A Security-Enhanced Pairing-Free Certificateless Aggregate Signature for Vehicular Ad-Hoc Networks, Revisited

Zhengjun Cao, Lihua Liu

Abstract. We show that the aggregate signature scheme [IEEE Syst. J., 2023, 17(3), 3822-3833] is insecure against forgery attack. This flaw is due to that the ephemeral key or ephemeral value chosen in the signing phase is not indeed bound to the final signature. An adversary can sign any message while the verifier cannot find the fraud. We also suggest a revising method to frustrate this attack.
Keywords: Certificateless public key, aggregate signature, forgery attack, vehicular ad-hoc network, ephemeral key.

1 Introduction

Vehicular ad-hoc network (VANET) has become a hot research topic owing to the demand for road safety and management, in which there are two kinds of communication: vehicle to vehicle, and vehicle to infrastructure. To identify entities in VANETs, many authentication schemes are presented by using different techniques. Among these, Rasheed et al [7] proposed a group-based zero knowledge proof-authentication protocol. Wu et al [10] investigated a secure authentication and key exchange protocol. Kumar and Om [3] proposed a cache-based authentication scheme. Pulagara et al [6] presented a group-key management scheme. Cahyadi and Hwang [2] studied the batch verification techniques in authentication scheme. Shawky et al [9] presented a cross-layer authentication scheme for VANETs. Limbasiya et al [4] presented some lightweight communication protocols for smart parking management.

The certificate management in traditional public-key infrastructure (PKI) has become a major bottleneck. Identity-based cryptosystem (IBC) is barely satisfactory due to its key escrow problem. So, the certificateless public key cryptography [1] could be an ideal solution to many applications, in which the authority (KGC) only computes a partial private key of any user, who then combines this partial private key with some secret values (only known to him) to obtain an extra public key (uncertificated). Recently, Zheng et al. [11] have presented a certificateless aggregate signature scheme in order to meet many security requirements, including mutual authentication, message integrity, resistance to impersonation attack, signature forgery attack, etc.

In this note, we show that the Zheng et al's scheme is insecure against signature forgery attack. We also fix this flaw by inputting the unique ephemeral value to one hash function so as to construct a true intractable challenge.

Z. Cao, Department of Mathematics, Shanghai University, Shanghai, 200444, China.
L. Liu, Department of Mathematics, Shanghai Maritime University, Shanghai, 201306, China.
Email: liulh@shmtu.edu.cn

2 Review of the Zheng et al.'s scheme

In the considered scenario, there are five entities: TA, KGC, OBU, RSU, and AB. The notations and descriptions are listed below (Table 1).

Table 1: Notations and descriptions

TA	trusted authority
KGC	key generation center
OBU	on-board unit
RSU	road-side unit
AB	application backend
a, K_{pub}	KDC's master private key, public key
b, T_{pub}	TA's master private key, public key
ID_i, PID_i	ith vehicle V_i's real identity, pseudonym identity
PK_i, SK_i	V_i's public key, private key
d_i, x_i	V_i's partial private key, secret value
M_i, σ_i	V_i's message, signature
$\triangle T_i$	validity period of V_i's pseudonym identity

TA is responsible for generating pseudo identities and registration service, who can reveal the true identity using the master key. KGC is responsible for system parameters generation and partial private key extraction. OBU, a resource-constrained device, can communicate with other nodes. Each vehicle has its real identity, some pseudo-identities, and its public/private key pairs. RSU is a base station with more computation or storage resources. AB supplies vehicles with various kinds of data services.

For the certificateless signature scheme, two adversary models are considered [4, 11]. An external user can replace the public key of any entity, but cannot access the master key. A malicious KGC has the master key, but cannot replace the public key of a certain party.

The scheme can be reviewed as follows (Table 2).

