Supplementary Material for A Self-Generating Pseudonym-Based Certificateless Conditional Privacy Preservation Authentication Scheme for IoV

Qibang Zhan, Ming Luo, Minrong Qiu

The organization of this supplementary material is as follows: Section I provides a detailed review of the certificateless conditional privacy-preserving authentication scheme proposed by Genc et al. [1], followed by a comprehensive security analysis of their scheme, including forgery attacks, active attacks initiated by the Key Generation Unit (KGU), and persistent attacks by revoked users. Section II presents the full correctness verification of our proposed scheme, covering single verification, batch verification, malicious-user traceability, and the correctness derivation for detecting active attacks launched by the Key Generation Center (KGC). Section III formalizes the main security requirements of our SGPBC-CPPA scheme, including anonymity, unlinkability, and replay-attack resistance. This section provides the formal game-based definitions of these properties and presents the corresponding security proofs.

I. Review and Security Analysis of Genc et al.'s Scheme

In this section, we first provide a concise review of the certificateless conditional privacy-preserving authentication scheme proposed by Genc et al. [1]. The scheme defines an IoV architecture composed of a Central Unit (CU), KGU, Road-Side Units (RSUs), and vehicles. For clarity, Table I summarizes the notations used in their scheme.

After presenting the scheme, we conduct a detailed security analysis and demonstrate that the scheme remains vulnerable to several attacks, including forgery attacks, active attacks by KGU, and persistent attacks by identified malicious users.

A. Review of Genc et al.'s Conditional Privacy Preserving Authentication Scheme

The scheme proposed by Genc et al. [1] encompasses these five phases:

1) Setup phase:
The CU and KGU collaboratively determine the parameters: P, p, q, E, and opt for four secure hash functions: $h_1 : G \times \{0,1\}^* \to Z_q^*$, $h_2 : G \to Z_q^*$, $h_3 : G \to Z_q^*$, $h_4 : \{0,1\}^{\times}\{0,1\}^* \times G \times G \times \{0,1\}^* \to Z_q^*$. Both CU and KGU independently select secure values $c, k \in Z_q^*$ and compute their respective public keys as $Cpub = cP$ and $Kpub = kP$. Subsequently, CU and KGU jointly release the public system parameters $\{E, P, p, q, Kpub, C_{pub}, h_3, h_4\}$.
2) Register phase:
The vehicle or RSU undergoes registration with its authentic identity UID_i on the CU. The CU employs its private key c to compute $VRA_{id,i} = UID_i \oplus h_1(C_{pub}||c)$, subsequently loading $VRA_{id,i}$ into the TPD of the respective vehicle or RSU.
3) Partial anonymous identity and Partial Private Key generation phase:
The vehicle or RSU transmits its anonymous identity $VRA_{id,i}$ to the KGU, which proceeds to unlock it within its database, excluding penalized or unregistered vehicles or RSUs. Upon confirming the existence of the anonymous identity, KGU utilizes its private key k to compute the partial anonymous identity as $VRPA_{id,i} = VRA_{id,i} \oplus h_2(k \cdot C_{pub})$ for the respective vehicle or RSU. Subsequently, a random value $w \in Z_q^*$ is chosen to calculate the partial private key

TABLE I: Notations Description

Notation	Description
p,q	Two large prime numbers.
E/F_p	An elliptic curve over finite fields
G	The addition cycle group
VR_i	The i^{th} vehicle or RSU
$VRA_{id,i}$	Anonymous identity of the i^{th} vehicle or RSU
$VRPA_{id,i}$	Partial anonymous identity of the i^{th} vehicle or RSU
$VRTA_{id,i}$	Temporary anonymous identity of the i^{th} vehicle or RSU
$VR_{private,i}$	The private key of the i^{th} vehicle or RSU
c	Secret key of the CU
C_{pub}	Public key of the CU
k	Secret key of the KGU
K_{pub}	Public key of the KGU
$\oplus$	XOR Operation
$\|\|$	Concatenation Operation
T	The current timestamp
PPK	Partial private key
UID_i	The real identity of the i^{th} vehicle or RSU
$VR_{ppk,i}$	Partial private key of the i^{th} vehicle or RSU
$VR_{public,i}$	The public key of the i^{th} vehicle or RSU
$Ts_{vr,i}$	The timestamp of the i^{th} vehicle or RSU
$t_{vr,i}$	The i^{th} vehicle or RSU randomly selected private key
θ_{vr_i}	The signature of the i^{th} vehicle or RSU

This work was supported in part by the National Natural Science Foundation of China under Grant 62262041, in part by the Natural Science Foundation of Jiangxi Province under Grant 20252BAC250129, and in part by the Graduate Innovation Special Fund of Jiangxi Province under Grant YC2023-S013. (Corresponding author: Ming Luo.)

Qibang Zhan and Ming Luo are with the School of Software, Nanchang University, Nanchang, Jiangxi 330047, China (e-mail: zhanqibang@qq.com; lmhappy21@163.com).

Minrong Qiu is with the Gongqing Institute of Science and Technology, Jiujiang, Jiangxi 332020, China (e-mail: 13576203266@qq.com).

$VRppk,i = k+w\cdot h_1(VRPA_{id,i}||w||K_{pub})$. In the final step, KGU securely communicates $VRPA_{id,i}, VR_{ppk,i}$ to the vehicle or RSU through a protected channel.

4) Signing phase:
When a vehicle or RSU necessitates communication with other entities, it initiates the process by randomly selecting a value, denoted as $t_{vr,i} \in Z_q^*$. Subsequently, it computes its private key as $VRprivate,i = VR_{ppk,i} + t_{vr,i}$, the public key as $VR_{public,i} = VR_{private,i} \cdot P - K_{pub}$, and the anonymous identity as $VRTA_{id,i} = VRPA_{id,i} \cdot h_3(t_{vr,i} \cdot C_{pub})$. Further calculations involve determining $\phi_{vr,i} = t_{vr,i} \cdot P$ and $S_{vr,i} = h_4(m_{vr,i}||VRTA_{id,i}||VR_{public,i}||\phi_{vr,i}||Ts_{vr,i})$. Ultimately, the signature $\theta_{vr,i}$ is computed as $VR_{private,i} + t_{vr,i}S_{vr,i} \pmod q$. The resulting parameters, including $\{\theta_{vr,i}, VRTA_{vr,i}, VR_{public,i}, \phi_{vr,i}, Ts_{vr,i}\}$, are transmitted to the designated target user.

