed two linkable ring signature schemes. The first one is an identity-based linkable ring signature scheme where anyone can check whether two signatures are made by the same signer (i.e., providing public verifiability). This functionality could be employed in our usage. However, one drawback is the size of signatures. That is, a signature contains 12 group elements (and an event identity). In contrast, our scheme provides a shorter signature that contains 6 group elements. The second one is a ring signature scheme with escrowed linkability where a linking authority can link two signatures by using a secret key. As in group signatures with controllable linkability [14, 36, 35], pairing computations are required for linking.

Chow [23] proposed an efficient tracing mechanism where the user-specific tracing trapdoor allows the reconstruction of tags. In the Chow traceable signature scheme, the tracing authority computes N tags for each signer. A signer sets a counter between 0 and N, where N is the number of unlinkable signatures, and proves that the counter is in $[0, N]$ via a range proof technique. If the same counter is used for generating two or more signatures, then these signatures are traced because the same tag (specified to a signer and a counter value) is contained in them. This functionality could be employed in our usage. If we assume that each tag is computed for each time period (i.e., a counter value is set as T), then we could achieve the time-dependent linking property. Note that the range proof can be removed in our usage since the time period is a public value. The signature size then does not depend on N. Thus, our GS-TDL scheme can be regarded as a special case of the Chow scheme. Note that the Chow scheme provides additional functionalities (opening and claiming). Thus, in our proposal, we adequately customize the Chow scheme to fit our usage i.e., removing range proofs, opening, and claiming.

Remark: Because we mainly focus on how to control anonymity in the group signature context, we do not discuss how to preserve location privacy revealed from messages to be signed. For example, anonymity is never guaranteed if a message M to be signed contains the identifier of a vehicle itself. Even if we avoid such an extreme case, some personal information might be infringed from M. Here, we must assume that no personal information is revealed from a content M (or even its statistical values).

Though we may apply privacy preserving techniques for modifying contents or its statistical values, (e.g., [50, 26]), this is beyond the scope of this paper.

2 Preliminaries: Cryptographic Definitions

In this section, we give the definitions of bilinear groups, complexity assumptions, and digital signature as follows.

Complexity Assumptions: Let $\mathcal{G}$ be a probabilistic polynomial-time algorithm that takes a security parameter λ as input and generates a parameter $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ of bilinear groups, where p is a λ-bit prime, $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are groups of order p, e is a bilinear map from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$, and g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Here, for any $a, b \in \mathbb{Z}_p$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ holds. We use the asymmetric setting, i.e., $\mathbb{G}_1 \neq \mathbb{G}_2$, and assume that no efficient isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ is known.

Definition 2.1 (SDDHI Assumption [20]). *We say that the SDDHI (Strong Decisional Diffie-Hellman Inversion) assumption holds if for all PPT adversaries $\mathcal{A}$, $|\Pr[(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \xleftarrow{\$} \mathcal{G}(1^\lambda);\ x \xleftarrow{\$} \mathbb{Z}_p;\ (T, st) \leftarrow \mathcal{A}^{\mathcal{O}_x}(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, g_1^x);\ \tau_0 = g_1^{\frac{1}{x+T}};\ \tau_1 \xleftarrow{\$} \mathbb{G}_1;\ b \xleftarrow{\$} \{0,1\};\ b' \leftarrow A^{\mathcal{O}_x}(\tau_b, st) :\ b = b'] - \frac{1}{2}|$ is negligible, where $\mathcal{O}_x$ is an oracle which takes as input $z \in \mathbb{Z}_p^* \setminus \{T\}$, outputs $g_1^{\frac{1}{x+z}}$.*

We remark that the underlying bilinear group must not be symmetric.

Definition 2.2 (q-SDH Assumption [15]). *We say that the q-SDH (q-Strong Diffie-Hellman) assumption holds if for all PPT adversaries $\mathcal{A}$, $\Pr[(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \xleftarrow{\$} \mathcal{G}(1^\lambda);\ \gamma \xleftarrow{\$} \mathbb{Z}_p;\ (x, g_1^{\frac{1}{x+\gamma}}) \leftarrow \mathcal{A}(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_1^\gamma, \ldots, g_1^{\gamma^q}, g_2, g_2^\gamma);\ x \in \mathbb{Z}_p^* \setminus \{-\gamma\}]$ is negligible.*

Digital Signature: Let $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be a digital signature scheme. The key generation algorithm Gen takes as input a security parameter λ, and outputs a pair of verification/signing key $(\mathsf{vk}, \mathsf{sigk})$. The signing algorithm Sign takes as input sigk and a message to be signed M, and outputs a signature Σ. The verification algorithm Verify takes as input vk, Σ and M, and outputs $0/1$. We require the following correctness property: for all $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and M, $\Pr[\mathsf{Verify}(\mathsf{vk}, \mathsf{Sign}(\mathsf{sigk}, M), M) = 1] = 1$ holds. Next, we define existential unforgeability against chosen message attack (EUF-CMA) as follows. Let $\mathcal{C}$ be the challenger, and $\mathcal{A}$ be an adversary. $\mathcal{C}$ runs $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and gives vk to $\mathcal{A}$. $\mathcal{A}$ is allowed to issue signing queries M. $\mathcal{C}$ runs $\Sigma \leftarrow \mathsf{Sign}(\mathsf{sigk}, M)$ and returns Σ to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs (Σ^*, M^*). We say that a digital signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is EUF-CMA if the probability, that $\mathsf{Verify}(\mathsf{vk}, \Sigma^*, M^*) = 1$ and $\mathcal{A}$ did not send M^* as a signing query, is negligible.

3 The WDG Construction and its Vulnerability

In IEEE T. Vehicular Technology 2010, Wu, Domingo-Ferrer and González-Nicolás (WDG) [54] proposed message-linkable group signature (MLGS), where if a signer generates a group signature for the same message twice, then two group signatures become linkable. From MLGS, they constructed an anonymous threshold authentication for vehicle-to-vehicle communications, where if more than t vehicles send the same message (signed by vehicles) then this information is accepted. In their system, four entities are defined: a vehicle $\mathcal{V}$, the vehicle manufacturer $\mathcal{VM}$, the registration manager $\mathcal{RM}$, and the group tracing manager $\mathcal{TM}$. In this section, we give an attack against their MLGS scheme denoted by the WDG scheme. Briefly, we show that anyone can forge a valid group signature without knowing a signing key.

The WDG scheme is described as follows: Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, h_1, h_2, U_1, U_2, H_1, H_2)$ be public parameters, where $H_1 : \{0,1\}^* \to \mathbb{G}_1$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_p$ are hash functions, and for a computable isomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$, $g_1 = \phi(g_2)$, $h_1 = \phi(h_2)$, and $U_1 = \phi(U_2)$ hold. A vehicle $\mathcal{V}$ has a secret key $y \in \mathbb{Z}_p$ and a public key $Y = U_1^y$, and Y is signed by the vehicle manufacturer $\mathcal{VM}$. In the vehicle registration phase, $\mathcal{V}$ computes $T = g_2^y$ and sends (T, Y) to the group tracing manager $\mathcal{TM}$. $\mathcal{TM}$ checks the signature of Y and whether $e(Y, g_2) = e(U_1, T)$ holds or not. Then, $\mathcal{TM}$ saves (T, Y) into a local database. Moreover, $\mathcal{V}$ sends Y (and its signature) to the registration manager $\mathcal{RM}$, and runs a zero-knowledge protocol for $y = \log_{U_1} Y$. Then, $\mathcal{RM}$ chooses $k \leftarrow \mathbb{Z}_p$ and computes $K_1 = g_1^k$ and $K_2 = Z(h_1Y)^{-k}$, where Z (and $A = e(Z, g_2)$) is a public key of $\mathcal{RM}$. $\mathcal{V}$ obtains $K_v = (K_1, K_2)$ as a secret signing key. In a signing phase for a message M, $\mathcal{V}$ selects $s, r_y \leftarrow \mathbb{Z}_p$ and computes $\sigma_1 = K_1 g_1^s$, $\sigma_2 = K_2(h_1Y)^{-s}$, $\sigma_3 = \sigma_1^y$, $\sigma_4 = H_1(M)^y$, $\sigma_5 = H_2(M||\sigma_1||\sigma_2||\sigma_3||\sigma_4||H_1(M)^{r_y}||\sigma_1^{r_y})$, and $\sigma_6 = r_y - \sigma_5 y$, and outputs a group signature $\sigma = (\sigma_1, \dots, \sigma_6)$. That is, this proves that the discrete logarithm of σ_3 and that of σ_4 (i.e., y) are the same. In the verification phase, check $e(\sigma_2, g_2)e(\sigma_1, h_2)e(\sigma_3, U_2) = A$ and $\sigma_5 = H_2(M||\sigma_1||\sigma_2||\sigma_3||\sigma_4||H_1(m)^{\sigma_6}\sigma_4^{\sigma_5}||\sigma_1^{\sigma_6}\sigma_3^{\sigma_5})$.

The problem of this construction is that no $\mathcal{RM}$'s verification key is involved in the signing and verification phases. Before showing our attack, we explain a typical methodology for constructing group signature introduced in [11, 12] as follows: a key issuer ($\mathcal{RM}$ in the MLGS context) has a verification/signing key pair (vk, sk) of a signature scheme, and an opener ($\mathcal{TM}$ in the MLGS context) has a public/secret key pair of an encryption scheme. The key issuer then generates a signature for a user ($\mathcal{V}$ in the MLGS context) as a signing key (say *cert*). In the signing phase, a user encrypts *cert* by using the public key of the opener, and computes a non-interactive zero-knowledge (NIZK) proof that proves "encrypted *cert* is a valid signature under vk".

Since the WDG scheme did not include such a vk, anyone can forge a valid group signature. The concrete attack is described as follows: let $\mathcal{A}$ be an adversary. $\mathcal{A}$ chooses $y, k, s, r_y \leftarrow \mathbb{Z}_p$ and computes $Y = U_1^y$, $\sigma_1 = g_1^k g_1^s$, $\sigma_2 = Z(h_1Y)^{-k}(h_1Y)^{-s}$, $\sigma_3 = \sigma_1^y$, $\sigma_4 = H_1(M)^y$, $\sigma_5 = H_2(M||\sigma_1|| \sigma_2||\sigma_3||\sigma_4||H_1(M)^{r_y}||\sigma_1^{r_y})$, and $\sigma_6 = r_y - \sigma_5 y$, and outputs a group signature $\sigma_{\text{forge}} = (\sigma_1, \dots, \sigma_6)$. Then, σ_{forge} is a valid group signature though $\mathcal{A}$ does not have a signing key. Moreover, since $\mathcal{A}$ does not register Y, no $\mathcal{TM}$ can trace the signer of σ. This breaks traceability.

4 GS-TDL: Syntax and Our Construction

4.1 Syntax of GS-TDL

Before providing the syntax, we discuss whether the open functionality should be utilized in our usage. In the open functionality, an authority (called an opener) can determine the identity of the actual signer by using a secret opening key. For example, the open functionality is implemented by using public key encryption (PKE) or non-interactive zero-knowledge proof of knowledge, and could be an efficiency bottleneck. It has been reported that the signature size of the Furukawa-Imai group signature scheme [31] can be reduced by 50% if the open functionality is removed [28]. It also has been reported that implementing the open functionality without using PKE leads to a short group signature scheme at the expense of the signature opening costs [13, 25, 47]. Owing to the above facts, we do not consider the open functionality because we pursued a light-weight implementation of the system, and we consider only the linking functionality.

We note that MLGS [54] is essentially the same as the unique group signature proposed by Franklin and Zhang [30], and formal security definitions are given in [30]. That is, we may be able to apply a unique group signature to construct a road-to-vehicle communication system. However, our time-dependent linking seems suitable for the system rather than message-dependent linking as explained before. Moreover, the Franklin-Zhang model supports the open algorithm and considers a stronger anonymity (so called CCA anonymity). Since we customize the syntax to be suitable for light-weight realization and exclude the open algorithm, in this paper we do not directly apply the Franklin-Zhang unique group signature scheme to our system.

Moreover, we assume that the signing key of a signer is embedded in a device during the setup phase, so we remove an interactive join algorithm from our syntax. Finally, we consider the revocation functionality, especially verifier-local revocation (VLR) where no signer is involved in revocation procedures. That is, we introduce revocation lists RL that contain revocation tokens of revoked signers. The signing algorithm does not take RL as an input, whereas the verification algorithm takes RL as the input. For time-token dependent linking, we introduce the token key generation algorithm and the token generation algorithm, and the signing algorithm takes a time-dependent token as the input. Note that the revocation check can be implicitly used for tracing signers by generating revocation tokens for all signers and checking the revocation equation. This methodology is essentially the same as that of the open algorithm of Bichsel et al. [13] and its follow up works [25, 47].

