Journal of Information Hiding and Multimedia Signal Processing ©2016 ISSN 2073-4212
Ubiquitous International Volume **7**, Number **6**, November **2016**

Certificate-Based Aggregate Signature Scheme without Bilinear Pairings

Jian-Neng Chen, Qun-Shan Chen and Fu-Min Zou

College of computer, Minnan Normal University
Zhangzhou, Fujian, China, 363000
Beidou Navigation and Smart Traffic Innovation Center of Fujian Province
Fujian Provincial Key Laboratory of Big Data Mining and Applications
Fuzhou, Fujian, China, 350008
jn_chen@mnnu.edu.cn;fmzou@fjut.edu.cn

Received July, 2016; revised August, 2016

Abstract. *The certificate based digital signature technology overcomes the key escrow problem in identity based digital signature technology and the secret key transmission problem in the digital signature technology. The aggregate signature schemes combine a great deal of signatures signed by different signers on different messages into one short signature. However, the more the number of signer and bilinear pairing operations, the lower the efficiency of the aggregate signature scheme. In this paper, we present a scheme of certificate based aggregate signature without bilinear pairings, assuming the hardness of Computational Diffie-Hellman Problem. We then prove the security of the proposed scheme in the random oracle model.*
Keywords: Digital signature; Certificate-based signature; Aggregate signature.

1. **Introduction.** With the development of computer technology, network has become the main way of information storage and transmission. After entering the information age, many scientists have been committed to information security research and achieved fruitful results, such as research on digital watermarking [1], key agreement protocol [2] , digital signature research [3], and Research on the security of cloud computing [4]. Digital signature technology is widely used in E-government, e-commerce and ECcurrency. It consists of three systems: identity based digital signature, certificateless digital signature and certificate based digital signature. The certificate based digital signature technology overcomes the key escrow problem in identity based digital signature technology and the secret key transmission problem in the digital signature technology. The concept of aggregate signatures was first proposed by the Boneh et al. at the European cryptographic International Conference in 2003 [5]. Later Lysyanskaya et al. construct a sequential aggregate signature scheme [6].

Aggregate signature is a relatively new type of public key signature which enables any user to combine n signatures signed by different n signers on different n messages into a short signature [7]. Verifier only needs to verifies the final aggregate signature. This can greatly improve the efficiency of signature verification. However, the time required to generate the signature often increases with the increase of the length of the signature, the number of signatures, and the number of Bilinear Pairings. In recent years, the signature schemes without bilinear pairing had been proposed. Selvi SSD et al. proposed a identity based partial aggregate signature without using bilinear pairing [8].The next year, Jiguo

Li et al. proposed a certificate based digital signature scheme without bilinear pairing [9]. In the year 2015, Asaar et al. introduced an identity-based multi-proxy multi-signature scheme without bilinear pairings [10].Later Kuo Hui et al. proposed a certificateless signature scheme without bilinear pairing [11]. So far, the Certificate-Based aggregate signature scheme signature scheme without bilinear pairing is very seldom.

In this paper, we proposed a certificate-based aggregate signature scheme assuming the hardness of Computational Diffie-Hellman Problem without bilinear pairings.

1.1. **Preliminaries.** Let G be an additive cyclic group of some large prime q. Let P denotes a generator in G. Let P denotes a generator in G. By $x \in Z_q$, we mean picking an element x randomly fromZ_q. By Z_q^* denotes$Z_q/\{0\}$.

1.2. **Complexity Problems.** Computational Diffie-Hellman Problem (CDHP): Given (P, aP, bP), where $a, b \in Z_q^*$, compute abP. In this paper, we assume that there is no polynomial time algorithm which solves the Computational Diffie-Hellman Problem.

2. **Definition of Certificate-Based Aggregate Signature.** Certificate-Based Aggregate Signature (CBAS). A certificate-based aggregate signature scheme, which involves four parties, the Certificate Authority (CA), the signers, the aggregator and the verifier, consists of following algorithms.

Setup. This algorithm takes as input a security parameter 1^k, outputs the master private key s_c, which is kept privately, and the system public parameters *param*, which is shared in the system, such as PK_c .