5) Verification phase:
When a vehicle or RSU receives a message, it first checks whether the timestamp $Ts_{vr,i}$ of the message is within a valid range. If it is not, the message is directly discarded. On the contrary, if the timestamp is valid, $S_{vr,i} = h_4(m_{vr,i}||VRTA_{id,i}||VR_{public,i}||\phi_{vr,i}||Ts_{vr,i})$ is calculated. Then, it checks the equation $\theta_{vr,i} \cdot P \overset{?}{=} K_{pub} + V_{public,i} + S_{vr,i} \cdot \phi_{vr,i}$. If the equation holds, the message is received; otherwise, it is discarded.

B. Security Analysis of Genc et al.'s Conditional Privacy Preserving Authentication Scheme

We will point out three flaws in the scheme proposed by Genc et al. [1].

1) Vulnerable to forgery attacks:

a) Adversary Model:
We consider an external adversary $\mathcal{A}$ who is not registered with the Central Unit and therefore does not possess any partial keys issued by the KGU. The adversary's objective is to forge a valid authentication message.

b) Attack procedure:
$\mathcal{A}$ randomly chooses $VRPA_{id,j}, t_{vr,j}$, and $w \in Z_q^*$, and then constructs $VRTA_{id,j} = VRPA_{id,j} \cdot h_3(t_{vr,j} \cdot C_{pub})$, $\phi_{vr,j} = t_{vr,j} \cdot P$, $VR_{private} = w$, and $VR_{public} = wP - K_{pub}$. $\mathcal{A}$ computes $S_{vr,j} = h_4(m_{vr,j} \parallel VRTA_{id,j} \parallel VR_{public,j} \parallel \phi_{vr,j} \parallel Ts_{vr,j})$ and outputs $\theta_{vr,j} = w + t_{vr,j} \cdot S_{vr,j}$. The forged message $\{m_{vr,j}, \theta_{vr,j}, VRTA_{id,j}, VR_{public,j}, Ts_{vr,j}\}$ is then broadcast.

c) Verification:
Honest users execute the original verification equation: $\theta_{vr,j} \cdot P = K_{pub} + VR_{public,j} + S_{vr,j}.\phi_{vr,j}$. Substituting the forged values gives $\theta_{vr,j} \cdot P = w \cdot P + t_{vr,j} \cdot S_{vr,j} \cdot P$. which satisfies the verification equation without requiring any legitimate secret material.

d) Reason for success
In a secure CLC scheme, the verification of a signature must rely on the KGU's master public key to ensure its validity. However, in the scheme proposed by Genc et al. [1], an external adversary can eliminate the contribution of the KGU's master public key during the signature generation process. As a result, an attacker whose private key does not embed any secret information from the KGU can still produce signatures that successfully pass verification. This design flaw renders the scheme vulnerable to forgery attacks.

e) Impact:
This attack enables an adversary to impersonate legitimate vehicles and inject falsified traffic messages, thereby undermining the security and reliability of the IoV system.

2) Vulnerable to active attacks by KGU:

a) Adversary Model:
We consider a malicious KGU. The KGU is responsible for generating the partial anonymous identities and the corresponding partial private keys for all legitimate vehicles, and therefore possesses complete knowledge of these partial credentials. Unlike an external adversary, the KGU has full control over the key generation process, enabling it to forge the identities of any legitimate vehicle and generate valid authentication messages. Its objective is to impersonate legitimate vehicles within the system and inject forged messages.

b) Attack procedure:
The malicious KGU randomly selects a partial anonymous identity $VRPA_{id,j}$ and the corresponding partial private key $VR_{ppk,j}$ from the database. Using these partial keys together with the system's public parameters, the KGU can reconstruct the full private key in the same manner as a legitimate user. The KGU then prepares an arbitrary message $m_{vr,j}$ and generates the corresponding signature $\{\theta_{vr,j},\ VRTA_{id,j},\ VR_{public,j},\ Ts_{vr,j}\}$ Finally, the KGU broadcasts this forged authentication message into the network, where other vehicles and RSUs will accept it as a valid message.

c) Reason for success:
In this scheme, the KGU is assumed to be fully trustworthy. However, since the KGU holds all partial anonymous identities and partial private keys of legitimate users, a malicious KGU would pose a severe threat to the security of the entire system. This assumption is therefore overly idealized. Moreover, because the scheme treats the KGU as entirely trustworthy, it does not incorporate any mechanism for detecting malicious behavior from the KGU.

d) Impact:
An active attack by the KGU allows it to impersonate any legitimate vehicle and inject falsified

messages, thereby undermining the overall security and trust of the IoV system.

3) Vulnerable to persistent attacks by identified malicious users:
   a) Adversary Model:
   We consider a previously identified malicious user VR_m, whose real identity RID_m has been revealed by the CU. The CU has removed VR_m from the IoV network, and the KGU will no longer generate any partial anonymous identities or partial private keys for this user. Nevertheless, VR_m still retains the partial anonymous identity and partial private key that were legitimately issued by the KGU before its removal. The adversary's objective is to continue generating valid pseudonyms and authentication messages even after being formally removed from the system, thereby re-entering IoV communications.
   b) Attack procedure
   Before being revoked, the malicious user VR_m obtains its partial anonymous identity and partial private key from the KGU. After revocation, VR_m retains these partial keys and reconstructs the full private key in the same manner as a legitimate vehicle. With the reconstructed private key, VR_m can continue generating new temporary pseudonyms, public keys, and valid signatures. Using these newly generated pseudonyms, VR_m broadcasts forged authentication messages to the network, making them appear unrelated to its previous malicious activities. Since the verification algorithm remains unchanged, other vehicles and RSUs will incorrectly accept these messages as legitimate.
   c) Reason for success
   The fundamental reason behind this vulnerability is that the scheme relies on previously issued partial anonymous identities and partial private keys to generate new temporary pseudonyms and full private keys, without assigning any validity period to these keys. As a result, these partial keys do not expire after the user is revoked, nor do they require renewal from the KGU. Consequently, once an attacker obtains these partial keys, they can independently generate an unlimited number of new temporary pseudonyms and public keys for producing valid authentication messages.
   d) Impact:
   This persistent attack enables a revoked malicious user to continuously generate valid pseudonyms and inject forged messages, effectively bypassing the revocation mechanism and compromising the long-term security, reliability, and trust of the IoV system.