Definition 4.1 (Syntax of GS-TDL)**.** *A group signature scheme with time-token dependent linking* $\mathcal{GS}\text{-}\mathcal{TDL}$ *consists of the algorithms* ($\mathsf{Setup}, \mathsf{GKeyGen}, \mathsf{TKeyGen}, \mathsf{Join}, \mathsf{TokenGen}, \mathsf{GSign}, \mathsf{Revoke}, \mathsf{GVerify}, \mathsf{Link}$) *as follows:*

- Setup*: The setup algorithm takes as input a security parameter* λ*, and outputs a public parameter params.*
- $\mathsf{GKeyGen}$*: The group key generation algorithm takes as input params, and outputs a group public key* gpk*, a group master key* gsk*, an initial revocation storage* $\mathsf{grs} := \emptyset$ *and an initial revocation list* $\mathsf{RL}_0 := \emptyset$*.*
- $\mathsf{TKeyGen}$*: The token key generation algorithm takes as input params, and outputs a public key* tpk *and a secret key* tsk*.*
- Join*: The join algorithm takes as input* gsk*,* grs *and a unique identity* ID*, and outputs a signing key* $\mathsf{sigk}_{\mathsf{ID}}$ *and updated revocation storage. We remark that this algorithm is not required to be interactive.*
- $\mathsf{TokenGen}$*: The token generation algorithm takes as inputs* tsk *and a time* T*, and outputs a token* t_T*.*
- GSign*: The signing algorithm takes as inputs* gpk*,* tpk*,* t_T*,* $\mathsf{sigk}_{\mathsf{ID}}$*, and a message* M *to be signed, and outputs a group signature* σ*.*
- Revoke*: The revocation algorithm takes as inputs* gpk*,* grs*, and a set of revoked users at a time* T $\{\mathsf{ID}_{T,1}, \dots, \mathsf{ID}_{T,n_T}\}$*, and outputs* RL_T*. Here,* n_T *is the number of revoked users on* T*.*
- $\mathsf{GVerify}$*: The verification algorithm takes as inputs* gpk*,* tpk*,* RL_T*,* σ*, and* M*, and outputs* 0 (invalid) *or* 1 (valid)*.*
- Link*: The linking algorithm takes as inputs* gpk*,* tpk*, and* RL_T*, and two signatures and messages* (σ_0, M_0, T_0) *and* (σ_1, M_1, T_1)*, and outputs* 1 *if two signatures are made by the same signer, and* 0 *otherwise. We remark that the* Link *algorithm outputs* 0 *does not guarantee two signatures are made by the different signers. For example, if a signature is invalid, then the algorithm outputs* 0*.*

4.2 Security Definitions of GS-TDL

In this section, we give security definitions of GS-TDL by adding the time-dependent linkability to the Bellare-Micciancio-Warinschi (BMW) model [11] (which is recognized as a de-facto standard for group signature).[7]

We require the following correctness, where any honestly generated signatures are valid, and the Link algorithm correctly links two signatures if these are generated by the same signing key with the same token, unless the corresponding signer is not revoked. Moreover, we require that a signature is invalid if the corresponding signer is revoked.[8]

[7]Recently, Bootle et al. [18] consider fully dynamic group signatures where users can join and leave at any time. They point out that previous definitions are weak in the sense that these definitions allow members who joined at recent time periods to sign messages w.r.t earlier time periods during which they were not members of the group. Though this situation may be a problem in some applications (e.g., sign a document), however, in our usage, it seems easy to ignore such signatures by a verifier (Road Side Unit) if signatures sent from vehicles are generated in a past time period.

[8]As a remark, the case that an adversary generates a valid signature using a revoked user's signing key cannot be captured by unforgeability since the open algorithm is not defined. Instead, we consider the case that a signature is invalid when the corresponding signer is revoked in correctness, though it might be additionally defined such as revocation soundness.

Definition 4.2 (Correctness). *For any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ and the security parameter $\lambda \in \mathbb{N}$, we define the experiment $\mathrm{Exp}^{corr}_{\mathsf{GS\text{-}TDL},\mathcal{A}}(\lambda)$ as follows.*

```
$\mathrm{Exp}^{corr}_{\mathsf{GS\text{-}TDL},\mathcal{A}}(\lambda)$ :
    $params \leftarrow \mathsf{Setup}(1^\lambda)$
    $(\mathsf{gpk}, \mathsf{gsk}, \mathsf{grs}, \mathsf{RL}_0) \leftarrow \mathsf{GKeyGen}(params)$
    $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow \mathsf{TKeyGen}(params);\ \mathsf{GU} := \emptyset$
    $(\mathsf{ID}^*, T^*, M_0, M_1) \leftarrow \mathcal{A}^{\mathsf{AddU}(\cdot),\mathsf{Revoke}(\mathsf{grs},\cdot)}(\mathsf{gpk}, \mathsf{tpk})$
    $\mathsf{ID}^* \in \mathsf{GU};\ t_{T^*} \leftarrow \mathsf{TokenGen}(\mathsf{tsk}, T^*)$
    $\sigma_0 \leftarrow \mathsf{GSign}(\mathsf{gpk}, \mathsf{tpk}, t_{T^*}, \mathsf{sigk}_{\mathsf{ID}^*}, M_0)$
    $\sigma_1 \leftarrow \mathsf{GSign}(\mathsf{gpk}, \mathsf{tpk}, t_{T^*}, \mathsf{sigk}_{\mathsf{ID}^*}, M_1)$
    Return 1 if the following holds :
      $\big[\mathsf{ID}^* \notin \mathsf{RL}_{T^*} \wedge \big((\mathsf{GVerify}(\mathsf{gpk}, \mathsf{tpk}, \mathsf{RL}_{T^*}, M_0, \sigma_0) = 0$
       $\vee\ \mathsf{GVerify}(\mathsf{gpk}, \mathsf{tpk}, \mathsf{RL}_{T^*}, M_1, \sigma_1) = 0)$
       $\vee\ \mathsf{Link}(\mathsf{gpk}, \mathsf{tpk}, \mathsf{RL}_{T^*}, (M_0, \sigma_0, T^*), (M_1, \sigma_1, T^*)) = 0\big)\big]$
     $\vee\ \big[\mathsf{ID}^* \in \mathsf{RL}_{T^*} \wedge \big((\mathsf{GVerify}(\mathsf{gpk}, \mathsf{tpk}, \mathsf{RL}_{T^*}, M_0, \sigma_0) = 1$
       $\vee\ \mathsf{GVerify}(\mathsf{gpk}, \mathsf{tpk}, \mathsf{RL}_{T^*}, M_1, \sigma_1) = 1\big)\big]$
    Otherwise return 0
```

- AddU*: The add user oracle allows an adversary $\mathcal{A}$ to add honest users to the group. On input an identity* ID*, this oracle runs* $\mathsf{sigk}_{\mathsf{ID}} \leftarrow \mathsf{Join}(, \mathsf{gsk}, \mathsf{grs}, \mathsf{ID})$. ID *is added to* GU.
- Revoke*: Let $T-1$ be the time that the oracle is called. The revocation oracle allows an adversary $\mathcal{A}$ to revoke honest users. On input identities $\{\mathsf{ID}_{T,1}, \ldots, \mathsf{ID}_{T,n_T}\}$, this oracle runs* $\mathsf{RL}_T \leftarrow \mathsf{Revoke}(\mathsf{gpk}, \mathsf{grs}, \{\mathsf{ID}_{T,1}, \ldots, \mathsf{ID}_{T,n_T}\})$. *We remark that T^* is the challenge time that $\mathcal{A}$ outputs* $(\mathsf{ID}^*, M_0, M_1)$.

We say that $\mathcal{GS}$-$\mathcal{TDL}$ is correct if the advantage $\mathrm{Adv}^{corr}_{\mathsf{GS},\mathcal{A}}(\lambda) := \Pr[\mathrm{Exp}^{corr}_{\mathsf{GS\text{-}TDL},\mathcal{A}}(\lambda) = 1]$ is negligible for any PPT adversary $\mathcal{A}$.

Next, we give our anonymity definition which guarantees that no adversary who has tsk can distinguish whether two signatures are generated by the same signer or not, if the corresponding linkable signatures are not generated. In contrast to the BMW model, $\mathcal{A}$ is not allowed to obtain signing keys of challenge users (selfless anonymity). This is a reasonable setting since $\mathcal{A}$ can trivially break anonymity if $\mathcal{A}$ obtains such signing keys. For example, let $\mathcal{A}$ have $\mathsf{sigk}_{\mathsf{ID}_{i_0}}$. Then, $\mathcal{A}$ can make a signature σ on T_0 using $\mathsf{sigk}_{\mathsf{ID}_{i_0}}$ (with arbitrary message M), and can check whether $\mathsf{Link}(\mathsf{gpk}, \mathsf{tpk}, \mathsf{RL}_{T_0}, (M_0, \sigma^*, T_0), (M, \sigma, T_0)) = 1$ or not, where σ^* is the challenge signature. Instead, $\mathcal{A}$ is allowed to access the GSign oracle in our definition. Moreover, we consider backward unlinkability, where no adversary can break anonymity even after the challenge signers are revoked.

Definition 4.3 (Anonymity). *For any PPT adversary $\mathcal{A}$ and a security parameter $\lambda \in \mathbb{N}$, we define the experiment $\mathrm{Exp}^{anon\text{-}tg\text{-}b}_{\mathsf{GS\text{-}TDL},\mathcal{A}}(\lambda)$ as follows.*

```
$\mathrm{Exp}^{anon\text{-}tg\text{-}b}_{\mathsf{GS\text{-}TDL},\mathcal{A}}(\lambda)$ :
    $params \leftarrow \mathsf{Setup}(1^\lambda)$
    $(\mathsf{gpk}, \mathsf{gsk}, \mathsf{grs}, \mathsf{RL}_0) \leftarrow \mathsf{GKeyGen}(params)$;
    $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow \mathsf{TKeyGen}(params);\ \mathsf{GU} := \emptyset;\ \mathsf{STSet} := \emptyset$
    $d \leftarrow \mathcal{A}^{\mathsf{AddU}(\cdot),\mathsf{Revoke}(\mathsf{grs},\cdot),\mathsf{GSign}(\cdot,\cdot,\cdot),\mathsf{Ch}(b,\cdot,\cdot,\cdot,\cdot)}(\mathsf{gpk}, \mathsf{tpk}, \mathsf{tsk})$
    Return $d$
```

- AddU*: The same as before.*
- Revoke*: The same as before. We remark that if* $T_0 \neq T_1$ *and assume that* $T_0 < T_1$*, then* ID_{i_0} *and/or* ID_{i_1} *can be revoked after* T_1*. If* $T_0 = T_1$*, then* ID_{i_0} *and/or* ID_{i_1} *can be revoked after* T_0*.*
- GSign*: The signing oracle takes as input* ID*,* t_T*, and a message* M*. We assume that* t_T *is a valid token which means that the* $\mathsf{GVerify}$ *algorithm outputs 1 for all honestly generated signatures with* t_T*, even though this is made by* $\mathcal{A}$*. If* $\mathsf{ID} \notin \mathsf{GU}$*, then the oracle runs* $\mathsf{AddU}(\mathsf{ID})$*. The oracle returns* $\sigma \leftarrow \mathsf{GSign}(\mathsf{gpk}, \mathsf{tpk}, t_T, \mathsf{sigk}_{\mathsf{ID}}, M)$ *and stores* (ID, T) *in* STSet*.*
- Ch*: The challenge oracle takes as input* ID_{i_0}*,* ID_{i_1}*,* t_{T_0}*,* t_{T_1}*,* M_0^**, and* M_1^* *where* $\mathsf{ID}_{i_0} \neq \mathsf{ID}_{i_1}$ *and* $\mathsf{ID}_{i_0}, \mathsf{ID}_{i_1} \in \mathsf{GU}$*. Return signature(s) according to the following cases:*
  - $T_0 = T_1$*: If* $(\mathsf{ID}_{i_0}, T_0), (\mathsf{ID}_{i_1}, T_1) \notin \mathsf{STSet}$*, then compute* $\sigma^* \leftarrow \mathsf{GSign}(\mathsf{gpk}, \mathsf{tpk}, t_{T_b}, \mathsf{sigk}_{\mathsf{ID}_{i_b}}, M^*)$*, and return* σ^**. Without loss of generality, we set* $M^* = M_0^* = M_1^*$*.*
  - $T_0 \neq T_1$*: If* $(\mathsf{ID}_{i_0}, T_0), (\mathsf{ID}_{i_1}, T_1), (\mathsf{ID}_{i_0}, T_1) \notin \mathsf{STSet}$*, then compute* $\sigma_0^* \leftarrow \mathsf{GSign}(\mathsf{gpk}, \mathsf{tpk}, t_{T_0}, \mathsf{sigk}_{\mathsf{ID}_{i_0}}, M_0^*)$ *and* $\sigma_1^* \leftarrow \mathsf{GSign}(\mathsf{gpk}, \mathsf{tpk}, t_{T_1}, \mathsf{sigk}_{\mathsf{ID}_{i_b}}, M_1^*)$*, and return* σ_0^* *and* σ_1^**.*

