IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 26 January 2024, accepted 26 February 2024, date of publication 4 March 2024, date of current version 3 April 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3373617*

## COMMENTS AND CORRECTIONS

# Comments on "Certificateless Short Aggregate Signature Scheme for Mobile Devices"

**JE HONG PARK AND BONWOOK KOO**
The Affiliated Institute of ETRI, Daejeon 34044, Republic of Korea

Corresponding author: Je Hong Park (jhpark@nsr.re.kr)

This work was supported by the Institute of Information and Communications Technology Planning and Evaluation [Ministry of Science and ICT (MSIT)] Grant funded by Korean Government (MSIT), Development of Next-Generation Cryptosystem to Improve Security and Usability of National Information System, under Grant 2021-0-00046.

**ABSTRACT** In the above paper "Certificateless Short Aggregate Signature Scheme for Mobile Devices" a pairing-based certificateless aggregate signature (CLAS) scheme was proposed. Although the authors claim that their CLAS scheme is secure and provide mathematical proof to support this, we show that anyone can create an aggregate signature on any set of messages without the participation of other users.

**INDEX TERMS** Aggregate signature scheme, certificateless public key cryptography, signature, cryptanalysis.

## I. INTRODUCTION

The aggregate signature is a cryptographic primitive that allows individual signatures on different messages to be combined into a compact signature. The validity of an aggregate signature convinces a verifier that all the individual signatures involved are valid. Signature aggregation is useful in many applications to reduce bandwidth and storage volume, and is particularly attractive for mobile devices.

The notion of certificateless public key cryptography (CL-PKC) has been proposed to provide concise public key management without certificates while eliminating the key escrow problem. To realize this notion, the private key is composed of values generated independently by the third-party key generation center (KGC) and the owner, and the corresponding public key is determined by such private key components.

As a way to improve authentication efficiency while simplifying key management in large-scale IoT environments, various aggregate signature schemes defined on CL-PKC settings (usually referred to as certificateless aggregate signature (CLAS)) have been proposed. As one of them, Deng et al. proposed a pairing-based CLAS scheme and provided a mathematical proof to claim the security of their scheme [1]. In this paper, however, we show that anyone can create an aggregate signature of Deng et al.'s CLAS scheme without the participation of other users. Our attack simply exploits a redundant public key component that is generated by each user and is not certified by a trusted authority. Anyone can manipulate this component to remove the verification equation terms of other individual signatures from the aggregate signature verification equation.

The associate editor coordinating the review of this manuscript and approving it for publication was Jenny Mahoney.

## II. REVIEW OF DENG ET AL.'S CLAS SCHEME

### A. MATHEMATICAL NOTATIONS AND DEFINITIONS

Let $\lambda$ denote the security parameter. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of some large prime order $q$. And let $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$ denote the quotient ring of integers modulo $q$ and let $\mathbb{Z}_q^*$ denote the multiplicative group of $\mathbb{Z}_q$. We write $\mathbb{G}_1$ additively and $\mathbb{G}_2$ multiplicatively. Then $[k]P$ denotes $k$ times addition of $P \in \mathbb{G}_1$ and $g^k$ denotes $k$ times multiplication of $g \in \mathbb{G}_2$, respectively.

A pairing is a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ with the following properties.

- Bilinearity: $\hat{e}([a]P, [b]Q) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q$.
- Non-degeneracy: There exist $P, Q \in \mathbb{G}_1$ such that $\hat{e}(P, Q) \neq 1_{\mathbb{G}_2}$, where $1_{\mathbb{G}_2}$ is the identity of $\mathbb{G}_2$.
- Computability: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in \mathbb{G}_1$.

Note that $\hat{e}$ is symmetric ($\hat{e}(P, Q) = \hat{e}(Q, P)$ for all $P, Q \in \mathbb{G}_1$) since $\hat{e}$ is bilinear and $\mathbb{G}_1$ is a cyclic group.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

Let $\{0,1\}^*$ denote the set of all finite-length binary strings (including the empty string $\epsilon$), let $[1..m]$ denote the set $\{i \in \mathbb{Z} \mid 1 \le i \le m\}$, and let $\{a_i\}_{i=1}^n$ denote a tuple $(a_1, \ldots, a_n)$.