II. Verification of correctness

In this section, we verify that the proposed scheme satisfies the correctness requirements of single verification, batch verification, traceability of malicious users, and the detection of active attacks initiated by the KGC. To facilitate the derivation of the subsequent equations, we first summarize the key parameters and intermediate values involved in the correctness verification. The main definitions are listed in Table II.

TABLE II: Key Parameter Used in Correctness Verification

Description	**Expression**
First public key of KGC	$K_{1,pub} = k_1 \cdot P$
Second public key of KGC	$K_{2,pub} = k_2 \cdot P$
First partial private key of PP_i	$ppk_{1_i} = y_i + h_{2i} \cdot k_1$
Second partial private key of PP_i	$ppk_{2_i} = y_i + h_{2i} \cdot k_2$
Partial public key of PP_i	$PPK_i = y_i \cdot P$
Temporary secret value of TP_i	$tsv_i = x_i \cdot v_i$
First temporary partial private key of TP_i	$tppk_{1_i} = v_i + ppk_{1_i}$
Second temporary partial private key of TP_i	$tppk_{2_i} = v_i + ppk_{2_i}$
First temporary partial public key of TP_i	$TPPK_{1,i} = x_i \cdot v_i \cdot P$
Second temporary partial public key of TP_i	$TPPK_{2_i} = (v_i + y_i) \cdot P$
Message signature	$\theta = (tsv_i + tppk_{1_i}) \cdot h_{4i} + tppk_{2_i}$
Primary pseudonym of the user RID_i	$PP_i = TP_i \oplus H_3(x_i \cdot v_i \cdot T_{pub})$

A. Correctness of Single Verification

The single verification algorithm ensures that the signature θ' generated by a legitimate user can be successfully validated using the corresponding public information. The core of the verification lies in checking whether the following equation holds:

$$\begin{aligned}\theta' \cdot P =&(TPPK_{1i} + TPPK_{2i} + h_{2i} \cdot K_{1pub}) \cdot h_{4i} + \\ &TPPK_{2i} + h_{2i} \cdot K_{2pub}\end{aligned}$$

Fundamentally, this procedure is equivalent to verifying whether

$$\theta' \cdot P = \theta \cdot P$$

where θ' denotes the signature computed by the sender and θ represents the reconstructed signature derived from the corresponding public parameters. To make the derivation clear, we now present the correctness proof step by step.

1) By the definition of the verification algorithm.

$$\begin{aligned}\theta' \cdot P =&(TPPK_{1i} + TPPK_{2i} + h_{2i} \cdot K_{1pub}) \cdot h_{4i} + \\ &TPPK_{2i} + h_{2i} \cdot K_{2pub}\end{aligned}$$

2) Substituting the definitions of $TPPK_{1i}$, $TPPK_{2i}$, K_{1pub} and K_{2pub}.

$$\begin{aligned}\theta' \cdot P =&(x_i \cdot v_i \cdot P + (v_i + y_i) \cdot P + h_{2i} \cdot k_1 \cdot P) \cdot h_{4i} \\ &+ (v_i + y_i) \cdot P + h_{2i} \cdot k_2 \cdot P\end{aligned}$$

3) Applying scalar distributivity over the elliptic curve group.

$$\begin{aligned}\theta' \cdot P =&((x_i \cdot v_i + v_i + y_i + h_{2i} \cdot k_1) \cdot h_{4i} + v_i + y_i + \\ &h_{2i} \cdot k_2) \cdot P\end{aligned}$$

4) Using the definitions $tsv_i = x_i \cdot v_i$, $ppk_{1_i} = y_i + h_{2i} \cdot k_1$, $ppk_{2_i} = y_i + h_{2i} \cdot k_2$.

$$\theta' \cdot P =((tsv_i + v_i + ppk_{1_i}) \cdot h_{4i} + v_i + ppk_{2_i}) \cdot P$$

5) Using $tppk_{1_i} = v_i + ppk_{1_i}$ and $tppk_{2_i} = v_i + ppk_{2_i}$.

$$\theta' \cdot P =((tsv_i + tppk_{1_i}) \cdot h_{4i} + tppk_{2_i}) \cdot P$$

6) Using $\theta = (tsv_i + tppk_{1_i}) \cdot h_{4i} + tppk_{2_i}$.

$$\theta' \cdot P =\theta \cdot P$$

Since the two expressions match, the signature θ' is valid, which confirms the correctness of the single verification algorithm.

B. Correctness of Batch Verification

The batch verification algorithm ensures that a set of signatures $\{\theta'_i\}_{i=1}^n$ generated by legitimate users can be successfully validated using the corresponding public information. The core of the verification lies in checking whether the following aggregated equation holds:

$$\begin{aligned}\sum_{i=1}^{n}(\theta'_i) \cdot P =& \sum_{i=1}^{n}((TPPK_{1i} + TPPK_{2i}) \cdot h_{4i}) + \\ & \sum_{i=1}^{n}(h_{2i} \cdot h_{4i}) \cdot K_{1pub} + \sum_{i=1}^{n}(TPPK_{2i}) + \\ & \sum_{i=1}^{n}(h_{2i}) \cdot K_{2pub}\end{aligned}$$

Fundamentally, this procedure is equivalent to verifying whether

$$\sum_{i=1}^{n}(\theta'_i) \cdot P = \sum_{i=1}^{n}(\theta_i) \cdot P,$$

where each θ'_i denotes the signature computed by the sender and each θ_i represents the reconstructed signature derived from the corresponding public parameters. To make the derivation clear, we now present the correctness proof step by step.