  Moreover, we assume that t_{T_0} *and* t_{T_1} *are valid tokens even though these are made by* $\mathcal{A}$*, which means that the* $\mathsf{GVerify}$ *algorithm outputs 1 for all honestly generated signatures with* t_{T_0} *or* t_{T_1}*.*[9]

We say that $\mathcal{GS\text{-}TDL}$ *is anonymous if the advantage* $\mathrm{Adv}^{\textit{anon-tg}}_{\mathsf{GS\text{-}TDL},\mathcal{A}}(\lambda) := |\Pr[\mathrm{Exp}^{\textit{anon-tg-1}}_{\mathsf{GS\text{-}TDL},\mathcal{A}}(\lambda) = 1] - \Pr[\mathrm{Exp}^{\textit{anon-tg-0}}_{\mathsf{GS},\mathcal{A}}(\lambda) = 1]|$ *is negligible for any PPT adversary* $\mathcal{A}$*.*

When $T_0 = T_1$, our definition guarantees that two different signers are unlinkable even if they generate signatures at the same time period. We note that if $\mathcal{A}$ obtains two signatures even though $T_0 = T_1$, then $\mathcal{A}$ can break anonymity by using the Link algorithm. Therefore, $\mathcal{A}$ is allowed to obtain one challenge signature σ^* only. When $T_0 \neq T_1$, our definition guarantees that a signer is still unlinkable if the signer respectively generates two signatures on different time periods. That is, when $\mathcal{A}$ obtains σ_0^*, which is generated by ID_{i_0} at a time T_0, and σ_1^*, which is generated by ID_{i_b} at a time $T_1 \neq T_0$, no $\mathcal{A}$ can distinguish whether two signatures are respectively made by the same user ID_{i_0} or different users ID_{i_0} and ID_{i_1}. In order to prevent a trivial linking attack, $\mathcal{A}$ is not allowed to obtain a signature for (ID_{i_0}, T_1) in this case.

We note that we do not have to consider the case $\mathsf{ID}_{i_0} = \mathsf{ID}_{i_1}$ and $T_0 \neq T_1$, since time T is an input of the verification algorithm. That is, $\mathcal{A}$ can easily break anonymity in this case: $\mathcal{A}$ just obtains $\sigma^* \leftarrow \mathsf{GSign}(\mathsf{gpk}, \mathsf{tpk}, t_{T_b}, \mathsf{sigk}_{\mathsf{ID}_{i_0}}, M^*)$ and checks whether $\mathsf{GVerify}(\mathsf{gpk}, \mathsf{tpk}, \mathsf{RL}_{T_0}, M^*, \sigma^*) = 1$ or not.

Next, we define unforgeability which guarantees that nobody who does not have a signing key or does not have a token can generate a valid signature.

Definition 4.4 (Unforgeability)**.** *For any PPT adversary* $\mathcal{A}$ *and security parameter* $\lambda \in \mathbb{N}$*, we define the experiment* $\mathrm{Exp}^{\textit{unf}}_{\mathsf{GS\text{-}TDL},\mathcal{A}}(\lambda)$ *as follows, where* $\mathcal{O} := (\mathsf{AddU}(\cdot), \mathsf{Revoke}(\mathsf{grs}, \cdot), \mathsf{TokenGen}(\mathsf{tsk}, \cdot), \mathsf{SetToken}(\cdot), \mathsf{GSign}(\cdot,\cdot,\cdot), \mathsf{USK}(\cdot), \mathsf{TSK}(\cdot))$*.*

```
$\mathrm{Exp}^{unf}_{\mathsf{GS\text{-}TDL},\mathcal{A}}(\lambda)$ :
    $params \leftarrow \mathsf{Setup}(1^\lambda)$
    $(\mathsf{gpk}, \mathsf{gsk}, \mathsf{grs}, \mathsf{RL}_0) \leftarrow \mathsf{GKeyGen}(params)$
    $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow \mathsf{TKeyGen}(params)$
    $\mathsf{GU} := \emptyset$; $\mathsf{TSet} := \emptyset$; $\mathsf{SSet} := \emptyset$
    $(M, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk}, \mathsf{tpk})$
```

[9] This condition must be required to exclude the trivially-broken case, e.g., $\mathcal{A}$ honestly generates t_{T_0} and sets t_{T_1} as arbitrary value. Then, $\mathcal{A}$ can check whether σ^* is valid or not. If yes, then $b = 0$ and $b = 1$ otherwise.

```
Return 1 if (1) ∧ (2) ∧ ((3) ∨ (4)) hold :
  (1) $\mathsf{GVerify}(\mathsf{gpk}, \mathsf{tpk}, \mathsf{RL}_{T^*}, M, \sigma) = 1$
  (2) $(T^*, M, \sigma) \notin \mathsf{SSet}$
  (3) $T \notin \mathsf{TSet} \wedge \mathsf{TSK}(\cdot)$ has not been called
  (4) $\mathsf{TSK}(\cdot)$ has been called with non-$\perp$ output
Otherwise return 0
```

- AddU*: The same as before.*
- Revoke*: The same as before. We note that T^* is the challenge time that $\mathcal{A}$ outputs (M, σ).*
- TokenGen*: The token generation oracle takes as input a time T. This oracle runs $t_T \leftarrow \mathsf{TokenGen}(\mathsf{tsk}, T)$, stores T in* TSet*, and returns t_T.*
- SetToken*: The token setting oracle takes as input t_T, and sets t_T as the token at a time T. Without loss of generality, we assume that if the* TokenGen *oracle is called, the* SetToken *oracle is also called right after calling the* TokenGen *oracle. We remark that $\mathcal{A}$ can set arbitrary value as t_T via this oracle.*
- GSign*: The signing oracle takes as input* ID*, T, and a message M. If* $\mathsf{ID} \notin \mathsf{GU}$*, then the oracle runs* AddU(ID)*. If t_T is not generated via the* TokenGen *oracle, then call the oracle* $\mathsf{TokenGen}(\mathsf{tsk}, T)$ *and the* SetToken *oracle. The oracle returns* $\sigma \leftarrow \mathsf{GSign}(\mathsf{gpk}, \mathsf{tpk}, t_T, \mathsf{sigk}_{\mathsf{ID}}, M)$ *and stores (T, M, σ) in* SSet.
- USK*: The user key reveal oracle takes as input* ID*. If the* TSK *oracle was called before, then return $\perp$. If* $\mathsf{ID} \notin \mathsf{GU}$*, then the oracle runs* AddU(ID)*. Return* $\mathsf{sigk}_{\mathsf{ID}}$.
- TSK*: The token key reveal oracle returns $\perp$ if the* USK *oracle was called before and at least one identity is not revoked.*[10] *Otherwise, return* tsk.

We say that $\mathcal{GS}$-$\mathcal{TDL}$ is unforgeable if the advantage $\mathrm{Adv}^{unf}_{\mathsf{GS\text{-}TDL},\mathcal{A}}(\lambda) := \Pr[\mathrm{Exp}^{unf}_{\mathsf{GS\text{-}TDL},\mathcal{A}}(\lambda) = 1]$ *is negligible for any PPT adversary $\mathcal{A}$.*

Finally, we define linking soundness which guarantees that the Link algorithm does not return 1 when two valid signatures are made by either different signers or different time tokens.

Definition 4.5 (Linking Soundness)**.** *For any PPT adversary $\mathcal{A}$ and security parameter $\lambda \in \mathbb{N}$, we define the experiment* $\mathrm{Exp}^{snd}_{\mathsf{GS\text{-}TDL},\mathcal{A}}(\lambda)$ *as follows.*

```
$\mathrm{Exp}^{snd}_{\mathsf{GS\text{-}TDL},\mathcal{A}}(\lambda)$ :
    $params \leftarrow \mathsf{Setup}(1^\lambda)$
    $(\mathsf{gpk}, \mathsf{gsk}, \mathsf{grs}, \mathsf{RL}_0) \leftarrow \mathsf{GKeyGen}(params)$
    $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow \mathsf{TKeyGen}(params)$
    $(\mathsf{ID}_0, \mathsf{ID}_1, T_0, T_1, M, st) \leftarrow \mathcal{A}(\mathsf{gpk}, \mathsf{tpk})$
    $(\mathsf{ID}_0, T_0) \neq (\mathsf{ID}_1, T_1)$
    $\mathsf{sigk}_{\mathsf{ID}_0} \leftarrow \mathsf{Join}(\mathsf{gsk}, \mathsf{grs}, \mathsf{ID}_0);\ \mathsf{sigk}_{\mathsf{ID}_1} \leftarrow \mathsf{Join}(\mathsf{gsk}, \mathsf{grs}, \mathsf{ID}_1)$
    $t_{T_0} \leftarrow \mathsf{TokenGen}(\mathsf{tsk}, T_0);\ t_{T_1} \leftarrow \mathsf{TokenGen}(\mathsf{tsk}, T_1)$
    $\sigma_0 \leftarrow \mathsf{GSign}(\mathsf{gpk}, \mathsf{tpk}, t_{T_0}, \mathsf{sigk}_{\mathsf{ID}_0}, M)$
    $(M^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Revoke}(\mathsf{grs},\cdot)}(st, \mathsf{sigk}_{\mathsf{ID}_1}, t_{T_1}, \sigma_0)$
    Return 1 if
        $\mathsf{Link}(\mathsf{gpk}, \mathsf{tpk}, \mathsf{RL}_{T_1}, (M, \sigma_0, T_0), (M^*, \sigma^*, T_1)) = 1$
    Otherwise return 0
```

[10]That is, the TSK oracle returns tsk if all identities input in the USK oracle were revoked.

- Revoke*:The same as before.*

We say that $\mathcal{GS\text{-}TDL}$ satisfies linking soundness if the advantage $\mathrm{Adv}^{snd}_{\mathsf{GS\text{-}TDL},\mathcal{A}}(\lambda) := \Pr[\mathrm{Exp}^{snd}_{\mathsf{GS\text{-}TDL},\mathcal{A}}(\lambda) = 1]$ *is negligible for any PPT adversary $\mathcal{A}$.*

4.3 Proposed GS-TDL Scheme

In this section, we give our GS-TDL scheme. Since we mainly pursue a *light-weight realization* of the system, here we do not employ structure preserving signatures (e.g., [6]) and Groth-Sahai proofs [33] which are typically used for constructing group signature schemes secure in the standard model (e.g., [32]). We employ the Fiat-Shamir transformation [29] which converts a three-move Σ protocol to non-interactive zero-knowledge (NIZK) proof as in group signature schemes secure in the random oracle model.

The Basic Idea: We mainly follow a common construction methodology of pairing-based group signature schemes [16]. A Boneh-Boyen signature [15] (or a BBS+ signature [8]) is issued to a signer as a certificate ($A = (g_1 h^{-y})^{\frac{1}{\gamma+x}}$ in our scheme), and is encrypted by a public key of the opener. According to the underlying curve selection, basically either the linear encryption scheme or the ElGamal encryption scheme is employed. Because we exclude the open functionality, we exchange the encryption scheme with a commitment scheme ($C = Ah^{\beta}$ in our scheme). A signer proves that a certificate is valid under the group public key by proving the knowledge of the discrete logarithms on the target group (i.e., $\mathbb{G}_T$) of the pairing.

For the linking property, we apply the Franklin-Zhang technique [30] and the Chow technique [23], where a group signature contains Belenkiy et al.'s verifiable random function (VRF) [10]. Concretely, the value $\tau = g^{\frac{1}{x+T}}$ is contained in a signature at a time T, where x is a (part of) signing key. If a signer computes two or more group signatures at a time T, then the value τ is the same, and can be linked without any cryptographic operation. τ itself can be seen as a random value (under the SDDHI assumption), so a signer is still anonymous unless the signer computes two or more group signatures at the same time.

For (verifier-local) revocation, we also apply τ such that τ is added in a revocation list. Note that the verification cost of VLR-type group signatures schemes [17, 46] is $O(|\mathsf{RL}_T|)$, especially, $|\mathsf{RL}_T|$-times pairing computations are required. To avoid such an inefficiency, we use the linkable part τ for revocation and this setting requires no cryptographic operation.