UserKeyGen. This algorithm takes as inputs the system parameters *param* and users identity information, outputs the users public and private key pair.

CertGen. This algorithm takes as inputs the system parameters *param*, the maser private key s_c, some users identity information ID_i , public key PK_i and a random number, outputs a parameter X_i and a certificate $Cert_i$.

Sign. This algorithm takes as inputs the users private key s_i, X_i, certificate $Cert_i$, message m_i, and a random number, outputs a signature (σ_i, R_i, X_i) on message m_i.

Aggregate. This algorithm takes as inputs the users signatures (σ_i, R_i, X_i) on message m_i, outputs an aggregate signature.

AggVerify. This algorithm takes as inputs the system parameters *param*, users identity information ID_i, messages m_i and aggregate signatures (σ, T_1, T_2,$\cdots T_n$) with users public keys PK_i, returns a bit b. b=1 means that the signature is accepted, whereas b=0 means rejected.

Formally, the security notion of unforgeability is defined in terms of the following games between a challenger C and an adversary $A(A_I, A_{II})$:

Setup: C first initializes a key-pair list UK-List, a certificate list cert-list, tow hash value lists H_1-list and H_2-list as empty. Next, it runs Setup to obtain public parameters *param* and UserKeyGen to obtain a key pair(PK^*, SK^*), and gives PK^* to A.

Hush Query: A adaptively requests a hash value on a string for various hash functions, and receives a hash value.

Certification Query: A adaptively requests the certification of a public key by providing a key pair (PK_i,SK_i) and some users identity information ID_i. Then C returns a Certification $Cert_i$ and X_i to A and adds (ID_i,PK_i,$Cert_i$,X_i) into cert-List.

Signature Query: A adaptively requests a signature by providing a message M to sign under the challenge public key PK', and receives a signature (σ', T').

Output: Finally, A outputs a forged aggregate signature σ^* on messages M^* under public keys PK^*. C outputs 1 if the forged signature satisfies the following three conditions, or outputs 0 otherwise:

1) *AggVerify*(σ^* , $M^*, PK^*, param$)=1,
2) The challenge public key PK^*must exist in PK, and each public key in PK, except the challenge public key must be in UK-List.
3) The corresponding message M^*in M of the challenge public key PK^*must not have been queried by A_I to the signing oracle.

3. **Certificate-Based Aggregate Signature Scheme.** In this section, we propose a certificate-based aggregate signature scheme, as follows:
Setup:
1). Choose (P, q, G) as Section 2. Let G be an additive cyclic group.
2). Chooses the system master key$s \in Z_q^*$ and sets PK_c=sP.
3). Choose hash functions $H_1 : \{0,1\}^* \to G$, $H_2 : \{0,1\}^* \to Z_q^*$.Publish the system parameters $param(P, q, G, PK_C, H_1, H_2)$, and keep the master key s privately.
UserKeyGen: All of the signers choose their private keys $s_i \in_R Z_q^*$, set the public/ private key pairs$(PK_i, SK_i) = (s_iP, s_i)$.
CertGen: All of the signers send their public key PK_i and identity information ID_i to the certifier over an authentic channel. The certifier chooses $x_i \in_R Z_q^*$, computes X_i =x_i P, and returns X_i and a certificate $Cert_i$ =sQ_i to the signers. By the way,$1 \leq i \leq n$, n denotes the total number of signers.
Sign: To generate a signature on message m_i , the signer chooses $r_i \in_R Z_q^*$ and computes

$$R_i = r_iP \tag{1}$$
$$h_i = H_2(m_i//R_i//PK_i//PK_c) \tag{2}$$
$$\sigma_i = s_i + (Cert_i + r_iP)h_iP \tag{3}$$
$$T_i = R_i + X_i \tag{4}$$

and publish(σ_i, T_i)
Aggregate: The aggregator accept the signature if σ_iP=PK_i+(PK_cQ_i+T_i) holds. When all of the signatures from n signers have been collected, the aggregator computes $\sigma = \sum_{i=1}^{n} \sigma_i$, and then $(\sigma, T_1, T_2, ...T_n)$ is a valid certificate-based aggregate signature by n singers on message $M(m_1, m_2, ...m_n)$.
AggVerify To verify a signature $(\sigma, T_1, T_2, \cdots T_n)$ on message $M(m_1, m_2, \cdots m_n)$, the verifier accept the aggregate signature if the follow equation holds.

$$\sigma P = \sum_{i=1}^{n} PK_i + \sum_{i=1}^{n} h_i(PK_cQ_i + T_i) \tag{5}$$

This completes the description of our proposed certificate-based aggregate signature scheme without bilinear parings.