1) By the definition of the batch verification algorithm.

$$\begin{aligned}\sum_{i=1}^{n}(\theta'_i) \cdot P =& \sum_{i=1}^{n}((TPPK_{1i} + TPPK_{2i}) \cdot h_{4i}) + \\ & \sum_{i=1}^{n}(h_{2i} \cdot h_{4i}) \cdot K_{1pub} + \sum_{i=1}^{n}(TPPK_{2i}) + \\ & \sum_{i=1}^{n}(h_{2i}) \cdot K_{2pub}\end{aligned}$$

2) Substituting the definitions of $TPPK_{1i}$, $TPPK_{2i}$, K_{1pub} and K_{2pub}.

$$\begin{aligned}\sum_{i=1}^{n}(\theta'_i) \cdot P =& \sum_{i=1}^{n}((x_i \cdot v_i \cdot P + (v_i + y_i) \cdot P) \cdot h_{4i}) + \\ & \sum_{i=1}^{n}(h_{4i} \cdot h_{2i} \cdot k_1) \cdot P + \sum_{i=1}^{n}((v_i + y_i) \cdot P) + \\ & \sum_{i=1}^{n}(h_{2i} \cdot k_2) \cdot P.\end{aligned}$$

3) Applying scalar distributivity over the elliptic curve group.

$$\begin{aligned}\sum_{i=1}^{n}(\theta'_i) \cdot P =& \sum_{i=1}^{n}((x_i \cdot v_i + v_i + y_i + h_{2i} \cdot k_1) \cdot h_{4i} + v_i \\ & + y_i + h_{2i} \cdot k_2) \cdot P\end{aligned}$$

4) Using the definitions $tsv_i = x_i \cdot v_i$, $ppk_{1_i} = y_i + h_{2i} \cdot k_1$, $ppk_{2_i} = y_i + h_{2i} \cdot k_2$.

$$\begin{aligned}\sum_{i=1}^{n}(\theta'_i) \cdot P =& \sum_{i=1}^{n}((tsv_i + v_i + ppk_{1_i}) \cdot h_{4i} + v_i + \\ & ppk_{2_i}) \cdot P\end{aligned}$$

5) Using $tppk_{1_i} = v_i + ppk_{1_i}$ and $tppk_{2_i} = v_i + ppk_{2_i}$.

$$\sum_{i=1}^{n}(\theta'_i) \cdot P = \sum_{i=1}^{n}((tsv_i + tppk_{1_i}) \cdot h_{4i} + tppk_{2_i}) \cdot P.$$

6) Using the definition $\theta_i = (tsv_i + tppk_{1_i}) \cdot h_{4i} + tppk_{2_i}$.

$$\sum_{i=1}^{n}(\theta'_i) \cdot P = \sum_{i=1}^{n}(\theta_i) \cdot P.$$

Since the two aggregated expressions match, the batch verification is correct and all signatures in the batch are valid.

C. Correctness of TA Tracking Primary Pseudonyms of Malicious Users

The tracking mechanism ensures that the TA can correctly recover the primary pseudonym PP'_i of a malicious user from the temporary pseudonym TP_i. The core of the tracking process lies in checking whether the following equation holds:

$$PP'_i = TP_i \oplus H_3(t \cdot TPPK_{1i}).$$

Fundamentally, this procedure verifies whether the reconstructed pseudonym PP_i indeed equals the original PP'_i. To make the derivation clear, we now present the correctness proof step by step.

1) By the definition of the tracking algorithm.

$$PP'_i = TP_i \oplus H_3(t \cdot TPPK_{1i}).$$

2) Substituting the definition of $TPPK_{1i}$.

$$PP'_i = TP_i \oplus H_3(t \cdot x_i \cdot v_i \cdot P).$$

3) Using the definition of the TA's public key $T_{pub} = t \cdot P$.

$$PP'_i = TP_i \oplus H_3(x_i \cdot v_i \cdot T_{pub}).$$

4) Using the definition of the primary pseudonym $PP_i = TP_i \oplus H_3(x_i \cdot v_i \cdot T_{pub})$.

$$PP'_i = PP_i.$$

Since the expressions match, the TA can correctly recover the primary pseudonym of the malicious user, confirming the correctness of the tracking mechanism.

D. Correctness of Detecting KGC's Active Attack

The detection mechanism enables the TA to determine whether the KGC has launched any active attack against a user. The core of this mechanism is to verify whether the following equation holds:

$$TPPK_{2i}' = \frac{TPPK_{1i}}{x_i} + PPK_i.$$

Essentially, this process verifies whether the reconstructed temporary partial public key $TPPK_{2i}$ matches the original value $TPPK_{2i}'$. Since the KGC does not possess the user's secret value x_i, it cannot forge a consistent temporary partial public key. Therefore, the TA only needs to obtain the user's secret value and the second temporary partial public key contained in the suspicious message, recompute a fresh $TPPK_{2i}$, and compare the two values to determine whether the KGC has performed an active manipulation. For clarity and rigor, we now present the correctness derivation step by step.

1) By the definition of the detection algorithm.
$$TPPK_{2i}' = \frac{TPPK_{1i}}{x_i} + PPK_i.$$
2) Substituting the definitions of $TPPK_{1i} = x_i \cdot v_i \cdot P$ and $PPK_i = y_i \cdot P$.
$$TPPK_{2i}' = \frac{x_i \cdot v_i \cdot P}{x_i} + y_i \cdot P.$$
3) Applying cancellation over the elliptic-curve group.
$$TPPK_{2i}' = v_i \cdot P + y_i \cdot P.$$
4) Using the definition of the original temporary partial public key $TPPK_{2i} = (v_i + y_i) \cdot P$.
$$TPPK_{2i}' = TPPK_{2i}.$$

Since the two values match, the TA can correctly detect any active attack attempted by the KGC. This confirms the correctness of the KGC attack detection mechanism.

III. Main Security Requirements

A. Definitions of Main Security Requirements

In analyzing the main security requirements of anonymity, unlinkability, and replay-attack resistance, we introduce a unified privacy adversary, denoted as $\mathcal{A}_3$. This adversary neither possesses the public-key replacement capability inherent to certificateless cryptography nor has access to the system's master key. Its objectives include recovering the real identity of a message sender, determining whether multiple messages originate from the same vehicle, and attempting to make replayed messages pass verification.