Moreover, we should consider signers that select T by themselves. To protect against the signers, we set tpk and tsk as a verification key and a signing key of a signature scheme respectively. For a time period T, a signature $\Sigma \leftarrow \mathsf{Sign}(\mathsf{tsk}, W_T)$ is computed where $W_T = g_2^T$, and the token t_T is set as (T, W_T, Σ). Anyone can then check whether T is certified by the TGU that has tsk. Note that anyone can compute W_T since T is just the time period and g_2 is a public element. Nevertheless, for the sake of efficiency, we include W_T as a part of t_T in our scheme because signers and verifiers do not have to compute W_T.

Construction 1 (Proposed GS-TDL scheme)**.**

- $\mathsf{Setup}(1^\lambda)$*: Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ be a bilinear group with prime order p, where $\langle g_1 \rangle = \mathbb{G}_1$, $\langle g_2 \rangle = \mathbb{G}_2$, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map. Output $params = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$.*

- $\mathsf{GKeyGen}(params)$*: Choose $\gamma \xleftarrow{\$} \mathbb{Z}_p$, and $h \xleftarrow{\$} \mathbb{G}_1$, and compute $W = g_2^{\gamma}$. Output* $\mathsf{gpk} = (params, h, W, e(g_1, g_2), e(h, W), e(h, g_2), H)$, $\mathsf{gsk} = \gamma$, *where $H : \{0,1\}^* \to \mathbb{Z}_p$ is a hash function modeled as a random oracle,* $\mathsf{grs} := \emptyset$ *and* $\mathsf{RL}_0 := \emptyset$.

- $\mathsf{TKeyGen}(params)$*: Let* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *be a digital signature scheme. Run* $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{Gen}(1^\lambda)$*, and output* $\mathsf{tpk} := \mathsf{vk}$ *and* $\mathsf{tsk} := \mathsf{sigk}$.

- $\mathsf{Join}(\mathsf{gsk}, \mathsf{grs}, \mathsf{ID})$*: Choose $x, y \xleftarrow{\$} \mathbb{Z}_p$, compute $A = (g_1 h^{-y})^{\frac{1}{\gamma+x}}$, output* $\mathsf{sigk}_{\mathsf{ID}} = (x, y, A)$*, and update* $\mathsf{grs} := \mathsf{grs} \cup \{(\mathsf{ID}, x)\}$.

- $\mathsf{TokenGen}(\mathsf{tsk}, T)$: *Assume that* $T \in \mathbb{Z}_p$. *Compute* $W_T = g_2^T$ *and* $\Sigma \leftarrow \mathsf{Sign}(\mathsf{sigk}, W_T)$, *and output* $t_T = (T, W_T, \Sigma)$.

- $\mathsf{GSign}(\mathsf{gpk}, \mathsf{tpk}, t_T, \mathsf{sigk}_{\mathsf{ID}}, M)$: *Let* $\mathsf{sigk}_{\mathsf{ID}} = (x, y, A)$ *and* $t_T = (T, W_T, \Sigma)$. *If* $\mathsf{Verify}(\mathsf{vk}, W_T, \Sigma) = 0$, *then output* $\perp$. *Otherwise, choose* $\beta \xleftarrow{\$} \mathbb{Z}_p$, *set* $\delta = \beta x - y$, *and compute* $C = Ah^\beta$ *and* $\tau = g_1^{\frac{1}{x+T}}$. *Choose* $r_x, r_\delta, r_\beta \xleftarrow{\$} \mathbb{Z}_p$, *and compute*

$$R_1 = \frac{e(h, g_2)^{r_\delta} e(h, W)^{r_\beta}}{e(C, g_2)^{r_x}}, \ R_2 = e(\tau, g_2)^{r_x}$$
$$c = H(\mathsf{gpk}, \mathsf{tpk}, C, \tau, R_1, R_2, M)$$
$$s_x = r_x + cx, s_\delta = r_\delta + c\delta, \text{and } s_\beta = r_\beta + c\beta,$$

  and output $\sigma = (C, \tau, c, s_x, s_\delta, s_\beta)$. *This proves that (1)* (x, y, A) *is a valid Boneh-Boyen signature under* gpk *(i.e.,* $\mathsf{sigk}_{\mathsf{ID}}$ *is issued by Key Issuer) and (2)* τ *is computed by the same* x.

  Pairing-free Variant: *We remark that* $e(h, g_2)$ *and* $e(h, W)$ *are pre-computable and can be contained in* gpk. *Moreover* $e(C, g_2)^{r_x}$ *and* $e(\tau, g_2)^{r_x}$ *can be represented as* $e(A, g_2)^{r_x} e(h, g_2)^{\beta r_x}$ *and* $e(g_1, g_2)^{\frac{r_x}{x+T}}$, *respectively. Then,*

$$R_1 = \frac{e(h, g_2)^{r_\delta - \beta r_x} e(h, W)^{r_\beta}}{e(A, g_2)^{r_x}} \text{ and } R_2 = e(g_1, g_2)^{\frac{r_x}{x+T}}.$$

  So, by assuming that $e(A, g_2)$ *is pre-computable (we can simply assume that* $e(A, g_2)$ *is contained in* $\mathsf{sigk}_{\mathsf{ID}}$*), we can remove any pairing computation from the signing algorithm, instead of adding two exponentiations over* $\mathbb{G}_T$.

- $\mathsf{Revoke}(\mathsf{gpk}, \mathsf{grs}, \{\mathsf{ID}_{T,1}, \ldots, \mathsf{ID}_{T,n_T}\})$: *If there exists* $\mathsf{ID} \in \{\mathsf{ID}_{T,1}, \ldots, \mathsf{ID}_{T,n_T}\}$ *that is not joined to the system via the* Join *algorithm, then output* $\perp$. *Otherwise, extract* $(\mathsf{ID}_{T,1}, x_{T,1}), \ldots, (\mathsf{ID}_{T,n_T}, x_{T,n_T})$ *from* grs. *Output* $\mathsf{RL}_T := \{(\mathsf{ID}_{T,1}, g_1^{\frac{1}{x_{T,1}+T}}), \ldots (\mathsf{ID}_{T,n_T}, g_1^{\frac{1}{x_{T,n_T}+T}})\}$.

- $\mathsf{GVerify}(\mathsf{gpk}, \mathsf{tpk}, \mathsf{RL}_T, M, \sigma)$: *Assume that* $\mathsf{Verify}(\mathsf{vk}, W_T, \Sigma) = 1$ *(if not, output* $\perp$*). Parse* $\sigma = (C, \tau, c, s_x, s_\delta, s_\beta)$. *If* τ *is contained in* RL_T *such that* $(\mathsf{ID}, \tau) \in \mathsf{RL}_T$ *for some* ID, *then output* 0. *Otherwise, compute*

$$R_1' = \frac{e(h, g_2)^{s_\delta} e(h, W)^{s_\beta}}{e(C, g_2)^{s_x}} \left(\frac{e(C, W)}{e(g_1, g_2)}\right)^{-c} \text{ and}$$

$$R_2' = e(\tau, g_2)^{s_x} \left(\frac{e(g_1, g_2)}{e(\tau, W_T)}\right)^{-c},$$

  and output 1 *if* $c = H(\mathsf{gpk}, \mathsf{tpk}, C, \tau, R_1', R_2', M)$ *holds, and* 0 *otherwise. We remark that* $e(h, g_2)$, $e(h, W)$ *and* $e(g_1, g_2)$ *are pre-computable and contained in* gpk.

- $\mathsf{Link}(\mathsf{gpk}, \mathsf{tpk}, \mathsf{RL}_T, (M_0, \sigma_0, T_0), (M_1, \sigma_1, T_1))$: *Parse* $\sigma_0 = (C_0, \tau_0, c_0, s_{x,0}, s_{\delta,0}, s_{\beta,0})$ *and* $\sigma_1 = (C_1, \tau_1, c_1, s_{x,1}, s_{\delta,1}, s_{\beta,1})$. *If either* $T \neq T_0$ *or* $T \neq T_1$, *then output* 0. *Else if either* $\mathsf{GVerify}(\mathsf{gpk}, \mathsf{tpk}, \mathsf{RL}_{T_0}, M_0, \sigma_0) = 0$ *or* $\mathsf{GVerify}(\mathsf{gpk}, \mathsf{tpk}, \mathsf{RL}_{T_0}, M_1, \sigma_1) = 0$, *then output* 0. *Otherwise, output* 1 *if* $\tau_0 = \tau_1$, *and* 0 *otherwise.*

Since τ just depends on T and x, and does not contain any randomness, we can directly use τ for revocation. Then we can achieve the (almost) constant verification cost by using hash tables which are made in the Revoke algorithm.

As a remark, the open algorithm, where an authority can identify the actual signer, also can be implemented (though we do not use it) as follows: let (ID, g_2^x) be preserved in the join phase, and the open algorithm checks whether $e(\tau, g_2^x g_2^T) = e(g_1, g_2)$ or not. If the equation holds, then ID is the identity of the corresponding signer. This open algorithm is essentially the same as that of the Bichsel et al. scheme [13].

We give the security proofs of the following theorems in Appendix.

Theorem 4.1. *The proposed GS-TDL scheme has anonymity in the random oracle model under the SDDHI assumption, where H is modeled as a random oracle.*

Theorem 4.2. *The proposed GS-TDL scheme has unforgeability in the random oracle model if the q-SDH assumption holds and* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *is EUF-CMA, where q is the number of signers and H is modeled as a random oracle.*

Theorem 4.3. *The proposed GS-TDL scheme has linking soundness.*

5 Anonymous Time-dependent Authentication and its Experimental Results

In this section, we construct an anonymous time-dependent authentication system via GS-TDL, and provide its experimental results.

5.1 Experimental Results of Our GS-TDL

Our implementation uses the RELIC library (ver.0.4.1) [7] for elliptic curve operations and the pairing operation, OpenSSL[11] (ver.1.0.2d) for standard signing and verifying, and GLib[12] (ver.2.47.1) for the hash table for (almost) constant-time searching. We note that we employ asymmetric pairing settings ((type III) Barreto-Naehrig (BN) curves [9]) with 254-bit order. We use the parameter of "Nogami-Aranha" given in an IETF Internet-Draft [37]. Table 1 shows the specifications of the machines we employed for experiments.

Table 1: Machine specification

Machine	x86_64 Server	Raspberry Pi (Model B)
CPU	Xeon E5-2660 v3 (2.60 GHz)	ARM1176JZF-S (700 MHz)
RAM	128GB	512 MB

Table 2 shows the running time of group operations.

Table 2: Group operations over 254-bit BN curves (μseconds)

Operations		x86_64 Server	Raspberry Pi
Mul($\mathbb{G}_1$,U)		160	6,814
Mul($\mathbb{G}_1$,K)		51	2,081
Mul($\mathbb{G}_2$,U)		310	28,011
Mul($\mathbb{G}_2$,K)		67	5,388
Exp($\mathbb{G}_T$,U)		467	36,653
Pairing	Miller loop	466	35,477
	Final exp.	208	14,663

Here, Mul($\mathbb{G}_1$,Type), Mul($\mathbb{G}_2$,Type), and Exp($\mathbb{G}_T$,Type) are scalar multiplication on $\mathbb{G}_1$ and $\mathbb{G}_2$, and exponentiation on $\mathbb{G}_T$, respectively. If a base point is previously known, then Type is set as K, and U otherwise. We note that we always use Type U for operations on $\mathbb{G}_T$ since the RELIC library does not support Type K operation on $\mathbb{G}_T$.

Table 3 shows the running time of singing and verification algorithms of OpenSSL 3072-bit RSA, 3072-bit DSA, and prime 256v1 ECDSA on our environment.