3 Insecure against forgery attack

3.1 Partially bound ephemeral key

Notice that in the verification equation

$$s_i P = U_i + (R_i + h_{1i} K_{pub}) h_{2i} + X_i h_{3i} \quad (1)$$

both P and K_{pub} are two system public parameters, and authentic. R_i, X_i are not authentic. Since the ephemeral key $u_i \in Z_q^*$, is only bound to

$$h_{3i} = h_3(PID_i \| M_i \| PK_i \| U_i \| T_i),$$
$$s_i = u_i + d_i h_{2i} + x_i h_{3i},$$

Table 2: The Zheng et al.'s aggregate signature scheme

Initialization. KGC chooses a group G over the elliptic curve E, with a generator $P \in G$ of a prime order q. Set $a \in Z_q^*, K_{pub} = aP$ as its master key and public key, respectively. TA sets $b \in Z_q^*, T_{pub} = bP$ as its master key and public key, respectively. Let $h_1, h_2, h_3 : \{0,1\}^* \to Z_q^*$ be three hash functions. Publish the system parameters $\{P, q, E, h_1, h_2, h_3, K_{pub}, T_{pub}\}$.
Pseudo-identity Generation. V_i picks $y_i \in Z_q^*$ to compute $PID_{i,1} = y_i P,\ Y_i = y_i T_{pub} \oplus RID_i$. TA computes $RID_i = Y_i \oplus bPID_{i,1}$ to verify the real identity and generates the pseudo-identity $PID_i = \{PID_{i,1}, PID_{i,2}, \triangle T_i\}$, where $PID_{i,2} = RID_i \oplus h_1(bPID_{i,1} \| \triangle T_i)$.
Partial Private Key Extraction. KGC picks $r_i \in Z_q^*$ to compute $R_i = r_i P$, $h_{1i} = h_1(PID_i \| R_i \| K_{pub})$, $d_i = r_i + ah_{1i}$. Send $\{PID_i, d_i, R_i\}$ to V_i via a secure channel.
Vehicle Key Generation. V_i picks $x_i \in Z_q^*$ to compute $X_i = x_i P$. Set $SK_i = \{x_i, d_i\}$ as its private key, $PK_i = \{X_i, R_i\}$ as its public key.
Signing. Given a message M_i, V_i picks a nonce $u_i \in Z_q^*$ to compute $U_i = u_i P$, $h_{2i} = h_2(PID_i \| X_i \| K_{pub} \| R_i \| T_i)$, $h_{3i} = h_3(PID_i \| M_i \| PK_i \| U_i \| T_i)$, $s_i = u_i + d_i h_{2i} + x_i h_{3i}$, $\sigma_i = (U_i, s_i)$, where T_i is the timestamp. Output $\{PID_i, PK_i, M_i, T_i, \sigma_i\}$.
Verification. The verifier checks the timestamp T_i. Then compute $h_{1i} = h_1(PID_i \| R_i \| K_{pub})$, $h_{2i} = h_2(PID_i \| X_i \| K_{pub} \| R_i \| T_i)$, $h_{3i} = h_3(PID_i \| M_i \| PK_i \| U_i \| T_i)$. Check if $s_i P \stackrel{?}{=} U_i + (R_i + h_{1i} K_{pub}) h_{2i} + X_i h_{3i}$.
Aggregate. RSU collects the signatures $\{PID_i, PK_i, M_i, T_i, \sigma_i\}_{i=1}^n$, and forwards them to the verifier.
Aggregate Verification. Compute $U = \sum_{i=1}^n U_i$, $s = \sum_{i=1}^n s_i$, $h_{1i} = h_1(PID_i \| R_i \| K_{pub})$, $h_{2i} = h_2(PID_i \| X_i \| K_{pub} \| R_i \| T_i)$, $h_{3i} = h_3(PID_i \| M_i \| PK_i \| U_i \| T_i)$, $i = 1, \cdots, n$. Check if $sP \stackrel{?}{=} U + \sum_{i=1}^n (R_i + h_{1i} K_{pub}) h_{2i} + \sum_{i=1}^n X_i h_{3i}$.

not bound to the hash values h_{1i}, h_{2i}, we find an adversary can make use of this flaw to launch signature forgery attack.

3.2 Unrecognizable pseudo-identities

Notice that only the trusted authority TA can recognize a faked pseudo-identity by checking whether there exists $RID_i \in \mathcal{ID}$ such that $PID_{i,2} = RID_i \oplus h_1(bPID_{i,1} \| \triangle T_i)$, where b is the TA's master key, and $\mathcal{ID}$ is the set of all registered users's IDs. Other entities, especially the verifier, cannot detect the faked pseudo-identity PID_i, because both the master key b and the set $\mathcal{ID}$ are inaccessible to him.