Create User Oracle: When adversary $\mathcal{A}_3$ queries the challenger $\mathcal{C}$ for user PP_i, $\mathcal{C}$ will respond by outputting the partial public key PPK_{1i}.

The H_i Oracle $(i = 2, 3, 4)$, Partial Private Key Oracle, Secret Value Oracle, Signing Oracle, and Temporary Private Key Oracle are defined in the same manner as in the formal model presented in Section II of the main text.

Definition 4 (Anonymity). If a polynomial-bounded adversary $\mathcal{A}_3$ wins Games III with a negligible advantage, then we can consider that our SGPBC-CPPA scheme satisfies anonymity.

Game III:

Setup: The adversary $\mathcal{A}_3$ initiates this query to the challenger $\mathcal{C}$, which then executes the system initialization algorithm to generate the system parameters $params$. Subsequently, $\mathcal{C}$ sends $params$ as a response to the adversary.

Phase 1: During this phase, $\mathcal{A}_3$ can query all oracles.During this phase, $\mathcal{A}_3$ may query all oracles, with three exception:

1) $\mathcal{A}_3$ is not allowed to query the Secret Value Oracle for PP_i^*.
2) $\mathcal{A}_3$ is not allowed to query the Partial Private Key Oracle Oracle for PP_i^*.
3) $\mathcal{A}_3$ is not allowed to query the Temporary Private Key Oracle for PP_i^*.

Challenge: The adversary $\mathcal{A}_3$ submits two identities of equal length, PP_0 and PP_1, together with the timestamp Ts_i and the message m. The challenger randomly selects an identity PP_i from PP_0, PP_1. If this identity is not equal to the challenge identity PP_i^*, the challenge terminates; otherwise, the challenger generates the corresponding challenge signature under this identity and returns it to $\mathcal{A}_3$.

Phase 2: Consistent with Phase 1.

Guess: The adversary $\mathcal{A}_3$ outputs a guess PP' for the signature σ. If $PP' = PP_i$, then the adversary wins the above game.

Definition 5 (Unlinkability). If a polynomial-bounded adversary $\mathcal{A}_3$ wins Games IV with a negligible advantage, then we can consider that our SGPBC-CPPA scheme satisfies unlinkability.

Game IV:

Setup: Consistent with Game III.

Phase 1: Consistent with Game III.

Challenge: The adversary $\mathcal{A}_3$ submits two messages m_0 and m_1, together with an identity PP_i, and the timestamp Ts_i. If $PP_i \neq PP_i^*$, the challenge aborts; otherwise, the challenger randomly selects a bit $b \in \{0, 1\}$ and proceeds as follows:

1) If $b = 0$, the challenger generates two valid signatures (σ_0, σ_1) on (m_0, m_1) using the same identity PP_i^*;
2) If $b = 1$, the challenger generates a signature σ_0 on m_0 using PP_i^*, and generates a signature σ_1 on m_1 using another valid identity PP_j such that $PP_j \neq PP_i^*$.

The challenger then returns (σ_0, σ_1) to $\mathcal{A}_3$.

Phase 2: Consistent with Phase 1.

Guess: The adversary outputs a bit b' as its guess for whether the two signatures (σ_0, σ_1) were generated under the same identity. The adversary wins the game if $b' = b$.

Definition 6 (Resist replay attack). If a polynomial-bounded adversary $\mathcal{A}_3$ wins Games V with a negligible advantage, then we can consider that our SGPBC-CPPA scheme satisfies resist replay attack.

Game V:

Setup: Consistent with Game III.

Phase 1: Consistent with Game III.

Challenge: The adversary selects a message m^*, an identity PP_i, and a timestamp Ts_i^*. If $PP_i \neq PP_i^*$, the challenge aborts; otherwise, the challenger returns the message together with its corresponding signature $\sigma^* = (m^*, \theta^*, TP_i^*, CTPK_i^*, Ts_i^*, TS_i^*)$ to the adversary $\mathcal{A}_3$.

Phase 2: Consistent with Phase 1.

Replay: The adversary $\mathcal{A}_3$ outputs a message $\sigma' = (m^*, \theta', TP_i^*, CTPK_i^*, Ts_i^*, TS_i')$ in an attempt to replay the challenged message. The adversary is considered to win Game V if any of the following conditions is met:

1) $\sigma' = \sigma^*$, the adversary replays an old message without any modification, but the verifier still accepts it.
2) $TS_i' \neq TS_i^*$, the adversary modifies the timestamp and succeeds in producing a new valid signature θ' for the modified message.

B. Analysis of Main Security Requirements

Theorem 3. If the adversary A_3 can break the anonymity of our SGPBC-CPPA scheme in the ROM with a non-negligible advantage ξ in probabilistic polynomial time, then we can use this adversary to solve the ECCDH problem with a non-negligible advantage ξ'.

$$\xi' \geq \frac{\xi}{e \cdot (1 + q^{\square}) \cdot q_{H3}}$$

Proof: Assuming our scheme is compromised by adversary A_3, in this case, the simulator S is given an instance tuple of the ECCDH problem $(P, aP, bP) \in G$, where the task is to solve the ECCDH problem through interaction with the adversary.