### B. SCHEME DESCRIPTION

The CLAS scheme proposed in [1] consists of following algorithms:

- Setup: The KGC generates parameters and keys as follows:
  - generates groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order $q > 2^\lambda$ with a pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.
  - chooses an arbitrary generator $P \in \mathbb{G}_1$.
  - randomly selects $s \in \mathbb{Z}_q^*$, and sets $P_{\text{pub}} \leftarrow [s]P$.
  - chooses cryptographic hash functions $H_1, H_2, H_3 : \{0,1\}^* \to \mathbb{Z}_q^*$ and $H_4 : \{0,1\}^* \to \mathbb{G}_1$.
  - broadcasts the public parameters $params = \{\mathbb{G}_1, \mathbb{G}_2, q, \hat{e}, P, P_{\text{pub}}, H_1, H_2, H_3, H_4\}$, and keeps the master private key $msk = s$ secret.
- PPK-Extract: For a user with $ID_i \in \{0,1\}^*$, KGC randomly chooses $r_i, s_i \in \mathbb{Z}_q^*$, computes $R_i \leftarrow [r_i]P$, $S_i \leftarrow [s_i]P, h_{1i} \leftarrow H_1(ID_i, R_i, S_i), c_i \leftarrow r_i + h_{1i} \cdot msk$ and $d_i \leftarrow s_i + h_{1i} \cdot msk$, and then sends $D_i = (c_i, d_i, R_i, S_i)$ to the user via a secure channel.
- SV-Set: The user with $ID_i$ randomly chooses two different numbers $x_i, y_i \in \mathbb{Z}_q^*$.
- UPK-Set: The user with $ID_i$ computes $X_i \leftarrow [x_i]P, Y_i \leftarrow [y_i]P$ and sets $pk_i = (X_i, Y_i, R_i, S_i)$.
- Sign: For a message $m_i \in \{0,1\}^*$, the signer with $ID_i$ first chooses a one-time-use state information $\triangle$, then performs the following steps:
  - computes $h_{2i} \leftarrow H_2(m_i, ID_i, pk_i, \triangle)$, $h_{3i} \leftarrow H_3(m_i, ID_i, pk_i, \triangle)$ and $Q \leftarrow H_4(\triangle)$
  - computes $\sigma_i \leftarrow [h_{2i} \cdot c_i + d_i + h_{3i} \cdot x_i + y_i]Q$, and outputs $\sigma_i$ as a signature on $m_i$.
- Aggregate: For a tuple $(\triangle, \sigma_1, \cdots, \sigma_n, M, A)$ where $M = \{m_i\}_{i=1}^n$ and $A = \{(ID_i, pk_i)\}_{i=1}^n$, anyone computes

$$\sigma = \sum_{i=1}^{n} \sigma_i,$$

  and outputs $(\sigma, \triangle, M, A)$.
- Agg-verify: On receiving a tuple $(\sigma, \triangle, M, A)$ where $M = \{m_i\}_{i=1}^n$ and $A = \{(ID_i, pk_i)\}_{i=1}^n$, a verifier performs the following steps:
  1) computes $h_{1i} \leftarrow H_1(ID_i, R_i, S_i)$, $h_{2i} \leftarrow H_2(m_i, ID_i, pk_i, \triangle)$ and $h_{3i} \leftarrow H_3(m_i, ID_i, pk_i, \triangle)$ for each $i \in [1..n]$.
  2) computes

$$V \leftarrow \sum_{i=1}^{n} \Big( [h_{2i}]R_i + [h_{3i}]X_i + S_i + Y_i + [h_{1i}(h_{2i}+1)]P_{\text{pub}} \Big).$$

  3) checks $\hat{e}(\sigma, P) \stackrel{?}{=} \hat{e}(V, Q)$, where $Q = H_4(\triangle)$. If the equation holds, accepts $\sigma$. Otherwise, rejects.