4. **Scheme Analysis.**

4.1. **Correctness.** If$(\sigma, T_1, T_2, \cdots T_n)$ is a valid aggregate signature of message M, we have

$$\begin{aligned} \sigma P &= \sigma_1 + \sigma_2 + \cdots + \sigma_n & (6)\\ &= PK_1 + (PK_cQ_1 + T_1)h_1 + ...PK_n + (PK_cQ_n + T_n)h_n & (7)\\ &= \sum_{i=1}^{n} PK_i + \sum_{i=1}^{n} h_i(PK_cQ_i + T_i) & (8) \end{aligned}$$

4.2. **Unforgeability.** First, we consider the Uncertified User attack, i.e., an adversary A_I tries to get a certificate $Cert_i = sH_1(X_i \| PK_c \| PK_i \| ID_i) + x_i$ where (PK_i, ID_i) has never be asked to CertGen by A_I, This is equivalent to solves Computational Diffie-Hellman Problem and is infeasible.

4.3. **Proof.** Suppose there is an adversary A_I that forges the above aggregate scheme with non-negligible advantage. A simulator C will solves Computational Diffie-Hellman Problem. Then C that interacts with A_I is described as follows:

C first initializes a key-pair list UK-List, a certificate list cert-list, tow hash value lists H_1-list and H_2-list as empty. Next, it runs Setup to obtain public parameters *param* and UserKeyGen to obtain a key pair (PK^*, SK^*) , and gives PK^* to A_I and sets $PK_c = aP$.
UserKeyGen Query: On a new ID_i UserKeyGen query, C chooses a random number $s_i \in Z_q^*$,and sets $(SK_i, PK_i) = (s_i, s_i P)$. Then, he adds (ID_i, SK_i, PK_i) into the UK-list and returns (SK_i, PK_i) to A_I.
H_1 **Query:** On a new H_0 query ω_i, C first chooses a random number $coin_i \in \{0, 1\}$, such that $Pr[coin_i = 1] = \frac{1}{q_E + N}$
1). If $coin_i = 1$, C chooses a random number $d_i \in Z_q^*$ and sets $H_1(\omega_i) = d_i P$.
2). Else $coin_i = 0$, C chooses a random number $b \in Z_q^*$ and sets $H_1(\omega_i) = bP$.
In both cases, C will add $(\omega_i, coin_i, H_1(\omega_i))$ into H_1-List and return $H_1(\omega_i)$ to A_I.
H_2 **Query:** On a new H_2 query ξ_i, C chooses a random number $h_i \in Z_q^*$. Then, he adds(ξ_i, h_i) into H_2-List and returns h_i to A_I.
Certification Query: On a certificate query for ID_i, C first checks the UK-list to obtain this user's public key PK_i. We assume that $(ID_i, PK_i, *)$ has been in H_1-List. Otherwise, C can add $(ID_i, PK_i, coin_i)$ into H_1-List as the same way he responds to H_1 queries.
1). If $coin_i = 1$, which means $Q_i = H_1(X_i \| PK_c \| PK_i \| ID_i)$, C returns the certificate $Cert_i = d_i PK_c + x_i = d_i bP + x_i$ and X_i.
2). Otherwise, C aborts.
Then C adds $(ID_i, PK_i, Cert_i, x_i)$ into cert-List if $coin_i = 1$.
Signature Query: On a sign query (m_i, ID_i, PK_i), C will check UK-list. If $(ID_i, PK_i, *)$ does not exit in UK-list, C will add (ID_i, PK_i, SK_i) into UK-List as the same way he responds to UserKeyGen queries. Then C checks H_1-List to obtain $(ID_i, PK_i, coin_i)$.
1). If $coin_i = 1$, C chooses tow random number r_i and $h_i \in Z_q^*$, and sets $R_i = r_i P$. He further sets the certificate $Cert_i = d_i PK_c + x_i = d_i aP + x_i$ and computes:

$$\sigma_i = s_i + (Cert_i + r_i P) h_i P \tag{9}$$

$$T_i = R_i + X_i \tag{10}$$

Finally he outputs (σ_i, T_i) as the signature.
2).If $coin_i = 0$, C aborts.
Output: Finally , A outputs a forged aggregate signature $(\sigma', T_1', T_2', ...T_n')$ on messages M' under public keys PK', C outputs 1 if the forged signature satisfies the following three conditions, or outputs 0 otherwise:
1). $AggVerify(\sigma', M', PK', param) = 1$
2).The challenge public key PK_j^* must exist in PK',and each public key in PK' must be in UK-List;
3).The corresponding message M_j^* in M' of the challenge public key PK_j^* must not have been queried by A_I to the signing oracle.
We assume that (σ_j, T_j') is a valid signature of (ID_j^*, PK_j^*, m_j^*) , then $\sigma_j = \sigma' - \sum_{i=1, i \neq j}^{n} \sigma_i$.
Therefore, C can compute: $abP = \frac{\sigma_j - s_j}{h_j} - x_i - r_i$

According to the simulation, C can compute the value of abP if and only if all the following three events happen:
E1: C does not fail during the games.
E2: A_I output a valid forgery.
E3: In the forgery output by A_I, $coin_i$=*1*.
Therefore, the probability that C can solve CDH problem is:
$Succ_{CDH} = Pr[E1 \wedge E2 \wedge E3] = Pr[E1]Pr[E2 \mid E1]Pr[E3 \mid E1 \wedge E2]$
From the games, we have Pr[E1]$\geq (1-\frac{1}{q_E+N})^{q_E}$, $Pr[E2 \mid E1]$=$Succ_{A_I}$ and $Pr[E3 \mid E1 \wedge E2] \geq (1-\frac{1}{q_E+N})^{q_E}\frac{1}{q_E+N}$. Thus,
$Succ_{CDH}$=$(1-\frac{1}{q_E+N})^{q_E} \cdot \varepsilon \cdot (1-\frac{1}{q_E+N})^{q_E}\frac{1}{q_E+N} \geq \frac{\varepsilon}{e^2(q_E+N)}$
Next we consider the certifier attack, an adversary A_{II}, who knows each signers certificate and each signers private key in addition to the target user, tries to forge a signature $(\sigma, T_1, T_2, ...T_n)$.

Proof: Suppose there exists an adversary A_{II} that forges the above aggregate scheme with non-negligible advantage, A simulator C will solves Computational Diffie-Hellman Problem. Then C that interacts with A_{II} is described as follows:

C first initializes a key-pair list UK-List, a certificate list cert-list, tow hash value lists H_1-list and H_2-list as empty. Next, it runs Setup to obtain public parameters *param* and CertGen to obtain $Cert_i$ and gives them to A_{II}.Without loss of generality, C chooses ID_l,$l \in (1,n)$as the challenge identity.