1) Setup: The simulator $\mathcal{S}$ executes the system initialization algorithm, generating the system parameters $params = \{G, P, H_2, H_3, H_4\}$, randomly generates three random values $k_1, k_2 \in Z_q^*$, then sets the TA's public key $T_{pub} = a \cdot P$, KGC's public keys $K_{1pub} = k_1 \cdot P$, $K_{2pub} = k_2 \cdot P$. And it sends the parameters $params = \{G, P, T_{pub}, K_{1pub}, K_{2pub}, H_2, H_3, H_4\}$ to the adversary.
2) Phase 1: At this stage, $\mathcal{A}_3$ can adaptively query the following oracles for a polynomially bounded number of times. The simulator $\mathcal{S}$ sets the challenge identity to $PP_i{}^*$ and maintains five lists $L_1, L_2, L_3, L_{uid}, L_{sig}$ that are initially empty.
   a) Create User Oracle: $\mathcal{A}_3$ queries the simulator $\mathcal{S}$ for the tuple (PP_i). If $PP_i = PP_i{}^*$, $\mathcal{S}$ randomly selects values $w_i \in Z_q^*$ and sets the partial public key $PPK_{1_i}^* = \frac{b \cdot P}{w_i}$ and returns $PPK_{1_i}^*$ to the adversary $\mathcal{A}_3$, and recording $\{PP_i{}^*, \perp, \perp, \perp, \perp, PPK_{1_i}^*, \perp\}$ in list L_{uid}. Conversely, if $PP_i \neq PP_i{}^*$, $\mathcal{S}$ randomly selects values $x_i, y_i \in Z_q^*$ and $Ts_i \in \{0,1\}^{l_2}$, and subsequently queries the H_2 oracle to obtain $h_{2i} = H_2(K_{1pub}, K_{2pub}, Ts_i)$. It calculates the partial public key $PPK_{2_i} = y_i \cdot P$ and the partial privare keys $ppk_{1_i} = y_i + h_{2i} \cdot k_1$ and $ppk_{2_i} = y_i + h_{2i} \cdot k_2$. The simulator returns PPK_{1_i} to the adversary $\mathcal{A}_3$ and records $\{PP_i, x_i, y_i, ppk_{1_i}, ppk_{2_i}, PPK_{1_i} = x_i \cdot P, PPK_{2_i}\}$ in list L_{uid}.
   b) Signing Oracle: $\mathcal{A}_3$ queries the simulator $\mathcal{S}$ for the tuple (PP_i, Ts_i, m_i).
      i) If $PP_i = PP_i{}^*$, $\mathcal{S}$ proceeds by first computing $h_{2i} = H_2(K_{1pub}, K_{2pub}, Ts_i)$. It then randomly selects $\theta^*, v_i, h_{4i} \in Z_q^*$, $h_{3i} \in \{0,1\}^{l_1}$, and $TS_i \in \{0,1\}^{l_2}$. Following this, it computes $TP_i = PP_i{}^* \oplus h_{3i}$. Subsequently, it sets $TPPK_{1i} = v_i \cdot PPK_{1_i}^*$ and $TPPK_{2i} = (\theta^* \cdot P - h_{2i} \cdot K_{2\text{pub}} - h_{4i} \cdot (TPPK_{1i} + h_{2i} \cdot K_{2\text{pub}}))/(1 + h_{4i})$. Finally, $\mathcal{S}$ returns the message $\sigma = \{m_i, \theta^*, TP_i, TPPK_{1i}, TPPK_{2i}, Ts_i, TS_i\}$ to the adversary $\mathcal{A}_3$ and records $\{PP_i{}^*, TP_i, v_i, \perp, \perp, \perp, \theta^*, h_{4i}, TPPK_{1i}, TPPK_{2i}, Ts_i, TS_i\}$ in the list L_{sig}.
      ii) If $PP_i \neq PP_i{}^*$, $\mathcal{S}$ executes the signing algorithm and records $\{PP_i, TP_i, v_i, tsv_i = x_i \cdot v_i, tppk_{1_i} = v_i + ppk_{1_i}, tppk_{2_i} = v_i + ppk_{2_i}, \theta_i, h_{4i}, TPPK_{1i}, TPPK_{2i}, Ts_i, TS_i\}$ in list L_{sig}.
   c) The H_2 Oracle, H_3 Oracle, H_4 Oracle, the Partial Private Key Oracle, the Secret Value Oracle, and the Temporary Private Key Oracle are defined in the same manner as in Theorem 1.
3) Challenge: The adversary $\mathcal{A}_3$ sends two identities of equal length, PP_0 and PP_1, together with the timestamp Ts_i and the message m, to the simulator $\mathcal{S}$. The simulator $\mathcal{S}$ randomly selects a primary pseudonym PP_i from $\{PP_0, PP_1\}$. If $PP_i \neq PP_i^*$, the challenge terminates; otherwise, $\mathcal{S}$ performs the following operations: First, $\mathcal{S}$ sets $TPPK_{1_i}^* = b \cdot P$. Then, it randomly selects θ, h_{2i}, h_{4i} from Z_q^*, randomly selects $U^* \in G$, and randomly selects $h_{3,i}$ and TS_i from $\{0,1\}^{l_1}$ and $\{0,1\}^{l_2}$, respectively. Next, it computes $TP_i^* = PP_i^* \oplus H_3(U^*)$. It then computes $TPPK_{2_i}^* = (\theta \cdot P - h_{2i} \cdot K_{2\text{pub}} - h_{4i} \cdot (TPPK_{1_i}^* + h_{2i} \cdot K_{2\text{pub}}))/(1 + h_{4i})$. Finally, the simulator $\mathcal{S}$ returns the message $\sigma = \{m, \theta, TP_i^*, TPPK_{1_i}^*, TPPK_{2_i}^*, Ts_i, TS_i\}$ to the adversary $\mathcal{A}_3$.
4) Phase 2: Consistent with Phase 1.
5) Guess: The adversary $\mathcal{A}_3$ outputs its guess $PP_i' \in \{PP_0, PP_1\}$. If $PP_i' = PP_i$, then $\mathcal{A}_3$ wins Game III.

In the above game, $\mathcal{A}_3$ cannot determine whether TP_i corresponds to the challenge identity unless it queries the H_3 oracle on (U^*). According to the construction of U^*, we have $U^* = abP$. Therefore, if $\mathcal{A}_3$ wins Game III with non-negligible advantage, then the simulator $\mathcal{S}$ can directly use $abP = U^*$ as the solution to the ECCDH problem.

Probability Analysis. From the analysis of the above games, we can conclude that if the simulation does not abort, the simulator $\mathcal{S}$ is able to successfully solve the ECCDH problem. To guarantee that $\mathcal{S}$ succeeds, the following conditions must hold:

1) π_1: In the partial private key oracle phase, $\mathcal{A}_3$ makes queries with $PP_i \neq PP_i^*$. $Pr[\pi_1] = \varphi^{q_{pp}}$.