By the way, in the subsequent computations, the three components $PID_{i,1}, PID_{i,2}, \triangle T_i$ are concatenated and input into the three hash functions h_1, h_2, h_3 as

$$h_{1i} = h_1(PID_i \| R_i \| K_{pub}),$$
$$h_{2i} = h_2(PID_i \| X_i \| K_{pub} \| R_i \| T_i),$$
$$h_{3i} = h_3(PID_i \| M_i \| PK_i \| U_i \| T_i).$$

The dependency

$$PID_{i,2} = RID_i \oplus h_1(bPID_{i,1} \| \triangle T_i)$$

is never used in the later Verification phase, Aggregate phase, and Aggregate Verification phase.

3.3 Signature forgery attack

First, the adversary randomly picks $PID_{i,1}, PID_{i,2} \in G$, chooses a validity period $\triangle T_i$, and sets $PID_i = \{PID_{i,1}, PID_{i,2}, \triangle T_i\}$ as his pseudo-identity.

Second, for a message M_i, the adversary can pick $x_i, r_i, u_i \in Z_q^*$ and a timestamp T_i to compute

$$\begin{aligned}
&X_i = x_i P,\ R_i = r_i P,\ PK_i = (X_i, R_i),\\
&h_{1i} = h_1(PID_i \| R_i \| K_{pub}),\\
&h_{2i} = h_2(PID_i \| X_i \| K_{pub} \| R_i \| T_i),\\
&U_i = u_i P - (R_i + h_{1i} K_{pub}) h_{2i},\\
&h'_{3i} = h_3(PID_i \| M_i \| PK_i \| U_i \| T_i),\\
&s_i = u_i + x_i h_{3i}
\end{aligned}$$

The forged signature is $\sigma_i = (U_i, s_i)$.

Third, the adversary outputs the message and signature as $\{PID_i, PK_i, M_i, T_i, \sigma_i\}$.

Correctness. The forged signature can pass the verification process. In fact, we have

$$\begin{aligned}
&U_i + (R_i + h_{1i} K_{pub}) h_{2i} + X_i h_{3i}\\
=&u_i P - (R_i + h_{1i} K_{pub}) h_{2i} + (R_i + h_{1i} K_{pub}) h_{2i} + X_i h_{3i}\\
=&u_i P + X_i h_{3i} = (u_i + x_i h_{3i}) P = s_i P
\end{aligned}$$

That means the signature will be accepted by the verifier.

4 A revision

The above flaw is due to that an adversary can freely choose the ephemeral key $u_i \in Z_q^*$ to generate a proper term

$$U_i = u_i P - (R_i + h_{1i} K_{pub}) h_{2i} \tag{2}$$

after the hash value h_{2i} is computed. In order to restrict the adversary's capability to create such a term, one needs to use the ephemeral key and the hash function h_2 to construct a true intractable challenge. To do so, it can specify that

$$h_{2i} = h_2(PID_i \| X_i \| K_{pub} \| R_i \| T_i \| U_i) \tag{3}$$

in the replacement of

$$h_{2i} = h_2(PID_i \| X_i \| K_{pub} \| R_i \| T_i) \tag{3'}$$

i.e., binding the ephemeral value U_i both to the hash values h_{2i}, h_{3i}. In this case, the related computation becomes

$$U_i = u_i P - (R_i + h_1(PID_i \| R_i \| K_{pub}) K_{pub}) \cdot h_2(PID_i \| X_i \| K_{pub} \| R_i \| T_i \| U_i)$$

which is an intractable problem due to the unpredictability of hash function [5]. As a result, the generation order between U_i and h_{2i} is exactly restricted to

$$U_i = u_i P,\ \ h_{2i} = h_2(PID_i \| X_i \| K_{pub} \| R_i \| T_i \| U_i).$$

An adversary cannot first generate h_{2i}, then generate a proper ephemeral value U_i. Therefore, the above forgery attack is frustrated. Now, the new verification equation becomes

$$\begin{aligned} s_i P = & U_i + (R_i + h_1(PID_i \| R_i \| K_{pub})K_{pub}) \cdot h_2(PID_i \| X_i \| K_{pub} \| R_i \| T_i \| U_i) \\ & + X_i h_3(PID_i \| M_i \| PK_i \| U_i \| T_i) \end{aligned}$$