Note that this specification is not well defined. Each signer independently chooses a one-time-use state information $\triangle$, but this state information is used to aggregate individual signatures into the scalar of the common point $Q = H_4(\triangle)$. So, an additional phase is required where each signer has the same $\triangle$. One candidate is to generate individual signatures sequentially. For example, the first signer chooses a state information $\triangle$, and each subsequent signer checks that it has not been used before.

### C. SECURITY CLAIM

To guarantee the security of this CLAS scheme, [1] considers two different games. Game I models the attack where the adversary (denoted $\mathcal{A}_I$) acts as a dishonest user while Game II models the attack where the adversary (denoted $\mathcal{A}_{II}$) acts as a malicious KGC. $\mathcal{A}_I$ is not given $msk$, but can replace user-chosen public key components, extract partial private keys, extract user-chosen secret values, and obtain individual signatures. $\mathcal{A}_{II}$ controls $msk$, but cannot replace user public key components and extract user-chosen secret values. The adversary's goal in both games is to produce a valid aggregate signature of a sequence of selected users where at least one is *honest*, with a meaningful probability. Specifically, the $\mathcal{A}_I$ outputs a tuple $(\sigma^*, \Delta^*, M^*, A^*)$ and wins in Game I if the following conditions hold:

1) $(\sigma^*, \Delta^*, M^*, A^*)$ is valid
2) $\exists ID_j \in W^*$ such that $\mathcal{A}_I$ did not obtain $D_j$ or replace $R_j$ or $S_j$
3) $\mathcal{A}_I$ has not requested an individual signature for $(\Delta^*, m_j^*, ID_j^*, pk_j^*)$,

where $W^*$ is the set of $n$ identities in $A^*$. And $\mathcal{A}_{II}$ wins in Game II if its output $(\sigma^*, \Delta^*, M^*, A^*)$ satisfies the same conditions as in Game I with the following modification of condition 2).

2) $\exists ID_j \in W^*$ such that $\mathcal{A}_{II}$ has not received $(x_j, y_j)$ or replaced $X_j$ or $Y_j$

The authors of [1] provide proof that no adversary can win with a non-negligible probability in either Game I or Game II.

## III. CRYPTANALYSIS

Let $\mathcal{A}$ be an adversary trying to forge an aggregate signature. Without loss of generality, assume that $\mathcal{A}$ has an identity $ID_n$ and receives $D_n = (c_n, d_n, R_n, S_n)$ from the KGC. As a preparatory step, $\mathcal{A}$ chooses a sequence of messages $\{m_i\}_{i=1}^n$ and collects the public keys $\{pk_i\}_{i=1}^{n-1}$ of other $(n-1)$ users with $\{ID_i\}_{i=1}^{n-1}$. Each collected public key $pk_i = (X_i, Y_i, R_i, S_i)$ is assumed to be valid. Then $\mathcal{A}$ sets its own public/private key pair $(pk_n, sk_n)$ as follows:

1) computes $h_{1i} \leftarrow H_1(ID_i, R_i, S_i)$ for each $i \in [1..(n-1)]$.
2) chooses a one-time-use state information $\triangle$, and computes $h_{2i} \leftarrow H_2(m_i, ID_i, pk_i, \triangle)$ and $h_{3i} \leftarrow H_3(m_i, ID_i, pk_i, \triangle)$ for each $i \in [1..(n-1)]$.

3) randomly selects $x_n \in \mathbb{Z}_n^*$, and computes

$$X_n \leftarrow [x_n]P.$$

4) randomly selects $y_n \in \mathbb{Z}_n^*$, and computes

$$Y_n \leftarrow [y_n]P - \sum_{i=1}^{n-1}\Big([h_{2i}]R_i + [h_{3i}]X_i + S_i + Y_i + [h_{1i}(h_{2i}+1)]P_{\text{pub}}\Big).$$

5) sets $pk_n \leftarrow (X_n, Y_n, R_n, S_n)$ and $sk_n \leftarrow (x_n, y_n, c_n, d_n)$.

Using this public/private key pair, $\mathcal{A}$ creates an aggregate signature $\sigma$ on the sequence of messages $\{m_i\}_{i=1}^n$ as the following procedure.