[11] `https://www.openssl.org`
[12] `https://wiki.gnome.org/Projects/GLib`

Table 3: OpenSSL signing and verification costs (milliseconds)

		x86_64 Server	Raspberry Pi
3072-bit RSA	Sign	2.844	236.736
	Verify	0.061	5.165
3072-bit DSA	Sign	0.942	75.864
	Verify	1.154	90.917
prime256v1 ECDSA	Sign	0.041	12.383
	Verify	0.104	14.716

Table 4: The number of operations for each algorithms

Algorithms	Operations
GSign (Original)	2 Mul($\mathbb{G}_1$,K) + 4 Exp($\mathbb{G}_T$) + 2 Pairing (+Verify)
GSign (Pairing-free)	2 Mul ($\mathbb{G}_1, K$) + 4 Exp($\mathbb{G}_T$) (+Verify)
GSign (Optimized)	4 Mul($\mathbb{G}_1$,K) + 2 Exp($\mathbb{G}_T$) + 1 Pairing (+Verify)
TokenGen	1 Mul ($\mathbb{G}_2$,K) + Sign
GVerify (Original)	6 Exp($\mathbb{G}_T$) + 4 Pairing (+ Verify)
GVerify (Optimized)	5 Mul($\mathbb{G}_2$,K) + 1 Exp($\mathbb{G}_T$) + 3 Miller loop + 2 Final exp. (+Verify)
Revoke	$\lvert \mathsf{RL}_T \rvert$ Mul($\mathbb{G}_1$,K)

Optimization of GSign/GVerify: Originally, our GSign algorithm requires 2 Mul($\mathbb{G}_1$,K) + 4 Exp($\mathbb{G}_T$,U) + 2 Pairing (+Verify) and our GVerify algorithm requires 6 Exp($\mathbb{G}_T$,U) + 4 Pairing (+Verify) Here, we apply bilinearity of pairing in order to optimize the costs of algorithms by using Table 2 as follows. For GSign, we modify R_1 and R_2 as

$$R_1 = e(h^{r_\delta - r_x\beta} A^{-r_x}, g_2)\ e(h, W)^{r_\beta},$$
$$R_2 = e(g_1, g_2)^{\frac{r_x}{x+T}},$$

and this modification allows us to run the GSign algorithm with 4 Mul($\mathbb{G}_1$,K) + 2 Exp($\mathbb{G}_T$,U) + 1 Pairing (+Verify) Moreover, for GVerify, we modify R_1' and R_2' as

$$R_1' = e(h, g_2^{s_\delta} W^{s_\beta})\ e(C, g_2^{-s_x} W^{-c})\ e(g_1, g_2)^c,$$
$$R_2' = e(\tau, g_2^{s_x} W_T^c)\ e(g_1, g_2)^{-c},$$

and this modification allows us to run the GVerify algorithm with 5 Mul($\mathbb{G}_2$,K) + 1 Exp($\mathbb{G}_T$,U) + 3 Miller loop + 2 Final exp. (+Verify). As a remark, we use Mul($\mathbb{G}_2$,K) for computing W_T^c since W_T is fixed during the time period T. If T is frequently updated, Mul($\mathbb{G}_2$,U) should be used instead. We summarize the number of operations for each algorithms in Table 4. We can reduce the number of Exp($\mathbb{G}_T$,U) and pairings from the originals. In our environment, the running time of GSign (Optimized) is faster than that of GSign (Pairing-free). However, if the running time of 2 Exp($\mathbb{G}_T$,Type) is faster than that of 1 Pairing, then Pairing-free variant is more efficient.

Benchmarks of Each Algorithms: We give our experimental results of our GS-TDL scheme in Table 5. The total number of signers is 10,000,000, and the number of revoked signers is specified in parentheses () in the GVerify algorithm and the Revoke algorithm. We employ 3072-bit RSA for (TokenGen, Sign, Verify) which is used in our GS-TDL scheme since the verification cost (which is run by signers in the GSign algorithm) is faster than that of DSA and ECDSA in our environment.

We evaluate our results as follows.

- **(Almost) Constant Verification Cost**: First of all, we should highlight that cryptographic operations in the GVerify algorithm do not depend on the number of revoked vehicles (i.e., scalable) due to our time-dependent linkability, i.e., τ is deterministic, though we employ VLR-type revocation. In our implementation, a table preserves (ID, x) in the Join algorithm, and the table

Table 5: Benchmarks (milliseconds)

Algorithms	x86_64 Server	Raspberry Pi
Join	0.146	-
TokenGen	2.92	238.553
GSign (Optimized)	2.078	142.472
GSign (Pairing-free)	2.212	161.616
GVerify(1,000)	2.889	210.883†
GVerify(10,000)	2.891	-
GVerify(100,000)	2.892	-
GVerify(1,000,000)	2.888	-
Revoke(1,000)	55.706	-
Revoke(10,000)	555.466	-
Revoke(100,000)	5,578.380	-
Revoke(1,000,000)	55,640.379	-

† The GVerify algorithm on Raspberry Pi does not contain the cost of serching on the revocation list.

is regarded as grs. We set ID as a searching key (i.e., the table takes ID as input, and outputs the corresponding x). In the Revoke algorithm, an array of (ID, x, τ) is constructed, and each τ contained in the array is updated at T such that $\mathsf{RL}_T := \{(\mathsf{ID}_{T,1}, g_1^{\frac{1}{x_{T,1}+T}}), \dots (\mathsf{ID}_{T,n_T}, g_1^{\frac{1}{x_{T,n_T}+T}})\}$. The corresponding hash table of τ is generated for (almost) constant-time searching. Therefore, the cost of the Revoke algorithm depends on the number of revoked vehicles (but we emphasize that this procedure is run by the key issuer, who has plenty of computational resource). In the GVerify algorithm, the verifier can easily check whether τ is contained in RL_T or not by using the hash table without any cryptographic operation.

- **Practically Efficient Signing Cost**: In some situation, such as a road-to-vehicle communication system introduced in the next subsection, a signer has a constrained computational power, and moreover the signer needs to generate signatures frequently. In our experimental result, the GSign algorithm can be run at 142.472 milliseconds even on Raspberry Pi, and it is comparable to RSA/DSA with similar security level, though our system additionally supports anonymity. If the signer has a standard computational power (as in x86_64 server), then the GSign algorithm can be run at 2.078 milliseconds. This result shows that our system is feasible enough in practice.

5.2 Network Simulation

For constructing an actual system, we need to find a suitable time period interval. If we set T as short as possible, then the anonymity level of our system comes closer to that of group signature schemes (i.e., signatures are always unlinkable). If we set T as long as possible, then the anonymity level of our system comes closer to that of usual signature schemes (i.e., signatures are always linkable). Because a suitable time period depends on applications, here we investigate the minimum time period interval in which the system works. For example, if T is too short, then RSUs may not be able to start verification of signatures since they need to obtain the revocation list at the beginning of the time period.

Moreover, we should consider a "blank" time period during which vehicles stop to generate signatures. This blank time is necessary to cut the linkage of location information which would be contained in messages. For example, suppose that a vehicle broadcasts a message and a corresponding signature periodically, e.g. in a second, and there is no such blank time. The last message generated in time period T and the first message generated by the next time period are then linkable although corresponding signatures are unlinkable, because locations indicated by these messages are too close to guess that these are generated by the same vehicle.

Estimating the Minimum Time Period Interval: First, we estimate the minimum time period interval, say T_{min} seconds, during which the system works. Let N_{receive} be the number of signatures that an RSU receives per second, and N_{verify} be the number of signatures that an RSU can verify per second.

Let t_{RL} be the time during which an RSU waits to receive the revocation list. During the current time period T, an RSU receives $N_{\mathsf{receive}} \cdot T$ signatures. On the other hand, the RSU can verify $N_{\mathsf{verify}}(T - t_{\mathsf{RL}} + T_{\mathsf{blank}})$ signatures, where T_{blank} is the blank time period. Recall that RSUs cannot start the verification procedure before receiving the revocation list. If the verification procedure is finished before the next time periods starts, then $N_{\mathsf{receive}} \cdot T \leq N_{\mathsf{verify}}(T - t_{\mathsf{RL}} + T_{\mathsf{blank}})$ need to hold. Remark that the purpose of introducing T_{blank} in our simulations is to cut the linkage of location information. Thus, the system should work even if $T_{\mathsf{blank}} = 0$. In this case, if $N_{\mathsf{receive}} \geq N_{\mathsf{verify}}$ then no RSU can complete the verification procedure. Thus, we assume that $N_{\mathsf{verify}} > N_{\mathsf{receive}}$ here, and we can estimate T_{min} as follows.

$$T_{\mathsf{min}} = N_{\mathsf{verify}} \cdot t_{\mathsf{RL}} / (N_{\mathsf{verify}} - N_{\mathsf{receive}})$$

Although N_{verify} and t_{RL} can be estimated by Table 5, for estimating N_{receive}, it is necessary to simulate in a reasonable setting because it is very situation-dependent.

System Architecture: Here, we define four entities as follows: a vehicle $\mathcal{V}$, the vehicle manufacturer $\mathcal{VM}$, the Road Side Unit $\mathcal{RSU}$, and the Token Generation Unit $\mathcal{TGU}$. We can consider multiple $\mathcal{RSU}$s but we assume only one $\mathcal{TGU}$ in this paper. We assume that $params \leftarrow \mathsf{Setup}(1^\lambda)$ has been honestly run, and all entities share $params$. First, $\mathcal{VM}$ runs $(\mathsf{gpk}, \mathsf{gsk}, \mathsf{grs}, \mathsf{RL}_0) \leftarrow \mathsf{GKeyGen}(params)$ and $\mathcal{TGU}$ runs $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow \mathsf{TKeyGen}(params)$. When a vehicle $\mathcal{V}$ is sold, $\mathcal{VM}$ runs $\mathsf{sigk}_{\mathsf{ID}} \leftarrow \mathsf{Join}(\mathsf{gsk}, \mathsf{grs}, \mathsf{ID})$, and $\mathcal{V}$ preserves $\mathsf{sigk}_{\mathsf{ID}}$. At a time period T, $\mathcal{TGU}$ runs $t_T \leftarrow \mathsf{TokenGen}(\mathsf{tsk}, T)$ and broadcasts t_T. Moreover, $\mathcal{VM}$ updates the revocation list, and sends RL_T to all $\mathcal{RSU}$. A vehicle $\mathcal{V}$ generates a group signature on local information M such that $\sigma \leftarrow \mathsf{GSign}(\mathsf{gpk}, \mathsf{tpk}, t_T, \mathsf{sigk}_{\mathsf{ID}}, M)$.

Simulation Setup: We consider the following cases based on the timing of generating signatures.

- Method 1: A vehicle $\mathcal{V}$ generates σ every few seconds and broadcasts (M, σ, T). Here, we assume that each vehicle moves at a speed of approximately 40 km/h in our simulation, and we set three seconds as the signature generation interval. Each vehicle then reports local information to $\mathcal{RSU}$s at intervals of approximately three meters. It seems reasonable to trace travel routes.
- Method 2: A vehicle $\mathcal{V}$ generates σ and sends (M, σ, T) to an $\mathcal{RSU}$ when $\mathcal{V}$ enters a certain range of the $\mathcal{RSU}$. In our simulation, the range is set as 239m.[13]

In Method 1, vehicles must generate and broadcast signatures even no $\mathcal{RSU}$ exists around them. Thus, vehicles send more signatures than Method 2. Moreover, $\mathcal{RSU}$s also receive many signatures than those of Method 2. At the expense of these costs, Method 1 seems better to collect linkable information. We will discuss which method is better from the simulation results, and will conclude that Method 1 seems better in a rural area, and Method 2 seems better in an urban area.

We set T as 600 seconds and T_{blank} as 600 seconds. After the blank time period T_{blank}, the next time period will start, and vehicles generate signatures (according to either Method 1 or 2). Since vehicles send signatures to $\mathcal{RSU}$s via wireless communications, $\mathcal{RSU}$s may not be able to receive signatures owing to the wireless communication environments. Thus, we need to consider the visibility of the location that relates the path loss of the communication. In radio wave propagation analysis, there are two path loss models for line-of-sight (LoS) and non-line-of-sight (NLoS) environments. In an urban area (which has low visibility), the LoS model is employed. In a rural area (which has high visibility), the NLoS model is employed. We select two places with different levels of visibility. One is Ginza area, which has low visibility owing to tall buildings, and the LoS model is employed. The other place is Koganei area, which has low visibility (around the station located in the lower-left part of the map (Fig 5 and 6)) and high visibility (other places of the map), and the LoS model and the NLoS model are employed according to the visibility. The road network considered covers an area of 2.4 $\times$ 2.4 km^2. Moreover, we set 2,000 vehicles and 1,000 vehicles in these areas, respectively. As approximately 200,000

[13]This range (239m) is specified as the requirement of road-to-vehicle communication systems when they support safe driving: `http://www.soumu.go.jp/main_content/000128458.pdf` (Japanese). We adopt this range in our simulation.

vehicles are discarded per year in Japan[14], we set the number of revoked vehicles as 4,000,000 which covers the number of discarded vehicles in a period of 20 years. Moreover, we employ IEEE 802.11p as the communication protocol from a vehicle $\mathcal{V}$ to an $\mathcal{RSU}$.

The detailed simulation flow is given as follows:

1. At the beginning of time period T, $\mathcal{VM}$ generates the current revocation list, and each $\mathcal{RSU}$ receives it. In the simulation, each $\mathcal{RSU}$ waits t_{RL} seconds which are determined by the size of the revocation list and Table 5. We set $t_{\mathsf{RL}} = 278.2$ seconds, and assume that there is a channel to send the revocation list to each $\mathcal{RSU}$, and we ignore the sending time.