In the right side, all three operands are bound to the ephemeral key u_i. Only the singer who knows the trapdoor between the point $R_i + h_1(PID_i \| R_i \| K_{pub})K_{pub}$ and the base point P, i.e., $r_i + h_{1i}a$, can generate the proper pair (s_i, U_i) satisfying the above verification equation. Its security is directly based on the hybrid intractability of elliptic curve discrete logarithm problem, one-way and collision-free features of hash functions, just like that of the famous Schnorr signature [8].

Notice that the three hash functions h_1, h_2, h_3 have the same domain and codomain. In this case, it suffices to specify a unique hash function $h : \{0,1\}^* \to Z_q^*$. Thus, the scheme's description can be further simplified.

5 The security argument revisited

The original security argument did consider three events, but not checked the dependency between the ephemeral value U_i and the hash value h_{2i}. It argues that (page 3829, Zheng23):

> *Based on forking lemma, $\mathcal{C}_I$ has the capability to replay multiple times of game with identical random type but different hash values $h_{2i}^{*(j)}$ and $h_{3i}^{*(j)}$. Thus, $\mathcal{A}_I$ is able to produce four valid signatures $(U_i^*, s_i^{*(j)}), j \in (1,2,3,4)$. Without loss of generality the following equation is true:*
>
> $$s_i^{*(j)} = u_i^* + (r_i + ah_{1i})h_{2i}^{*(j)} + x_i h_{3i}^{*(j)}.$$

It also argues that (page 3830, Zheng23):

> *In the same way, $\mathcal{C}_{II}$ replays the game multiple times with identical random type but different hash values $h_{2i}^{*(j)}$ and $h_{3i}^{*(j)}$ and produces three valid signatures $(U_i^*, s_i^{*(j)}), j \in (1,2,3)$ hold that*
>
> $$s_i^{*(j)} = u_i^* + (r_i + ah_{1i})h_{2i}^{*(j)} + x_i h_{3i}^{*(j)}.$$

We find that both the challenger $\mathcal{C}_I$ and $\mathcal{C}_{II}$ are subjectively assumed to produce such $s_i^{*(j)}$ concurrently involving the three hash values $h_{1i}, h_{2i}^{*(j)}, h_{3i}^{*(j)}$, and the KGC's master key a. But it failed to construct a true challenge based on the dependency between the signature and verification equation. So, the original proof should be revised as below.

Theorem 1. Suppose there is a PPT (probabilistic polynomial time) adversary $\mathcal{A}_I$ can forge a valid signature with a non-negligible probability through the interaction with challenger $\mathcal{C}_I$.

Then $\mathcal{C}_I$ can make use of $\mathcal{A}_I$'s capability to solve an ECDLP instance with a non-negligible probability.

Proof. The adversary $\mathcal{A}_I$ can replace the public key of any entity, but cannot access the master key a as well as the other master key b. $\mathcal{C}_I$ initializes the system's parameters $params = \{P, q, E, G, h, K_{pub}\}$ and sends $params$ to $\mathcal{A}_I$. $\mathcal{C}_I$ simulates h as a hash oracle, with the query record lists L_1, L_2, L_3. Besides, $\mathcal{A}_I$'s queries, user public key and partial private key are kept in the lists L_{sv}, L_{pk}, L_{ppk}.

HashQurey1: Let $L_1 := \{(PID_i, R_i, K_{pub}, h_{1i})\}$. For an $\mathcal{A}_I$'s query, $\mathcal{C}_I$ checks its freshness. If so, $h_{1i} = h(PID_i \| R_i \| K_{pub})$ is returned to $\mathcal{A}_I$. Otherwise, $\mathcal{C}_I$ submits a query to CreateUser with PID_i, returns to $\mathcal{A}_I$ with such h_{1i} obtained from the oracle, and inserts the new entry $(PID_i, R_i, K_{pub}, h_{1i})$ to the target list.