UserKeyGen Query: On a new UserKeyGen query ID_i, if $ID_i \neq ID_l$, C selects a random number $s_i \in Z_q^*$ and runs UserKeyGen to obtain a key pair (SK_i, PK_i)=(s_i, s_iP)Then, he adds into the UK-list and returns (SK_i, PK_i) to A_{II}.
If ID_i=ID_l, C gives PK_l to A_{II}.
H_1 **Query:**On a new H_1 query (X_i,PK_c,PK_i,ID_i), C chooses a random number $Q_i \in Z_q^*$. Then, he adds (X_i,PK_c,PK_i,ID_i,Q_i) into H_1-List and returns Q_i to A_{II}.
H_2 **Query:** On a new H_2 query (m_i,R_i,PK_i,PK_c),
1).If $PK_i \neq PK_l$, C chooses a random number $d_i \in Z_q^*$ and sets h_i=H_2(m_i,R_i,PK_i,PK_c)=d_iP.
2).Else PK_i=PK_l, C chooses a random number $b \in Z_q^*$ and sets h_i=H_2(m_i,R_i),PK_c,PK_i)=bP.
C will add (m_i,R_i),PK_c,PK_i,h_i) into H_2-List and return h_i to A_{II} in both cases.
Certification Query: On a certificate query ID_i , C first checks the UK-list to obtain this users public key PK_i. We assume that (ID_i ,PK_i,*) has been in H_1-List. Otherwise, C can add (ID_i ,PK_i,$Coin_i$) into H_1-List as the same way he responds to H_1 queries. Then C chooses a random number $x_i \in Z_q^*$ and sets X_i=x_iP. After that, C returns A_{II} the certificate $Cert_i$=$sQ_i + x_i$ and X_i Finally C adds (ID_i ,PK_i, $Cert_i$, x_i) into cert-List.
Signature Query: On a sign query (m_i, ID_i, PK_i), C will check UK-list. If $(ID_i, PK_i, *)$ does not exit in UK-list, C will add (ID_i, PK_i, SK_i) into UK-List as the same way he responds to UserKeyGen queries. Then C checks cert-List to obtain certificate $Cert_i$=$sQ_i + x_i$.
1). If $ID_i \neq ID_l$, C chooses a random number $r_i \in Z_q^*$, and sets $R_i = r_iP$. He computes: σ_i=s_i+($Cert_i$+r_i)h_i, T_i=R_i+X_i
Finally he outputs (σ_i,T_i) as the signature.
2).If ID_i=ID_l, C chooses a random number $a \in Z_q^*$, and sets R_i=aP. He computes: σ_i=($Cert_i$+a)h_i, T_i=R_i+X_i - $PK_ih_i^{-1}$
Finally he outputs (σ_i,T_i) as the signature.
Output: Finally, A outputs a forged aggregate signature $(\sigma', T_1', T_2', ...T_n')$ on messages M' under public keys PK', C outputs 1 if the forged signature satisfies the following three conditions, or outputs 0 otherwise:
1).*AggVerify*(σ', M', PK', *param*)=1

2). The challenge public key PK_j^* must exist in PK',and each public key in PK', except the challenge public key must be in UK-List,
3). The corresponding message M_j^* in M' of the challenge public key PK_j^* must not have been queried by A_{II} to the signing oracle.

We assume that (σ_j, T_i') is a valid signature of (ID_j^*,PK_j^*,m_j^*), then

$$\sigma_j = \sigma' - \sum_{i=1, i\neq j}^{n} \sigma_i \tag{11}$$

Therefore, C can compute $abP = \sigma_j - (sQ_i+x_i)h_i$

According to the simulation, C can compute the value of abP if and only if all the following three events happen:
E1: C does not fail during the games.
E2: A_{II} output a valid forgery.
E3: In the forgery output by A_{II}, The challenge public key PK_j^* exist in PK'.

Therefore, the probability that C can solve CDH problem is:
$Succ_{CDH} = Pr[E1 \wedge E2 \wedge E3] = Pr[E1]Pr[E2 \mid E1]Pr[E3 \mid E1 \wedge E2]$

From the games, we have $Pr[\text{E1}] \geq (1-\frac{1}{N})^{q_E}$, $Pr[E2 \mid E1]{=}Succ_{A_{II}}$ and
$Pr[E3 \mid E1 \wedge E2] \geq (1-\frac{1}{N})^{q_E}\frac{1}{N}$. Thus,
$Succ_{CDH} \geq (1-\frac{1}{N})^{q_E}\frac{\varepsilon}{N}$

Therefore, our scheme is impossible to fake under the hardness assumption of Computational Diffie-Hellman Problem.

4.4. **Performance analysis.** In this section, we analyze the efficiency of the proposed scheme. To the best of our knowledge, the proposed scheme is the first certificate aggregate scheme proposed without bilinear pairings. Therefore, we compare our scheme with Kwangsu, Dong Hoon and Motis scheme. The comparison is performed in terms of computation complexity.