2. Immediately after T starts, each vehicle $\mathcal{V}$ generates a group signature and send it to $\mathcal{RSU}$s (according to either Method 1 or 2). We assume that each vehicle can obtain a time token immediately, and assume that this time can be ignored. In the simulation, each vehicle $\mathcal{V}$ waits 142.472 milliseconds (see Table 5) for generating a group signature. If $\mathcal{RSU}$s wait to receive the revocation list sent from $\mathcal{VM}$, then $\mathcal{RSU}$s preserve group signatures sent from $\mathcal{V}$s, and verify them after receiving the list. In the simulation, each $\mathcal{RSU}$ waits 2.90 milliseconds (see Table 5) for a group signature verification.

3. During the blank time T_{blank}, each vehicle $\mathcal{V}$ does nothing. $\mathcal{RSU}$s verify the signatures, if $\mathcal{RSU}$s have group signatures that have not been verified.

We employ a traffic simulator PTV, a radio wave propagation analysis tool RapLab, and a network simulator QualNet. Simulation environments are shown in Table 6. The running time of PTV and RapLab is approximately 20 minutes in total, and that of QualNet is approximately 8 hours in total.

Table 6: Simulation Environments

Software	CPU	RAM
PTV VISSIM 5.4	Core i7-4500U 1.8GHz	4GB
RapLab 8.0.4	Core i5-3470 3.2GHz	12GB
QualNet 7.4	Xeon E5-2643v3 3.40GHz $\times$ 2	256GB

The Number of Stacked Signatures on RSUs: Because each vehicle sends group signatures during $\mathcal{RSU}$s and wait to receive the revocation list, these signatures are stacked in $\mathcal{RSU}$s until they receive the revocation list. If the number of stacked signatures becomes large, then $\mathcal{RSU}$s may not complete the verification of signatures before the next time period starts. Thus, we pick up the $\mathcal{RSU}$s that preserve the highest number of stacked signatures in our simulations. See Table 7. In both cities, the highest number of stacked signatures appears in the case of Method 1. This is a reasonable result since vehicles do not send new signatures in Method 2 if they do not newly enter the range of $\mathcal{RSU}$s.

Table 7: The highest number of stacked signatures

City	Method	#Sig
Ginza	1	3,441
Ginza	2	525
Koganei	1	9,371
Koganei	2	1,414

Somewhat surprisingly, the busiest $\mathcal{RSU}$ which accumulates 9,371 signatures, appears in Koganei area, though it is a relatively rural area compared to Ginza area. In the case of Koganei with Method 1, the $\mathcal{RSU}$ is located in a crossroad (we display the location of the $\mathcal{RSU}$ by the red circle in Fig 5), and there is a traffic light. Since vehicles broadcast signatures when they pause at a light, the $\mathcal{RSU}$ receives more signatures than the others. We show the number of stacked signatures for the $\mathcal{RSU}$ in Fig 2. As the figure shows, the number of stacked signatures is immediately reduced after the $\mathcal{RSU}$s start to verify the signatures.

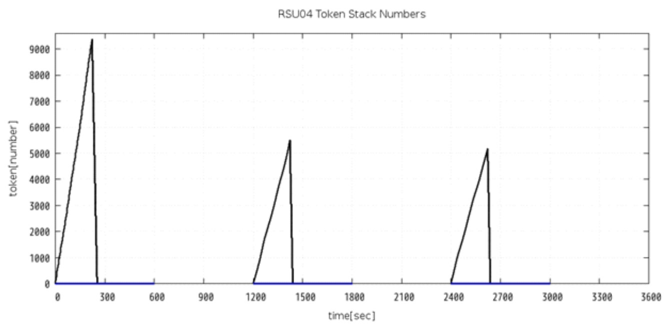

Figure 2: RSUstack

In our simulations, an $\mathcal{RSU}$ receives no more than 10,000 signatures per 300 seconds. Thus, we roughly estimate and set N_{receive} as 40. From our benchmarks (Table 5), each $\mathcal{RSU}$ can complete the verification of a signature within 2.9 milliseconds. Thus, each $\mathcal{RSU}$ can verify approximately 344 signatures per second, so we set $N_{\mathsf{verify}} = 344$. Moreover, from a reasonable setting and our benchmarks (Table 5), we set $t_{\mathsf{RL}} = 278.2$ seconds. From these values, we can estimate $T_{\mathsf{min}} = 314.8$ seconds.

Visualization of our simulation: Finally, we display our simulation results over maps in Fig 3, 4, 5, and 6. We use OpenStreetMap.[15] Each dot indicates a group signature generated by a vehicle. Red triangles indicate $\mathcal{RSU}$s. We display the location of the $\mathcal{RSU}$s picked up in Table 7 by the red circles. In our simulation (60 minutes), a vehicle generates signatures for 600 seconds which are colored by magenta. After the first (resp. second) blank time (600 seconds), the vehicle again generates signatures for 600 seconds which are colored by black (resp. brown). Recall that signatures generated in the same time period are linkable (these colors are the same), and signatures generated at different time periods are unlinkable (these colors are different). For the sake of clarity, we draw a blue line as the route through which the vehicle travelled during the simulation. We track the same vehicle in Ginza area and Koganei area.

Finally, we discuss which method is better from the simulation results. In Ginza area (an urban area), each $\mathcal{RSU}$ is frequently located as there are many crossroads. Thus, both methods work well for correcting linkable information. As the number of stacked signatures in Method 2 is smaller than that of Method 1, Method 2 seems to work better in an urban area. In Koganei area (a rural area), each $\mathcal{RSU}$ is infrequently located. Thus, Method 1 (Fig 5) performs superior to Method 2 (Fig 6) for correcting linkable information because we can easily follow the track of the route that a vehicle passes through in 600 seconds. Thus, Method 1 has better performance in rural areas.

6 Conclusion

In this paper, we propose a road-to-vehicle communication system via group signatures with time-token dependent linking (GS-TDL), and our network simulation demonstrates that our algorithm fits well for road-to-vehicle communication systems. Our system not only has good performance but also supports the revocation property which is essential in real systems. It is particularly worth noting that no vehicle is involved in the revocation procedure and the signing/verification costs do not depend on the number of revoked vehicles.

We set RSUs in appropriate locations, e.g., at traffic lights of crossroads, in the middle of a long road, etc. However, we may consider alternative installation locations for RSUs that may better reflect real environments. We leave this as a future work to follow up this paper.

[14] Japan Automobile Dealers Association (Japanese): `http://www.jada.or.jp/`

[15] OpenStreetMap: `https://www.openstreetmap.org/`

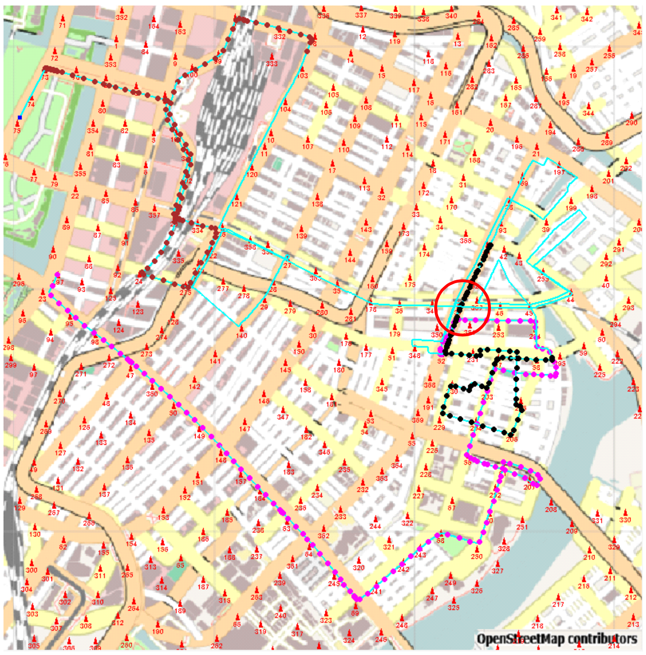
Figure 3: Method 1: Ginza, 2,000 vehicles

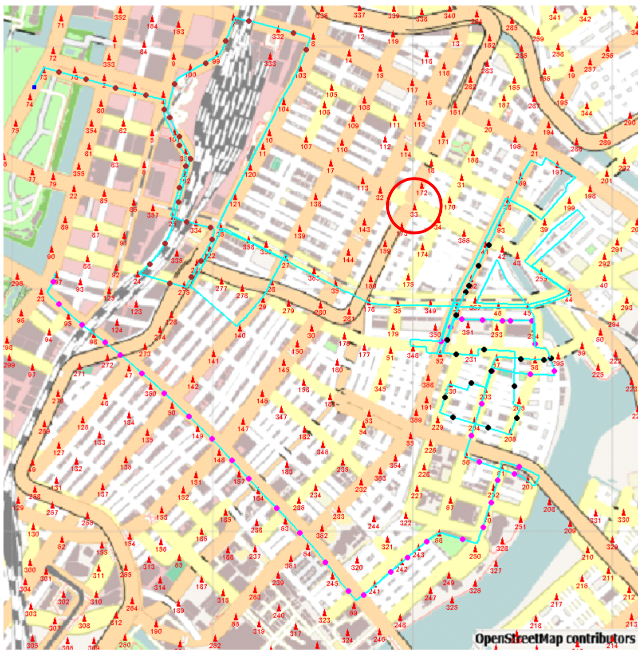
Figure 4: Method 2: Ginza, 2,000 vehicles

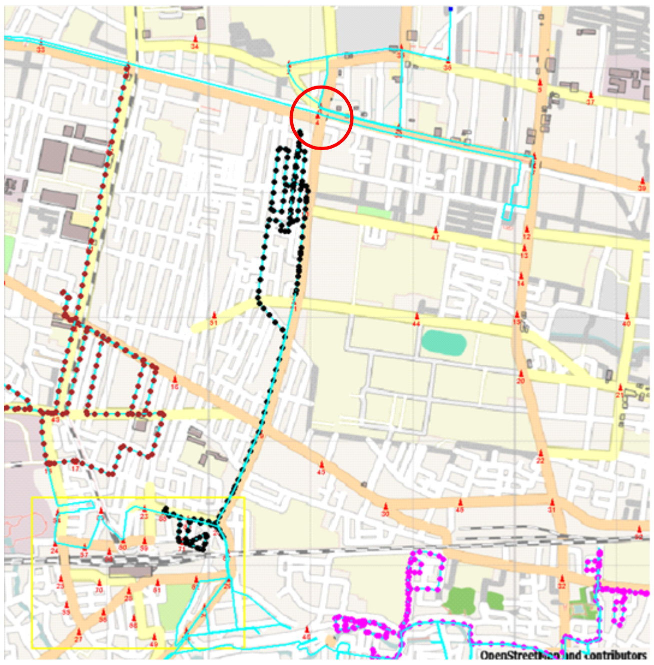
Figure 5: Method 1: Koganei, 1,000 vehicles

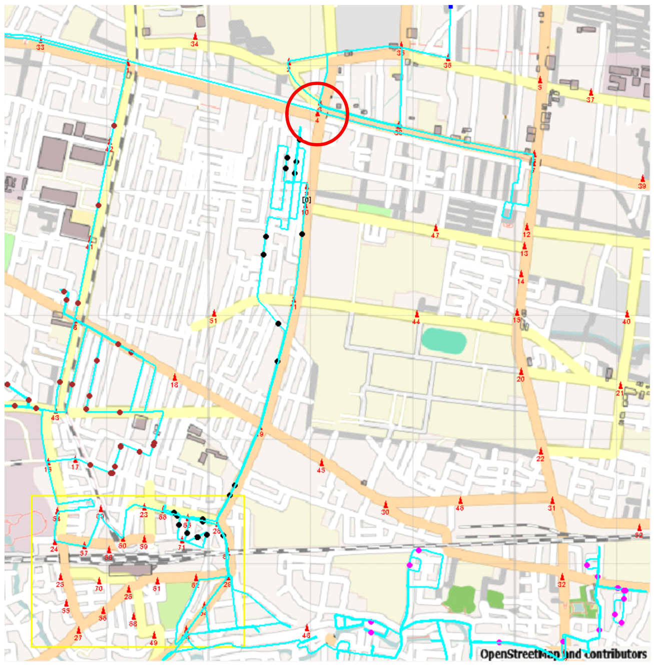
Figure 6: Method 2: Koganei, 1,000 vehicles

In these figures, red triangles indicate $\mathcal{RSU}$s and each dot indicates a group signature generated by a vehicle. We pick up the $\mathcal{RSU}$s that preserve the highest number of stacked signatures in our simulations, and display the location of these $\mathcal{RSU}$s by the red circle. We draw a blue line as the route through which the vehicle travelled during the simulation. We track the same vehicle in Ginza area and Koganei area.