HashQurey2: $L_2 := \{(PID_i, X_i, K_{pub}, R_i, T_i, U_i, h_{2i})\}$, where $U_i \in G$ is randomly picked by the challenger. For an $\mathcal{A}_I$'s query, $\mathcal{C}_I$ checks its freshness. If so, $h_{2i} = h(PID_i \| X_i \| K_{pub} \| R_i \| T_i \| U_i)$ is returned to $\mathcal{A}_I$. Otherwise, $\mathcal{C}_I$ randomly picks $h_{2i} \in Z_q^*$, and returns it to $\mathcal{A}_I$. Then insert the new entry $(PID_i, X_i, K_{pub}, R_i, T_i, U_i, h_{2i})$ to the list.

HashQurey3: $L_3 := \{(PID_i, M_i, PK_i, U_i, T_i, h_{3i})\}$. For an $\mathcal{A}_I$'s query, $\mathcal{C}_I$ checks its freshness. If so, $h_{3i} = h(PID_i \| M_i \| PK_i \| U_i \| T_i)$ is returned to $\mathcal{A}_I$. Otherwise, $\mathcal{C}_I$ randomly picks $h_{3i} \in Z_q^*$, and returns it to $\mathcal{A}_I$. Then insert the new entry $(PID_i, M_i, PK_i, U_i, T_i, h_{3i})$ to the list.

We refer to the original descriptions [page 3828, Zheng23] for the other three phases, PublicKeyReplace, PartialPrivateKeyExtract, SecretValueExtract, because they have no relation to the additional input item U_i.

Sign: Upon receiving a query (PID_i, M_i) from $\mathcal{A}_I$, $\mathcal{C}_I$ checks its freshness. If so, $\mathcal{C}_I$ checks if $PID_i \neq PID_i^*$ where PID_i^* be the target identity, and the public key hasn't been replaced. Then execute Sign' algorithm and add h_{2i}, h_{3i} to L_2, L_3. Return (U_i, s_i) to $\mathcal{A}_I$. Otherwise, randomly pick $U_i \in G, s_i \in Z_q^*$ and return (U_i, s_i) to $\mathcal{A}_I$.

Forge: $\mathcal{A}_I$ outputs a valid signature (U_i^*, s_i^*) for (PID_i^*, M_i^*) without querying *PartialPrivateKeyExtract* and *Sign* by PID_i^*.

Suppose that $\mathcal{C}_I$ can replay games with identical random type but different hash value. Thus, $\mathcal{A}_I$ can produce four valid signatures $(U_i^*, s_i^{*(j)}), j \in (1, 2, 3, 4)$, satisfying

$$\begin{cases} h_{2i}^{*(j)} = h(PID_i^* \| X_i^* \| K_{pub} \| R_i^* \| T_i^* \| U_i^*), \\ \text{(the original argument forgot to check this dependency.)} \\ h_{3i}^{*(j)} = h(PID_i^* \| M_i^* \| PK_i^* \| U_i^* \| T_i^*), \\ \text{(the original forgot to check this dependency, either.)} \\ s_i^{*(j)} = u_i^* + (r_i + a h_{1i}) h_{2i}^{*(j)} + x_i h_{3i}^{*(j)}. \end{cases}$$

So, $\mathcal{C}_I$ can figure out a. Let $Succ_{CI}$ be the probability of this event. It only needs to consider the below three events:

E1. $\mathcal{C}_I$ doesn't abort the game for all queries.

E2. $\mathcal{A}_I$ successfully forges a valid tuple (PID_i, M_i, U_i, s_i).

E3. $PID_i = PID_i^*$.

Let Q_H and Q_{ppk} be the times of accessing *HashQurey1* and *PartialPrivateKeyExtract*, respectively. Notice that $Q_H \gg Q_{ppk}$, because the number of registered users in the system is limited.