2) π_2: In the secret value oracle phase, $\mathcal{A}_3$ makes queries with $PP_i \neq PP_i^*$. $Pr[\pi_2] = \varphi^{q_{sv}}$.
3) π_3: In the temporary private key oracle phase, $\mathcal{A}_3$ makes queries with $PP_i \neq PP_i^*$. $Pr[\pi_3] = \varphi^{q_{tpk}}$.
4) π_4: The simulation does not abort in the challenge phase. $Pr[\pi_4] = 1 - \varphi$.
5) π_5: The adversary makes a correct guess that enables the simulator $\mathcal{S}$ to extract the ECCDH solution. This event occurs with probability $Pr[\pi_5] = \xi / q_{H_3}$.

To minimize the probability of simulation termination, it is essential to maximize $Pr[\mathcal{S}$ success$]$. This maximum is achieved when $\varphi = \frac{q_{pp}+q_{sv}+q_{tpk}}{1+q_{pp}+q_{sv}+q_{tpk}}$. Let $q^{\square} = q_{pp}+q_{sv}+q_{tpk}$.

Therefore, with the assistance of the adversary, the probability that $\mathcal{S}$ successfully solves the ECCDH problem is:

$$\begin{aligned} Pr[\mathcal{S} \text{ success}] &= Pr[\pi_1 \wedge \pi_2 \wedge \pi_3 \wedge \pi_4 \wedge \pi_5] \\ &= \varphi^{q^{\square}}(1-\varphi) \cdot \frac{\xi}{q_{H_3}} \\ &\geq \frac{\xi}{e(1+q^{\square})q_{H_3}}. \end{aligned}$$

Theorem 4. If the adversary A_3 can break the unlinkability of our SGPBC-CPPA scheme in the ROM with a non-negligible advantage ξ in probabilistic polynomial time, then we can use this adversary to solve the ECCDH problem with a non-negligible advantage ξ'.

$$\xi' \geq \frac{\xi}{e \cdot (1+q^{\square}) \cdot q_{H_3}}$$

Proof: Assuming our scheme is compromised by adversary A_3, in this case, the simulator S is given an instance tuple of the ECCDH problem $(P, aP, bP) \in G$, where the task is to solve the ECCDH problem through interaction with the adversary.

1) Setup and Phase 1 are identical to those in Theorem 3.
2) Challenge: The adversary $\mathcal{A}_3$ submits two messages m_0 and m_1, together with an identity PP_i, and the timestamp Ts_i^*. If $PP_i \neq PP_i^*$, the challenge aborts; otherwise, the challenger randomly selects a bit $b \in \{0,1\}$ and proceeds as follows:
   a) If $b = 0$, the simulator $\mathcal{S}$ must generate two signatures on m_0 and m_1 under the same identity PP_i^*. To accomplish this, $\mathcal{S}$ applies the same signing simulation procedure independently for each message. For a given message m, the simulator first randomly selects a value $v_i \in Z_q^*$ and computes $TPPK1_i^* = v_i \cdot PPK1_i^*$. It then randomly chooses $\theta, h_{2i}, h_{4i} \in Z_q^*$, selects $U^* \in G$, $h_{3i} \in \{0,1\}^{l_1}$, and $TS_i^* \in \{0,1\}^{l_2}$, respectively. Next, it computes $TP_i^* = PP_i^* \oplus H_3(U^*)$, and then evaluates $TPPK2_i^* = (\theta \cdot P - h_{2i} \cdot K_{2\text{pub}} - h_{4i} \cdot (TPPK1_i^* + h_{2i} \cdot K_{2\text{pub}}))/(1+h_{4i})$. Finally, the simulator outputs the signature $\sigma = \{m, \theta, TP_i^*, TPPK1_i^*, TPPK2_i^*, Ts_i^*, TS_i^*\}$. By executing the above procedure twice with fresh randomness, $\mathcal{S}$ obtains two signatures σ_0 and σ_1 on (m_0, m_1) under the same identity PP_i^*.
   b) If $b = 1$, the simulator $\mathcal{S}$ first generates a signature on m_0 under the challenge identity PP_i^* by applying the same signing simulation procedure as described above for PP_i^*. It then selects another valid identity $PP_j \neq PP_i^*$ and runs the original signing algorithm of the scheme to generate a valid signature on m_1 under PP_j.

   Finally, the simulator $\mathcal{S}$ returns the two signatures (σ_0, σ_1) to the adversary $\mathcal{A}_3$.
3) Phase 2: Consistent with Phase 1.
4) Guess: The adversary $\mathcal{A}_3$ outputs a bit b' as its guess for whether the two signatures (σ_0, σ_1) were generated under the same identity. The adversary wins Game IV if $b' = b$.

In the above game, the adversary $\mathcal{A}_3$ cannot determine whether (σ_0, σ_1) originate from the same user unless it queries the H_3 oracle on U^*. According to the construction of U^*, we have $U^* = v_i \cdot abP/w_i$. Therefore, if $\mathcal{A}_3$ wins Game IV with a non-negligible advantage, the simulator $\mathcal{S}$ can directly use $abP = U^* \cdot w_i/v_i$ as the solution to the ECCDH problem.

Probability Analysis. Same as in Theorem 3, therefore, with the assistance of the adversary, the probability that $\mathcal{S}$ successfully solves the ECCDH problem is:

$$\begin{aligned} Pr[\mathcal{S} \text{ success}] &= Pr[\pi_1 \wedge \pi_2 \wedge \pi_3 \wedge \pi_4 \wedge \pi_5] \\ &= \varphi^{q^{\square}}(1-\varphi) \cdot \frac{\xi}{q_{H_3}} \\ &\geq \frac{\xi}{e(1+q^{\square})q_{H_3}}. \end{aligned}$$

Theorem 5. If the adversary A_3 can break the replay-attack resistance our SGPBC-CPPA scheme in the ROM with a non-negligible advantage ξ in probabilistic polynomial time, then we can use this adversary to solve the ECCDH problem with a non-negligible advantage ξ'.

$$\xi' \geq \frac{\xi}{e \cdot (1+q^{\square})}.$$

Proof. Assuming our scheme is compromised by adversary A_3, in this case, the simulator S is given an instance tuple of the ECDL problem $(P, aP) \in G$, where the task is to solve the ECDL problem through interaction with the adversary.