The results indicate the efficiency of the proposed method. We summarize the results in Table 1 where the following notations are used:
T_G: computation time for a multiplication in a multiplicative group or an addition in an additive group.
T_{Exp}: computation time for an exponentiation in a multiplicative group (like G_2).
T_{BP}: computation time of one bilinear pairing operation.
T_h: computation time of one hash operation.
n: the number of signers.

Table 1. Comparsion of Aggregate Signature Schemes

Schemes	Sign	Verify
LLY13[5]	$8T_{BP}+5nT_{Exp}$	$8T_{BP}+4nT_{Exp}$
ours	$5nT_G+nT_h$	$(4n+1)T_G$

5. **Conclusions.** Certificate-based public key technology was introduced to remove the use of certificate to ensure the authentication of the users public key in the traditional cryptography and overcome the key escrow problem in the identity-based public key signature. Aggregate signature enables any user to combine n signatures signed by different n signers on different n messages into a short signature. Combining the concept of certificate-based signature with the concept of aggregate signature, in this paper, we

present a certificate-based aggregate signature scheme without bilinear pairing and proved the security under the Computational Diffie-Hellman Problem assumption.

Acknowledgment. This work is partially supported by the National Natural Science Foundation of China (No.61304199) and Fujian Science and Technology Department(No.20 13HZ0002-1,No.2014H0008); the Natural Science Foundation of Fujian Provincial Department of Education under Grant No.JAT160306.The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers,which have improved the presentation.

REFERENCES

[1] J. S. Pan and P. Shi, Guest Editorial: Digital Watermarking and Multimedia Security, *Circuits Systems and Signal Processing*, vol. 27, pp. 133-136, 2008.
[2] C. M. Chen, L. L. Xu, T. Y. Wu, and C. R. Li, On the Security of a Chaotic Maps-based Three-party Authenticated Key Agreement Protocol , *Journal of Network Intelligence*, vol. 1, no. 2, pp. 61-66, May 2016.
[3] J. Q. Li, X. Y. Huang, Y. C. Zhang, L. Z. Xu, An efficient short certificate-based signature scheme, *The Journal of Systems and Software*, vol. 85, pp. 314-322, 2012.
[4] J. S. Pan and Tsu-Yang Wu, Special Issue on Cloud Security, *Journal of Computers*, vol. 24, no. 3, pp.1-2, 2013.
[5] D. Boneh, C. Gentry, B. Lynn, H. Shacham, Aggregate and verifiably encrypted signatures from bilinear maps, *European Cryptology Conferenc 2003*, ed. by E. Biham. LNCS, vol. 2656, pp. 416C432,2003.
[6] Lysyanskaya, S. Micali, L. Reyzin, H. Shacham, Sequential aggregate signatures from trapdoor permutations, *European Cryptology Conferenc 2004*, ed. by C. Cachin, J. Camenisch. vol. 3027, pp. 74C90, 2004.
[7] K. Lee, D. H. Lee, M. Yung, Sequential Aggregate Signatures with Short Public Keys: Design, Analysis and Implementation Studies, *Public-Key CryptographyCPKC 2013*.Lecture Notes in Computer Science,vol. 7778, pp. 423-442, 2013.
[8] J. Q. Li, Z. W. Wang, Y. C. Zhang, Provably secure certificate-based signature scheme without pairings, *Information Sciences*, vol.233, pp. 313-320, 2013.
[9] S.S.D. Selvi, S.S. Vivek Shriram, C.P. Rangan, Identity based partial aggregate signature scheme without pairing, *IEEE Sarnoff Symposium*, vol. 2010, pp. 1-6, 2012.
[10] M. R. Asaar, M. Salmasizadeh,W. Susilo, An Identity-Based Multi-Proxy Multi- Signature Scheme Without Bilinear Pairings and its Variants, *The Computer Journal*, vol.58, pp.1021-1039, 2015.
[11] K. H. Yeh, K. Y. Tsai, C. Y. Fan, An efficient certificateless signature scheme without bilinear pairings, *Multimedia Tools and Applications*, vol. 74, pp. 6519-6530, 2015.