Short and Efficient Certificate-Based Signature *

Joseph K. Liu, Feng Bao, and Jianying Zhou

Cryptography and Security Department,
Institute for Infocomm Research, Singapore
{ksliu, baofeng, jyzhou}@i2r.a-star.edu.sg

Abstract. In this paper, we propose a short and efficient certificate-based signature (CBS) scheme. Certificate-based cryptography proposed by Gentry [6] combines the merit of traditional public key cryptography (PKI) and identity based cryptography, without use of the costly certificate chain verification process and the removal of key escrow security concern. Under this paradigm, we propose the shortest certificate-based signature scheme in the literature. We require one group element for the signature size and public key respectively. Thus the public information for each user is reduced to just one group element. It is even shorter than the state-of-the-art PKI based signature scheme, which requires one group element for the public key while another group element for the certificate. Our scheme is also very efficient. It just requires one scalar elliptic curve multiplication for the signing stage. Our CBS is particularly useful in power and bandwidth limited environment such as Wireless Cooperative Networks.

1 Introduction

Different Cryptosystems: Public Key Infrastructure (PKI). In a traditional public key cryptography (PKC), a user Alice signs a message using her private key. A verifier Bob verifies the signature using Alice's public key. However, the public key is just merely a random string and it does not provide authentication of the signer by itself. This problem can be solved by incorporating a certificate generated by a trusted party called the Certificate Authority (CA). The hierarchical framework is called the public key infrastructure (PKI). Prior to the verification of a signature, Bob needs to obtain Alice's certificate in advance and verify the validity of her certificate. If it is valid, Bob extracts the corresponding public key which is then used to verify the signature. This approach seems inefficient, in particular when the number of users is large.

Identity-Based cryptography (IBC). Identity-based cryptography (IBC) solves the aforementioned problem by using Alice's identity (or email address) which is an arbitrary string as her public key while the corresponding private key is a result of some mathematical operation that takes as input the user's identity and the master secret key of a trusted authority, referred to as "Private

* The work is supported by A*STAR project SEDS-0721330047.

Key Generator (PKG)". The main disadvantage of identity-based cryptography is an unconditional trust to the PKG. Hence, IBC is only suitable for a closed organization where the PKG is completely trusted by everyone in the group.

Certificate-Based cryptography (CBC). To integrate the merits of IBC into PKI, Gentry [6] introduced the concept of certificate-based encryption (CBE). A CBE scheme combines a public key encryption scheme and an identity based encryption (IBE) scheme between a certifier and a user. Each user generates his/her own private and public keys and requests a certificate from the CA while the CA uses the key generation algorithm of an IBE [4] scheme to generate the certificate from the user identity and his public key. The certificate is implicitly used as part of the user decryption key, which is composed of the user-generated private key and the certificate. Although the CA knows the certificate, it does not have the user private key. Thus it cannot decrypt any ciphertexts. In addition to CBE, the notion of certificate-based signature (CBS) was first suggested by Kang *et al.* [9]. In a CBS scheme, the signing process requires both the user private key and his certificate, while verification only requires the user public key.

Certificateless cryptography (CLC). Certificateless cryptography [1] is another stream of research, which is to solve the key escrow problem inherited by IBC. In a CLC, the CA generates a *partial secret key* from the user's identity using the master secret key, while the user generates his/her own private and public keys which are independent to the partial secret key. Decryption and signature generation require both the user private key and partial secret key. The concept is similar to CBC. The main different is the generation of the partial secret key in CLC (it is called *certificate* in CBC). Here the CA does not require the user's public key for the generation of the partial secret key, while in CBC the CA does require the user's public key for the generation of the certificate.

Level of Trust: Girault [7] defined three levels of trust of a PKG or CA:
Level 1: The PKG can compute users' secret keys and, therefore, can impersonate any user without being detected. ID-based signatures belong to this level.
Level 2: The PKG cannot compute users' secret keys. However, the PKG can still impersonate any user without being detected. Certificateless signatures belong to this level.
Level 3: The PKG cannot compute users' secret keys, and the PKG cannot impersonate any user without being detected. This is the highest level of trust. Traditional PKI signatures and certificate-based signatures belong to this level.

We summarize different kinds of cryptosystems in table 1. We compare the public information (per user) and the level of trust between different cryptosystems. "ID" represents the identity of the user. "PK" represents the public key of the user. "Cert" represents the certificate. Note that although certificate-based cryptosystem also requires a certificate, that one is *not* public information. In contrast, it is part of secret information and should be kept secret by the user.