1) Setup: consistent with Theorem 1.
2) Phase 1: Same as Theorem 1, but without the public key replacement oracle.
3) Challenge: The adversary selects a message m^*, an identity PP, and a timestamp Ts_i^*. If $PP \neq PP_i^*$, the challenge aborts; otherwise, the challenger generates the corresponding challenge signature as follows. The simulator $\mathcal{S}$ first computes $h_{2i} = H_2(K_{1pub}, K_{2pub}, Ts_i^*)$, and then randomly selects $\theta^*, v_i, h_4{}^* \in Z_q^*$, $h_{3i} \in \{0,1\}^{l_1}$, and $TS_i^* \in \{0,1\}^{l_2}$. Next, it computes $V_i = v_i \cdot P$ and derives the temporary pseudonym $TP_i{}^* = PP_i{}^* \oplus h_{3i}$. The simulator then sets $TPPK1_i^* = (\theta^* \cdot P - V_i - PPK_2{}^* - h_{2i} \cdot K_{2pub}) \cdot (h_4{}^*)^{-1} - V_i - PPK_2{}^* - h_{2i} \cdot K_{1pub}$ and $TPPK2_i^* = PPK_2{}^* + V_i$. Finally, $\mathcal{S}$ returns the challenge signature $\sigma^* = (m^*, \theta^*, TP_i{}^*, TPPK1_i^*, TPPK2_i^*, Ts_i^*, TS_i^*)$ to the adversary $\mathcal{A}_3$.
4) Phase 2: consistent with Phase 1.

5) Replay: The adversary $\mathcal{A}_3$ outputs a message $\sigma' = (m^*, \theta', TP_i{}^*, TPPK1_i^*, TPPK2_i^*, Ts_i^*, TS_i')$ as its replay attempt. The adversary is considered to win Game V if either of the following conditions holds:
   a) $\sigma' = \sigma^*$, that is, the adversary successfully replays the old message without modification and the verifier still accepts it. (This case cannot occur in practice since a verbatim replay of the challenge signature σ^* with the same timestamp Ts_i^* will be rejected by the verifier.)
   b) $TS_i' \neq TS_i^*$ and θ' is a valid signature on the modified message, meaning that the adversary successfully forges a fresh timestamped signature.
   If either of the above conditions is satisfied, then $\mathcal{A}_3$ wins Game V.

If $\mathcal{A}_3$ succeeds in producing a new valid signature θ' on the modified timestamp $TS_i' \neq TS_i^*$, then we obtain

$$\begin{aligned}\theta' \cdot P =& (TPPK1_i^* + TPPK2_i^* + h_{2,i} \cdot K_{1_{\text{pub}}}) \cdot h_4' \\ & + TPPK2_i^* + h_{2,i} \cdot K_{2_{\text{pub}}}.\end{aligned}$$

Moreover, during the challenge phase, the simulator $\mathcal{S}$ has already produced the original challenge signature θ^* satisfying

$$\begin{aligned}\theta^* \cdot P =& (TPPK1_i^* + TPPK2_i^* + h_{2,i} \cdot K_{1_{\text{pub}}}) \cdot h_4^* \\ & + TPPK2_i^* + h_{2,i} \cdot K_{2_{\text{pub}}}.\end{aligned}$$

Since h_4' and h_4^* are independently and uniformly chosen from Z_q^*, the two equations are linearly independent with overwhelming probability.
According to the above two equations $\mathcal{S}$ can calculate the $a = \theta^* - v_i - h_{2i} \cdot k_2 - \frac{\theta^* - \theta'}{h_4^* - h_4'} \cdot h_4{}^*$ as a solution to the ECDL problem.

Probability Analysis. From the analysis of the above game, we can conclude that, if the simulation does not abort and the adversary $\mathcal{A}_3$ successfully outputs a replay message with a modified timestamp and a valid signature, then the simulator $\mathcal{S}$ is able to solve the ECDL problem. To guarantee that $\mathcal{S}$ succeeds, the following conditions must hold:

1) π_1: In the partial private key oracle phase, $\mathcal{A}_3$ makes all queries on identities $PP_i \neq PP_i{}^*$. $Pr[\pi_1] = \varphi^{q_{pp}}$.
2) π_2: In the secret value oracle phase, $\mathcal{A}_3$ makes all queries on identities $PP_i \neq PP_i{}^*$. $Pr[\pi_2] = \varphi^{q_{sv}}$.
3) π_3: In the temporary private key oracle phase, $\mathcal{A}_3$ makes all queries on identities $PP_i \neq PP_i{}^*$. $Pr[\pi_3] = \varphi^{q_{tpk}}$.
4) π_4: In the challenge phase, the challenge identity is exactly $PP_i{}^*$, so that the simulation does not abort. Then $Pr[\pi_4] = 1 - \varphi$.
5) π_5: In the replay phase, $\mathcal{A}_3$ outputs a replay message $\sigma' = (m^*, \theta', TP_i^*, TPPK1_i^*, TPPK2_i^*, Ts_i^*, TS_i')$ with $TS_i' \neq TS_i^*$ and a valid signature θ'. $Pr[\pi_5] = \xi$.

To minimize the probability of simulation termination, it is essential to maximize the probability of success for $\mathcal{S}$, denoted as $Pr[\mathcal{S}\text{ success}]$. This maximum success probability is achieved when the parameter φ is set to $\frac{q_{pp}+q_{sv}+q_{tpk}}{1+q_{pp}+q_{sv}+q_{tpk}}$. Let $q^{\square}$ denote $q_{pp} + q_{sv} + q_{tpk}$. Therefore, with the assistance of the adversary, the probability that $\mathcal{S}$ successfully solves the ECCDH problem is given by:

$$\begin{aligned}Pr[\mathcal{S}\text{ success}] &= Pr[\pi_1 \wedge \pi_2 \wedge \pi_3 \wedge \pi_4 \wedge \pi_5] \\ &= \varphi^{q^{\square}} \cdot (1-\varphi) \cdot \xi \\ &\geq \frac{\xi}{e \cdot (1+q^{\square})}.\end{aligned}$$

References

[1] Y. Genc, N. Aytas, A. Akkoc, E. Afacan, and E. Yazgan, "Elcpas: A new efficient lightweight certificateless conditional privacy preserving authentication scheme for iov," *Vehicular Communications*, vol. 39, p. 100549, 2023.