Acknowledgement

We would like to thank Dr. Ryo Nojima for his helpful comments and suggestions.

References

[1] IntelliDrive Program. Available at http://www.intellidrive.org.

[2] ITS Info-communications Forum. Available at http://www.itsforum.gr.jp/E_index.html.

[3] PRESERVE: Preparing Secure Vehicle-to-X Communication Systems. Available at http://www.preserve-project.eu/.

[4] TEPLA: University of Tsukuba Elliptic Curve and Pairing Library. Available at http://www.cipher.risk.tsukuba.ac.jp/tepla/index_e.html.

[5] M. Abe, S. S. M. Chow, K. Haralambiev, and M. Ohkubo. Double-trapdoor anonymous tags for traceable signatures. *Int. J. Inf. Sec.*, 12(1):19–31, 2013.

[6] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. Structure-preserving signatures and commitments to group elements. In *CRYPTO*, pages 209–236, 2010.

[7] D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic.

[8] M. H. Au, W. Susilo, and Y. Mu. Constant-size dynamic k-TAA. In *Security and Cryptography for Networks*, pages 111–125, 2006.

[9] P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography*, pages 319–331, 2005.

[10] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. Compact e-cash and simulatable VRFs revisited. In *Pairing*, pages 114–131, 2009.

[11] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT*, pages 614–629, 2003.

[12] M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA*, pages 136–153, 2005.

[13] P. Bichsel, J. Camenisch, G. Neven, N. P. Smart, and B. Warinschi. Get shorty via group signatures without encryption. In *SCN*, pages 381–398, 2010.

[14] O. Blazy, D. Derler, D. Slamanig, and R. Spreitzer. Non-interactive plaintext (in-)equality proofs and group signatures with verifiable controllable linkability. In *CT-RSA*, pages 127–143, 2016.

[15] D. Boneh and X. Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.

[16] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *CRYPTO*, pages 41–55, 2004.

[17] D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *ACM Conference on Computer and Communications Security*, pages 168–177, 2004.

[18] J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, and J. Groth. Foundations of fully dynamic group signatures. Cryptology ePrint Archive, Report 2016/368, 2016. http://eprint.iacr.org/.

[19] C. Busold, A. Taha, C. Wachsmann, A. Dmitrienko, H. Seudie, M. Sobhani, and A.-R. Sadeghi. Smart keys for cyber-cars: secure smartphone-based NFC-enabled car immobilizer. In *CODASPY*, pages 233–242, 2013.

[20] J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich. How to win the clone wars: efficient periodic n-times anonymous authentication. In *ACM Conference on Computer and Communications Security*, pages 201–210, 2006.

[21] D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT*, pages 257–265, 1991.

[22] B. K. Chaurasia, S. Verma, and S. M. Bhasker. Message broadcast in VANETs using group signature. In *WCSN*, pages 131–136, 2009.

[23] S. S. M. Chow. Real traceable signatures. In *Selected Areas in Cryptography*, pages 92–107, 2009.

[24] S. S. M. Chow, W. Susilo, and T. H. Yuen. Escrowed linkability of ring signatures and its applications. In *VIETCRYPT*, pages 175–192, 2006.

[25] D. Derler and D. Slamanig. Fully-anonymous short dynamic group signatures without encryption. Cryptology ePrint Archive, Report 2016/154, 2016. `http://eprint.iacr.org/`.

[26] C. Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.

[27] K. Emura and T. Hayashi. A light-weight group signature scheme with time-token dependent linking. In *LightSec*, pages 37–57, 2015.

[28] K. Emura, A. Kanaoka, S. Ohta, K. Omote, and T. Takahashi. Secure and anonymous communication technique: Formal model and its prototype implementation. *IEEE Trans. Emerging Topics Comput.*, 4(1):88–101, 2016.

[29] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

[30] M. K. Franklin and H. Zhang. Unique group signatures. In *ESORICS*, pages 643–660, 2012.

[31] J. Furukawa and H. Imai. An efficient group signature scheme from bilinear maps. *IEICE Transactions*, 89-A(5):1328–1338, 2006.

[32] J. Groth. Fully anonymous group signatures without random oracles. In *ASIACRYPT*, pages 164–180, 2007.

[33] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT*, pages 415–432, 2008.

[34] J. Y. Hwang, L. Chen, H. S. Cho, and D. Nyang. Short dynamic group signature scheme supporting controllable linkability. *IEEE Transactions on Information Forensics and Security*, 10(6):1109–1124, 2015.

[35] J. Y. Hwang, S. Lee, B.-H. Chung, H. S. Cho, and D. Nyang. Short group signatures with controllable linkability. In *LightSec*, pages 44–52, 2011.

[36] J. Y. Hwang, S. Lee, B.-H. Chung, H. S. Cho, and D. Nyang. Group signatures with controllable linkability for dynamic membership. *Inf. Sci.*, 222:761–778, 2013.

[37] K. Kasamatsu, S. Kanno, T. Kobayashi, and Y. Kawahara. Barreto-naehrig curves. Internet-Draft draft-kasamatsu-bncurves-01, IETF Secretariat, July 2015.

[38] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *EUROCRYPT*, pages 571–589, 2004.

[39] V. Kumar, H. Li, J. J. Park, K. Bian, and Y. Yang. Group signatures with probabilistic revocation: A computationally-scalable approach for providing privacy-preserving authentication. In *ACM Conference on Computer and Communications Security*, pages 1334–1345, 2015.

[40] B. Libert and D. Vergnaud. Group signatures with verifier-local revocation and backward unlinkability in the standard model. In *CANS*, pages 498–517, 2009.

[41] J. K. Liu, C. Chu, S. S. M. Chow, X. Huang, M. H. Au, and J. Zhou. Time-bound anonymous authentication for roaming networks. *IEEE Transactions on Information Forensics and Security*, 10(1):178–189, 2015.

[42] M. S. I. Mamun and A. Miyaji. Secure VANET applications with a refined group signature. In *PST*, pages 199–206, 2014.

[43] S. Meiklejohn, K. Mowery, S. Checkoway, and H. Shacham. The phantom tollbooth: Privacy-preserving electronic toll collection in the presence of driver collusion. In *USENIX Security Symposium*, 2011.

[44] T. Nakanishi, T. Fujiwara, and H. Watanabe. A linkable group signature and its application to secret voting. *JIP*, 40(7):3085–3096, 1999.

[45] T. Nakanishi and N. Funabiki. Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps. In *ASIACRYPT*, pages 533–548, 2005.

[46] T. Nakanishi and N. Funabiki. A short verifier-local revocation group signature scheme with backward unlinkability. In *IWSEC*, pages 17–32, 2006.

[47] D. Pointcheval and O. Sanders. Short randomizable signatures. In *CT-RSA*, pages 111–126, 2016.

[48] I. Rouf, R. D. Miller, H. A. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar. Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In *USENIX Security Symposium*, pages 323–338, 2010.

[49] Y. Sun, R. Lu, X. Lin, X. Shen, and J. Su. An efficient pseudonymous authentication scheme with strong privacy preservation for vehicular communications. *IEEE T. Vehicular Technology*, 59(7):3589–3603, 2010.

[50] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[51] S. Tillich and M. Wójcik. Security analysis of an open car immobilizer protocol stack. In *INTRUST*, pages 83–94, 2012.

[52] J. Wetzels. Broken keys to the kingdom: Security and privacy aspects of RFID-based car keys. *CoRR*, abs/1405.7424, 2014.

[53] W. Whyte, A. Weimerskirch, V. Kumar, and T. Hehn. A security credential management system for V2V communications. In *IEEE Vehicular Networking Conference*, pages 1–8, 2013.

[54] Q. Wu, J. Domingo-Ferrer, and Ú. González-Nicolás. Balanced trustworthiness, safety, and privacy in vehicle-to-vehicle communications. *IEEE T. Vehicular Technology*, 59(2):559–573, 2010.

[55] Y. Xi, W. Shi, and L. Schwiebert. Mobile anonymity of dynamic groups in vehicular networks. *Security and Communication Networks*, 1(3):219–231, 2008.

[56] M. Xu, W. Xu, J. Walker, and B. Moore. Lightweight secure communication protocols for in-vehicle sensor networks. In *CyCAR*, pages 19–30, 2013.

Appendix: Security Analysis of Our GS-TDL

Proof of Theorem 4.1

Proof. We define the following two games: Game 0 is the same as $\mathrm{Exp}^{anon\text{-}tg\text{-}b}_{\mathsf{GS\text{-}TDL},\mathcal{A}}(\lambda)$. Game 1 is the same as Game 0, except τ^* contained in σ^* is randomly chosen, and σ^* is generated by the simulation of NIZK. Here, we show that there exist an algorithm $\mathcal{B}$ that breaks the SDDHI problem by using $\mathcal{A}$ as follows.

Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ be a bilinear group, and (g_1, g_2, g_1^x) is an instance of the SDDHI problem. Let q be the number of AddU queries. $\mathcal{B}$ chooses $i^* \in [1, q]$ and set x is a part of signing key of the user. $\mathcal{B}$ chooses $\gamma \xleftarrow{\$} \mathbb{Z}_p$, and $h \xleftarrow{\$} \mathbb{G}_1$, computes $W = g_2^\gamma$, and sets $\mathsf{gpk} = (params, h, W, e(g_1, g_2), e(h, W), e(h, g_2), H)$, where $H : \{0,1\}^* \to \mathbb{Z}_p$ is a hash function modeled as a random oracle. $\mathcal{B}$ also runs $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{Gen}(1^\lambda)$, and sets $\mathsf{tpk} := \mathsf{vk}$ and $\mathsf{tsk} := \mathsf{sigk}$. $\mathcal{B}$ sends gpk, tpk, and tsk to $\mathcal{A}$.

In the i-th AddU query (with input ID), where $i \neq i^*$, $\mathcal{B}$ chooses $x, y \xleftarrow{\$} \mathbb{Z}_p$, computes $A = (g_1 h^{-y})^{\frac{1}{\gamma+x}}$, sets $\mathsf{sigk}_{\mathsf{ID}} = (x, y, A)$, and adds ID to GU. In the i^*-th AddU query (with input ID^*), $\mathcal{B}$ adds ID^* to GU.

For a GSign query with input (ID, t_T, M), if $\mathsf{ID} \notin \mathsf{GU}$, then $\mathcal{B}$ runs the simulation of the AddU oracle. If $\mathsf{ID} \neq \mathsf{ID}^*$, then $\mathcal{B}$ computes a group signature σ as in the actual GSign algorithm, returns σ to $\mathcal{A}$, and adds (ID, T) to STSet. Let $\mathsf{ID} = \mathsf{ID}^*$. $\mathcal{B}$ sends T to $\mathcal{O}_x$, and obtains $\tau = g_1^{\frac{1}{x+T}}$. $\mathcal{B}$ chooses $s_x, s_\delta, s_\beta, c \xleftarrow{\$} \mathbb{Z}_p$ and $C \xleftarrow{\$} \mathbb{G}_1$, computes $R_1 = \frac{e(h,g_2)^{s_\delta} e(h,W)^{s_\beta}}{e(C,g_2)^{s_x}} \big(\frac{e(C,W)}{e(g_1,g_2)}\big)^{-c}$ and $R_2 = e(\tau, g_2)^{s_x} \big(\frac{e(g_1,g_2)}{e(\tau,W_T)}\big)^{-c}$, and patches H such that $c := H(\mathsf{gpk}, \mathsf{tpk}, C, \tau, R_1, R_2, M)$. $\mathcal{B}$ returns $\sigma = (C, \tau, c, s_x, s_\delta, s_\beta)$ to $\mathcal{A}$.