If $\mathcal{A}_I$ can forge a signature with a non-negligible probability $Succ_{AI}$, then

$$Succ_{CI} = (1 - \frac{1}{Q_H})^{Q_{ppk}} Succ_{AI} > (1 - \frac{Q_{ppk}}{Q_H}) Succ_{AI},$$

which is also non-negligible.

Theorem 2. Suppose there is a PPT adversary $\mathcal{A}_{II}$ can forge a valid signature with a non-negligible probability through the interaction with challenger $\mathcal{C}_{II}$. Then $\mathcal{C}_{II}$ can make use of $\mathcal{A}_{II}$'s capability to solve an ECDLP instance with a non-negligible probability.

Proof. The adversary $\mathcal{A}_{II}$ can access the secret master private key, but cannot replace the legal public key of a target entity. $\mathcal{C}_{II}$ initializes the system's parameters $params = \{P, q, E, G, h, K_{pub}\}$, where $K_{pub} = aP$, and sends $a, params$ to $\mathcal{A}_{II}$. $\mathcal{C}_{II}$ simulates h as a hash oracle, with the query record lists L_1, L_2, L_3. Besides, $\mathcal{A}_{II}$'s queries, user public key and partial private key are kept in the lists L_{sv}, L_{pk}, L_{ppk}.

HashQurey1: Let $L_1 := \{(PID_i, R_i, K_{pub}, h_{1i})\}$. For an $\mathcal{A}_{II}$'s query, $\mathcal{C}_{II}$ checks its freshness. If true, $h_{1i} = h(PID_i \| R_i \| K_{pub})$ is returned to $\mathcal{A}_I$. Otherwise, $\mathcal{C}_{II}$ submits a query to CreateUser with PID_i, returns to $\mathcal{A}_{II}$ with such h_{1i} obtained from the oracle, and inserts the new entry $(PID_i, R_i, K_{pub}, h_{1i})$ to the list L_1.

HashQurey2: $L_2 := \{(PID_i, X_i, K_{pub}, R_i, T_i, U_i, h_{2i})\}$, where $U_i \in G$ is randomly picked by the challenger. For an $\mathcal{A}_{II}$'s query, $\mathcal{C}_{II}$ checks its freshness. If true, $h_{2i} = h(PID_i \| X_i \| K_{pub} \| R_i \| T_i \| U_i)$ is returned to $\mathcal{A}_{II}$. Otherwise, $\mathcal{C}_{II}$ randomly picks $h_{2i} \in Z_q^*$, and returns it to $\mathcal{A}_{II}$. Then insert the new entry $(PID_i, X_i, K_{pub}, R_i, T_i, U_i, h_{2i})$ to the list L_2.

HashQurey3: Let $L_3 := \{(PID_i, M_i, PK_i, U_i, T_i, h_{3i})\}$. For an $\mathcal{A}_{II}$'s query, $\mathcal{C}_{II}$ checks its freshness. If true, $h_{3i} = h(PID_i \| M_i \| PK_i \| U_i \| T_i)$ is returned to $\mathcal{A}_I$. Otherwise, $\mathcal{C}_{II}$ randomly picks $h_{3i} \in Z_q^*$, and returns it to $\mathcal{A}_{II}$. Then insert the new entry $(PID_i, M_i, PK_i, U_i, T_i, h_{3i})$ to the list L_3.

We refer to the original descriptions [page 3829, Zheng23] for the other four phases, UserCreate, PublicKeyReplace, PartialPrivateKeyExtract, SecretValueExtract, because they have no relation to the additional input item U_i.

Sign: Upon receiving a query (PID_i, M_i) from $\mathcal{A}_{II}$, $\mathcal{C}_{II}$ checks its freshness. Then $\mathcal{C}_{II}$ checks if $PID_i \neq PID_i^*$, where PID_i^* is the target identity. If true, execute Sign' algorithm and add h_{2i}, h_{3i} to L_2, L_3. Return (U_i, s_i) to $\mathcal{A}_{II}$. Otherwise, randomly pick $U_i \in G, s_i \in Z_q^*$ and return (U_i, s_i) to $\mathcal{A}_{II}$.

Forge: $\mathcal{A}_{II}$ outputs a valid signature (U_i^*, s_i^*) for (PID_i^*, M_i^*) without querying *SecretValueExtract* and *Sign* by PID_i^*.