- computes $h_{2n} \leftarrow H_2(m_n, ID_n, pk_n, \triangle)$, $h_{3n} \leftarrow H_3(m_n, ID_n, pk_n, \triangle)$ and $Q \leftarrow H_4(\triangle)$.
- computes

$$\sigma \leftarrow [h_{2n}\cdot c_n + d_n + h_{3n}\cdot x_n + y_n]Q,$$

and outputs $(\sigma, \triangle, \{m_i\}_{i=1}^n, \{(ID_i, pk_i)\}_{i=1}^n)$.

Then $(\sigma, \triangle, \{m_i\}_{i=1}^n, \{(ID_i, pk_i)\}_{i=1}^n)$ is accepted as valid for any verifier because

$$\begin{aligned}\hat{e}(\sigma, P) &= \hat{e}\big([h_{2n}\cdot c_n + d_n + h_{3n}\cdot x_n + y_n]Q, P\big)\\ &= \hat{e}\big(Q, [h_{2n}\cdot c_n + d_n + h_{3n}\cdot x_n + y_n]P\big)\\ &= \hat{e}\big(Q, [h_{2n}\cdot c_n + d_n + h_{3n}\cdot x_n]P + [y_n]P\big)\\ &= \hat{e}\Bigg(Q, \sum_{i=1}^{n}\Big([h_{2i}]R_i + [h_{3i}]X_i + S_i + Y_i + [h_{1i}(h_{2i}+1)]P_{\text{pub}}\Big)\Bigg)\\ &= \hat{e}(Q, V).\end{aligned}$$

This equation holds because

$$\begin{aligned}&[h_{2n}\cdot c_n + d_n + h_{3n}\cdot x_n]P\\ &= [h_{2n}]\big([c_n]P\big) + [d_n]P + [h_{3n}]\big([x_n]P\big)\\ &= [h_{2n}]\big(R_n + [h_{1n}]P_{\text{pub}}\big) + \big(S_n + [h_{1n}]P_{\text{pub}}\big) + [h_{3n}]X_n\\ &= [h_{2n}]R_n + [h_{3n}]X_n + S_n + [h_{1n}]\big([h_{2n}+1]P_{\text{pub}}\big),\end{aligned}$$

and

$$[y_n]P = Y_n + \sum_{i=1}^{n-1}\Big([h_{2i}]R_i + [h_{3i}]X_i + S_i + Y_i + [h_{1i}(h_{2i}+1)]P_{\text{pub}}\Big).$$

Note that $\mathcal{A}$ does not require any secret information about other $n-1$ users or the individual signatures they generate. So, $\mathcal{A}$ wins in Game I described in Subsec. II-C. As a result, we show that anyone can create a valid tuple $(\sigma, \triangle, M, A)$ without contribution of other users. We remark that this type of attack was mentioned in [2] as a "potential attack" on aggregate signatures.

In Deng et al.'s CLAS scheme, the components $X$ and $Y$ of a user public key are determined solely by the owner, and their validity is not certified by any third party. Our attack exploits this feature of the CL-PKC setting, as well as the independence of the individual signatures involved in the aggregation method.

## IV. CONCLUSION

In this paper, we show that Deng et al.'s CLAS scheme is weak against an aggregate signature forgery attack. Deng et al.'s CLAS scheme achieves a short signature, but instead uses public and private keys of relatively large size. However, such a long public key provides additional information that can be exploited by adversaries, leading to the problem described above. We believe that continued research is needed to design secure CLAS schemes that can simultaneously achieve short key and signature sizes.

## REFERENCES

[1] L. Deng, Y. Yang, and Y. Chen, "Certificateless short aggregate signature scheme for mobile devices," *IEEE Access*, vol. 7, pp. 87162–87168, 2019, doi: 10.1109/ACCESS.2019.2923697.

[2] D. Boneh, C. Gentry, B. Lynn, and B. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proc. ASIACRYPT*, in Lecture Notes in Computer Science, vol. 2656, pp. 416–432, doi: 10.1007/3-540-39200-9.