In the challenge phase, $\mathcal{A}$ sends $(\mathsf{ID}_{i_0}, \mathsf{ID}_{i_1}, t_{T_0}, t_{T_1}, M_0^*, M_1^*)$ to $\mathcal{B}$. $\mathcal{B}$ chooses $b \xleftarrow{\$} \{0,1\}$. If $\mathsf{ID}_{i_b} \neq \mathsf{ID}^*$, then $\mathcal{B}$ aborts. Let $\mathsf{ID}_{i_b} = \mathsf{ID}^*$ (this holds with the probability at least $1/q$). Next, we consider the following two cases:

- $T_0 = T_1$: Let (T, W_T) be contained in both t_{T_0} and t_{T_1}. $\mathcal{B}$ sends $T := T_0 = T_1$ to the challenger of the SDDHI problem, and obtains τ^*. We remark that T was not sent to $\mathcal{O}_x$. $\mathcal{B}$ chooses $s_x^*, s_\delta^*, s_\beta^*, c^* \xleftarrow{\$} \mathbb{Z}_p$ and $C^* \xleftarrow{\$} \mathbb{G}_1$, computes $R_1^* = \frac{e(h,g_2)^{s_\delta^*} e(h,W)^{s_\beta^*}}{e(C,g_2)^{s_x^*}} \big(\frac{e(C,W)}{e(g_1,g_2)}\big)^{-c^*}$ and $R_2 = e(\tau^*, g_2)^{s_x^*} \big(\frac{e(g_1,g_2)}{e(\tau^*,W_T)}\big)^{-c^*}$, and patches H such that $c^* := H(\mathsf{gpk}, \mathsf{tpk}, C^*, \tau^*, R_1^*, R_2^*, M^*)$. $\mathcal{B}$ returns $\sigma^* = (C^*, \tau^*, c^*, s_x^*, s_\delta^*, s_\beta^*)$ to $\mathcal{A}$.
- $T_0 \neq T_1$: Let (T_0, W_{T_0}) and (T_1, W_{T_1}) be contained in t_{T_0} and t_{T_1}, respectively. $\mathcal{B}$ sends T_0 to $\mathcal{O}_x$, and obtains $\tau_0^* = g_1^{\frac{1}{x+T_0}}$. $\mathcal{B}$ chooses $s_{x,0}^*, s_{\delta,0}^*, s_{\beta,0}^*, c_0^* \xleftarrow{\$} \mathbb{Z}_p$ and $C_0^* \xleftarrow{\$} \mathbb{G}_1$, computes $R_{1,0}^* = \frac{e(h,g_2)^{s_{\delta,0}^*} e(h,W)^{s_{\beta,0}^*}}{e(C_0^*,g_2)^{s_{x,0}^*}} \big(\frac{e(C_0^*,W)}{e(g_1,g_2)}\big)^{-c_0^*}$ and $R_{2,0}^* = e(\tau_0^*, g_2)^{s_{x,0}^*} \big(\frac{e(g_1,g_2)}{e(\tau_0^*,W_{T_0})}\big)^{-c_0^*}$, and patches H such that $c_0^* := H(\mathsf{gpk}, \mathsf{tpk}, C_0^*, \tau_0^*, R_{1,0}^*, R_{2,0}^*, M_0^*)$. Moreover, $\mathcal{B}$ sends T_1 to the challenger of the SDDHI problem, and obtains τ_1^*. We remark that T_1 was not sent to $\mathcal{O}_x$. $\mathcal{B}$ chooses $s_{x,1}^*, s_{\delta,1}^*, s_{\beta,1}^*, c_1^* \xleftarrow{\$} \mathbb{Z}_p$ and $C_1^* \xleftarrow{\$} \mathbb{G}_1$, computes $R_{1,1}^* = \frac{e(h,g_2)^{s_{\delta,1}^*} e(h,W)^{s_{\beta,1}^*}}{e(C_1^*,g_2)^{s_{x,1}^*}} \big(\frac{e(C_1^*,W)}{e(g_1,g_2)}\big)^{-c_1^*}$ and $R_{2,1}^* = e(\tau_1^*, g_2)^{s_{x,1}^*} \big(\frac{e(g_1,g_2)}{e(\tau_1^*,W_{T_1})}\big)^{-c_1^*}$, and patches H such that $c_1^* := H(\mathsf{gpk}, \mathsf{tpk}, C_1^*, \tau_1^*, R_{1,1}^*, R_{2,1}^*, M_1^*)$. $\mathcal{B}$ returns $\sigma_0^* = (C_0^*, \tau_0^*, c_0^*, s_{x,0}^*, s_{\delta,0}^*, s_{\beta,0}^*)$ and $\sigma_1^* = (C_1^*, \tau_1^*, c_1^*, s_{x,1}^*, s_{\delta,1}^*, s_{\beta,1}^*)$ to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs b'. If $\tau^* = g^{\frac{1}{x+T}}$ (or $\tau_1^* = g^{\frac{1}{x+T_1}}$), then $\mathcal{B}$ simulates Game 0, and if τ^* (or τ_1^*) is a random value, then $\mathcal{B}$ simulates Game 1. In Game 1, no information of the challenge bit b is revealed from σ^*, σ_0^*, and σ_1^*. So, $\mathcal{B}$ decides the challenge is a random value if $b' \neq b$, and it is not a random value, otherwise, and solves the SDDHI problem. We remark that $\mathcal{B}$ can revoke ID^* at a time $T' > T$ (or $T' > T_1$) using the $\mathcal{O}_x$ oracle. This concludes the proof. □

Proof of Theorem 4.2

Proof. We consider the following two cases. The first one is $\mathcal{A}$ produces a valid signature although $\mathcal{A}$ does not have t_T ((1) $\wedge$ (2) $\wedge$ (3) in the definition), and the second one is $\mathcal{A}$ produces a valid signature although $\mathcal{A}$ does not have a signing key ((1) $\wedge$ (2) $\wedge$ (4) in the definition).

First Case: We construct an algorithm $\mathcal{B}$ that breaks EUF-CMA security of the underlying signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$. The challenger of the signature scheme runs $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{Gen}(1^\lambda)$, and sends vk to $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{tpk} := \mathsf{vk}$, runs $params \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{gpk}, \mathsf{gsk}) \leftarrow \mathsf{GKeyGen}(params)$, and sends $(\mathsf{gpk}, \mathsf{tpk})$ to $\mathcal{A}$. For a $\mathsf{TokenGen}$ query T, $\mathcal{B}$ computes $W_T = g_2^T$, sends W_T to the challenger as a signing query, and obtains Σ. $\mathcal{B}$ sets $t_T = (T, W_T, \Sigma)$, and sends t_T to $\mathcal{A}$. Since $\mathcal{B}$ has gsk, $\mathcal{B}$ can respond all AddU, GSign, and USK queries. We remark that $\mathcal{A}$ does not access the TSK oracle. Finally, $\mathcal{A}$ outputs (T, M, σ). Since σ is a valid group signature, there exist (Σ, W_T) such that W_T is used in the verification algorithm, and Σ is a valid signature under vk. That is, $\mathcal{A}$ produces $t_T = (T, W_T, \Sigma)$, and sets it via the $\mathsf{SetToken}$ oracle. Since W_T is not sent to $\mathcal{B}$ as a $\mathsf{TokenGen}$ query, $\mathcal{B}$ outputs (Σ, W_T) as a forgery of the signature scheme.

Second Case: We construct an algorithm $\mathcal{B}$ that breaks the q-SDH problem as follows. Let $(g_1, g_1^\gamma, \ldots, g_1^{\gamma^q}, g_2, g_2^\gamma)$ be an SDH instance. Here, q be the number of AddU queries. $\mathcal{B}$ runs $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{Gen}(1^\lambda)$, and sets $\mathsf{tpk} := \mathsf{vk}$. $\mathcal{B}$ chooses $x_1, \ldots, x_q, y_1, \ldots, y_q \overset{\$}{\leftarrow} \mathbb{Z}_p$ and $\alpha, \theta \overset{\$}{\leftarrow} \mathbb{Z}_p^*$. Let us define $f(X) = \prod_{i=1}^q (X + x_q) := \sum_{i=0}^q \alpha_i X^i$ and $f_i(X) := f(X)/(X - x_i) = \prod_{j=1, j\neq i}^q (X + x_i) := \sum_{i=1}^{q-1} \beta_i X^i$, and set $g_1' = (\prod_{i=0}^q (g_1^{\gamma^i})_i^\alpha)^\theta = g_1^{\theta f(\gamma)}$. Then, for each $i \in [1, q]$ $(\prod_{j=0}^{q-1} (g_1^{\gamma^i})^{\beta_i})^\theta = g_1^{\theta f_i(\gamma)} = g_1'^{\frac{1}{\gamma + x_i}}$ hold. Set $h := g_1'^\alpha$. For each $i \in [1, q]$, $\mathcal{B}$ computes $A_i := (g_1'^{\frac{1}{\gamma + x_i}})^{1 - y_i \alpha} = (g_1' h^{-y})^{\frac{1}{\gamma + x_i}}$. $\mathcal{B}$ sets $W := g_2^\gamma$, $params = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1', g_2)$, and $\mathsf{gpk} = (params, h, W, e(g_1', g_2), e(h, W), e(h, g_2), H)$, and gives $(\mathsf{gpk}, \mathsf{tpk})$ to $\mathcal{A}$.

For an AddU query, $\mathcal{B}$ chooses unselected $x \in \{x_1, \ldots, x_q\}$ and sets the corresponding (x, y, A) as the signing key. Since $\mathcal{B}$ has tsk, $\mathcal{B}$ can respond $\mathsf{TokenGen}$ and TSK queries. Moreover, $\mathcal{B}$ can respond GSign and Revoke queries since $\mathcal{B}$ has all signing keys (x_i, y_i, A_i) for each $i \in [1, q]$.

Finally, $\mathcal{A}$ outputs a forge group signature $\sigma = (C, \tau, c, s_x, s_\delta, s_\beta)$. $\mathcal{B}$ rewinds $\mathcal{A}$ and obtains $\sigma' = (C, \tau, c', s_x', s_\delta', s_\beta')$ where $c \neq c'$ with non-negligible probability (due to the forking lemma). Set $\tilde{x} := \frac{s_x - s_x'}{c - c'}, \tilde{y} := \frac{(s_x - s_x')(s_\beta - s_\beta') - (s_\delta - s_\delta')(c - c')}{(c - c')^2}$, and $\tilde{\beta} := \frac{s_\beta - s_\beta'}{c - c'}$. Then, $\frac{e(C, W)}{e(g_1', g_2)} = \frac{e(h, g_2)^{\tilde{\beta}\tilde{x} - \tilde{y}} e(h, W)^{\tilde{\beta}}}{e(C, g_2)^{\tilde{x}}}$ and $e(\tau, g_2)^{\tilde{x}} = \frac{e(g_1', g_2)}{e(\tau, W_T)}$ hold. That is, $(\tilde{x}, \tilde{y}, \tilde{A})$ can be extracted. If $1 - \tilde{y}\alpha = 0$, then $\mathcal{B}$ aborts. Moreover, if $\tilde{x} \in \{x_1, \ldots, x_q\}$, then $\mathcal{B}$ aborts. Since α and all x are randomly chosen, the aborting probability is at most q/p, and is negligible. From now on, we assume that $1 - \tilde{y}\alpha \neq 0$ and $\tilde{x} \notin \{x_1, \ldots, x_q\}$. Since $\tilde{A}$ can be represented as $\tilde{A} = (g_1' h^{-\tilde{y}})^{\frac{1}{\gamma + \tilde{x}}}$, $\mathcal{B}$ can compute $\tilde{A}^{\frac{1}{1 - \tilde{y}\alpha}} = g_1'^{\frac{1}{\gamma + \tilde{x}}} = (g_1^{\theta f(\gamma)})^{\frac{1}{\gamma + \tilde{x}}}$. Next, $\mathcal{B}$ computes $F(X)$ and $\gamma_* \in \mathbb{Z}_p^*$ which satisfy $f(X) = (X + \tilde{x})F(X) + \gamma_*$. Finally, $\mathcal{B}$ computes $\left(((g_1^{\theta f(\gamma)})^{\frac{1}{\gamma + \tilde{x}}})^{\frac{1}{\theta}} \prod_{i=0}^{q-1} (g_1^{x^i})^{-F_i}\right)^{\frac{1}{\gamma_*}} = g_1^{\frac{1}{\gamma + \tilde{x}}}$, where $F(X) := \sum_{i=0}^{q-1} F_i X^i$, and outputs $(\tilde{x}, g_1^{\frac{1}{\gamma + \tilde{x}}})$ as a solution of the SDH problem.

$\square$

Proof of Theorem 4.3

Proof. Let $(\mathsf{ID}_0, \mathsf{ID}_1, T_0, T_1, M)$ and (M^*, σ^*) be the output of $\mathcal{A}$, where $(\mathsf{ID}_0, T_0) \neq (\mathsf{ID}_1, T_1)$. Let x_0 be contained in $\mathsf{sigk}_{\mathsf{ID}_0}$ and x_1 be contained in $\mathsf{sigk}_{\mathsf{ID}_1}$, respectively. If $\mathsf{Link}(\mathsf{gpk}, \mathsf{tpk}, \mathsf{RL}_T, (M, \sigma_0, T_0), (M^*, \sigma^*, T_1)) = 1$, then $g_1^{\frac{1}{x_0 + T_0}} = g_1^{\frac{1}{x_1 + T_1}}$ and $T = T_0 = T_1$ holds. Then, $x_0 = x_1$ holds. Since x_0 and x_1 are randomly chosen, this equation holds with probability at most $1/p$. This concludes the proof. $\square$