Suppose that $\mathcal{C}_{II}$ can replay games with identical random type but different hash value. Thus, $\mathcal{A}_{II}$ can produce three valid signatures $(U_i^*, s_i^{*(j)}), j \in (1, 2, 3)$, satisfying

$$\begin{cases} h_{2i}^{*(j)} = h(PID_i^* \| X_i^* \| K_{pub} \| R_i^* \| T_i^* \| U_i^*), \\ h_{3i}^{*(j)} = h(PID_i^* \| M_i^* \| PK_i^* \| U_i^* \| T_i^*), \\ s_i^{*(j)} = u_i^* + (r_i + a h_{1i}) h_{2i}^{*(j)} + x_i h_{3i}^{*(j)}. \end{cases}$$

So, $\mathcal{C}_{II}$ can figure out a. Let $Succ_{CII}$ be the probability of this event. It only needs to consider the below three events:

E1. $\mathcal{C}_{II}$ doesn't abort the game for all queries.

E2. $\mathcal{A}_{II}$ successfully forges a valid tuple (PID_i, M_i, U_i, s_i).

E3. $PID_i = PID_i^*$.

Let Q_H and Q_{sv} be the times of accessing *HashQurey1* and *SecretValueExtract*, respectively. Notice that $Q_H \gg Q_{sv}$, because the number of registered users in the system is limited. If $\mathcal{A}_{II}$ can forge a signature with a non-negligible probability $Succ_{AII}$, then

$$Succ_{CII} = (1 - \frac{1}{Q_H})^{Q_{sv}} Succ_{AII} > (1 - \frac{Q_{sv}}{Q_H}) Succ_{AII},$$

which is non-negligible, too.

6 Conclusion

We show that the Zheng et al.'s certificateless aggregate signature scheme should be revised due to its insecurity against signature forgery attack. We also suggest a revising method to resist this attack. The findings in this note could be helpful for the future work on designing such schemes.

References

[1] S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In *Advances in Cryptology - ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Springer, 2003.

[2] E. F. Cahyadi and M. S. Hwang. An improved efficient authentication scheme for vehicular ad hoc networks with batch verification using bilinear pairings. *Int. J. Embed. Syst.*, 15(2):139–148, 2022.

[3] P. Kumar and H. Om. Secure and efficient cache-based authentication scheme for vehicular ad-hoc networks. *Wirel. Networks*, 28(7):2821–2836, 2022.

[4] T. Limbasiya, S. K. Sahay, and D. Das. SAMPARK: secure and lightweight communication protocols for smart parking management. *J. Inf. Secur. Appl.*, 71:103381, 2022.

[5] A. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

[6] S. B. Pulagara and P. J. A. Alphonse. An intelligent and robust conditional privacy preserving authentication and group-key management scheme for vehicular ad hoc networks using elliptic curve cryptosystem. *Concurr. Comput. Pract. Exp.*, 33(3), 2021.

[7] A. A. Rasheed, R. N. Mahapatra, and F. G. Hamza-Lup. Adaptive group-based zero knowledge proof-authentication protocol in vehicular ad hoc networks. *IEEE Trans. Intell. Transp. Syst.*, 21(2):867–881, 2020.

[8] C. P. Schnorr. Efficient signature generation by smart cards. *J. Cryptol.*, 4(3):161–174, 1991.

[9] M. A. Shawky, M. Bottarelli, G. Epiphaniou, and P. Karadimas. An efficient cross-layer

authentication scheme for secure communication in vehicular ad-hoc networks. *IEEE Trans. Veh. Technol.*, 72(7):8738–8754, 2023.

[10] T. Y. Wu, Z. Lee, L. Yang, and C. M. Chen. A provably secure authentication and key exchange protocol in vehicular ad hoc networks. *Secur. Commun. Networks*, 2021:9944460:1–17, 2021.

[11] H. Zheng, M. Luo, Y. Zhang, C. Peng, and Q. Feng. A security-enhanced pairing-free certificateless aggregate signature for vehicular ad-hoc networks. *IEEE Syst. J.*, 17(3):3822–3833, 2023.