Cryptosystem	Public Information	Level of Trust
Traditional PKI	ID, PK, Cert	3
ID-based	ID	1
Certificateless	ID, PK	2
Certificate-Based	ID, PK	3

Table 1. Comparison of different cryptosystems

Application of CBS: CBS can be very useful in many scenarios. It can provide the highest level of trust, while also maintaining the shortest length and most efficient verification. It is especially useful in those environments such that computation power is very limited, or communication bandwidth is very expensive. Wireless Cooperative Networks (WCN) including wireless sensor networks or mesh networks are good examples of such environments. The efficient verification can let the node to enjoy a reduced complexity or energy consumption. On the other side, the elimination of certificate in the verification provides a shorter length of total transmitted data. In the case of WCNs, communication bandwidth is a very expensive resource. A shorter length means a cheaper cost to provide the same level of service without compromising security.

Contribution: We propose the shortest certificate-based signature (CBS). The signature size and public key of our scheme are both of one element respectively, which is around 160 bits. All previous CBS schemes [9, 10, 3, 11] require at least two elements for the signature size, except for the construction by aggregated BLS signature scheme [5][1]. Our scheme is comparable to the shortest PKI-based signature [5] which also contains one element for both the public key and signature size. However, as a PKI-based signature, a certificate should be attached to the public key. A certificate itself is a signature (signed by the CA). Thus the total length of the public information (public key + certificate) should be at least two elements. Our scheme only requires one element for both the signature and public information while maintaining the same level of trust as other traditional PKI-based signatures. In addition, our scheme requires two pairings for the verification, while the CBS from aggregated BLS requires three pairings. We also note that the CBS from aggregated BLS signature has not been formally proven secure, though we still include it in our efficiency comparison. Our scheme can be proven secure in the random oracle model (ROM), using the weak modified k-CAA assumption. The contribution of our scheme is summarized in table 2, by comparing with other most efficient state-of-the-art schemes of different cryptosystems.

[1] It is easy to observe that by using an aggregated BLS signature, one can get a CBS. The CA generates the certificate as a BLS signature. The user signs the message and his public key using his own secret key to generate another BLS signature. These two signatures are aggregated together to form the final signature. In this way, the length of the final signature is still one element.

Cryptosystem	State-of -the-art scheme	Number of pairing (veri.)	Security Model	Length of Public Info. (per user)	Length of Sign.	Level of Trust
PKI	[5]	4	ROM	320 bits + ID	160 bits	3
ID-based	[8]	2	ROM	ID	320 bits	1
Certificateless	[14]	2	ROM	320 bits + ID	160 bits	2
Certificate Based	Agg. BLS	3	not formally proven	160 bits + ID	160 bits	3
Certificate Based	our prop. scheme	2	ROM	160 bits + ID	160 bits	3

Table 2. Comparison of different signature schemes with different cryptosystems

2 Preliminaries and Security Model

2.1 Mathematical Assumptions

Definition 1 (k-Collusion Attack Algorithm Assumption (k-CAA)). *[12] The k-CAA problem in $\mathbb{G}$ is defined as follow: For some x, $h_1, \cdots, h_k \in \mathbb{Z}_p^*$ and $g \in \mathbb{G}$, given g, g^x and k pairs $(h_1, g^{(x+h_1)^{-1}}), \ldots, (h_k, g^{(x+h_k)^{-1}})$, output a new pair $(h^*, g^{(x+h^*)^{-1}})$ for some $h^* \notin \{h_1, \ldots, h_k\}$. We say that the (t, ϵ) k-CAA assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the k-CAA problem in $\mathbb{G}$.*

Definition 2 (Weak Modified k-CAA Assumption[2]). *The Weak Modified k-CAA problem in $\mathbb{G}$ is defined as follow: For some x, a, b, $h_1, \cdots, h_k \in \mathbb{Z}_p^*$ and $g \in \mathbb{G}$, given g, g^x, g^a, g^b and k pairs $(h_1, (g^{ab})^{(x+h_1)^{-1}}), \ldots, (h_k, (g^{ab})^{(x+h_k)^{-1}})$, output either a new pair $(h^*, (g^{ab})^{(x+h^*)^{-1}})$ for some $h^* \notin \{h_1, \ldots, h_k\}$, or g^{ab}. We say that the (t, ϵ) Weak Modified k-CAA assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the Weak Modified k-CAA problem in $\mathbb{G}$.*

2.2 Security Model

Definition 3. *A certificate-based signature (CBS) scheme is defined by six algorithms:*

- *Setup is a probabilistic algorithm taking as input a security parameter. It returns the certifier's master key msk and public parameters $param$. Usually this algorithm is run by the CA.*
- *UserKeyGen is a probabilistic algorithm that takes $param$ as input. When run by a client, it returns a public key PK and a secret key usk.*

[2] We define a weaker version of the Modified k-CAA Assumption [14]. The only difference in this version is that, we do not require g^{bx} as input. It is defined as follow:

- *Certify is a probabilistic algorithm that takes as input* $(msk, \tau, param, PK, ID)$ *where* ID *is a binary string representing the user information. It returns* $Cert'_\tau$ *which is sent to the client. Here* τ *is a string identifying a time period.*
- *Consolidate is a deterministic certificate consolidation algorithm taking as input* $(param, \tau, Cert'_\tau)$ *and optionally* $Cert_{\tau-1}$*. It returns* $Cert_\tau$*, the certificate used by a client in time period* τ*.*
- *Sign is a probabilistic algorithm taking as input* $(\tau, param, m, Cert_\tau, usk)$ *where* m *is a message. It outputs a ciphertext* σ*.*
- *Verify is a deterministic algorithm taking* $(param, PK, ID, \sigma)$ *as input in time period* τ*. It returns either* valid *indicating a valid signature, or the special symbol* $\perp$ *indicating invalid.*

We require that if σ *is the result of applying algorithm* Sign *with intput* $(\tau, param, m, Cert_\tau, usk)$ *and* (usk, PK) *is a valid key-pair, then* valid *is the result of applying algorithm* Verify *on input* $(param, PK, ID, \sigma)$*, where* $Cert_\tau$ *is the output of* Certify *and* Consolidate *algorithms on input* $(msk, param, \tau, PK)$*. That is, we have* $\mathsf{Verify}_{PK,ID}(\mathsf{Sign}_{\tau,Cert_\tau,usk}(m)) = \mathsf{valid}$ *We also note that a concrete CBS scheme may not involve certificate consolidation. In this situation, algorithm* Consolidate *will simply output* $Cert_\tau = Cert'_\tau$*.*

In the rest of this paper, for simplicity, we will omit Consolidate and the time identifying string τ in all notations.

The security of CBS is defined by two different games and the adversary chooses which game to play. In Game 1, the adversary models an uncertified entity while in Game 2, the adversary models the certifier in possession of the master key msk attacking a fixed entity's public key. We use the enhanced model by Li *et al.* [10] which captures key replacement attack in the security of Game 1.

Definition 4 (CBS Game 1 Existential Unforgeable). *The challenger runs* Setup*, gives* $param$ *to the adversary* $\mathcal{A}$ *and keeps* msk *to itself. The adversary then interleaves certification and signing queries as follows:*

- *On user-key-gen query* (ID)*, if* ID *has been already created, nothing is to be carried out. Otherwise, the challenger runs the algorithm* UserKeyGen *to obtains a secret/public key pair* (usk_{ID}, PK_{ID}) *and adds to the list* L*. In this case,* ID *is said to be 'created'. In both cases,* PK_{ID} *is returned.*
- *On corruption query* (ID)*, the challenger checks the list* L*. If* ID *is there, it returns the corresponding secret key* usk_{ID}*. Otherwise nothing is returned.*
- *On certification query* (ID)*, the challenger runs* Certify *on input* $(msk, param, PK, ID)$*, where* PK *is the public key returned from the user-key-gen query, and returns* $Cert$*.*
- *On key-replace query* (ID, PK, usk)*, the challenger checks if* PK *is the public key corresponding to the secret key* usk*. If yes, it updates the secret/public key pair to the list* L*. Otherwise it outputs* $\perp$ *meaning invalid operation.*

– *On signing query* (ID, PK, m)*, the challenger generates* σ *by using algorithm* Sign.

Finally $\mathcal{A}$ *outputs a signature* σ^**, a message* m^* *and a public key* PK^* *with user information* ID^**. The adversary wins the game if (1)* σ^* *is a valid signature on the message* m^* *under the public key* PK^* *with user information* ID^**, where* PK^* *is either the one returned from user-key-gen query or a valid input to the key-replace query. (2)* ID^* *has never been submitted to the certification query. (3)* (ID^*, PK^*, m^*) *has never been submitted to the signing query.*

We define $\mathcal{A}$*'s advantage in this game to be* $Adv(\mathcal{A}) = \Pr[\mathcal{A}$ *wins* $]$*.*

Definition 5 (CBS Game 2 Existential Unforgeable). *The challenger runs* Setup*, gives param and msk to the adversary* $\mathcal{A}$*. The adversary interleaves user-key-gen queries, corruption queries and signing queries as in Game 1. But different from Game 1, the adversary is not allowed to replace any public key.*

Finally $\mathcal{A}$ *outputs a signature* σ^**, a message* m^* *and a public key* PK^* *with user information* ID^**. The adversary wins the game if (1)* σ^* *is a valid signature on the message* m^* *under the public key* PK^* *with user information* ID^**. (2)* PK^* *is an output from user-key-gen query. (3)* ID^* *has never been submitted to corruption query. (4)* (ID^*, PK^*, m^*) *has never been submitted to the signing query.*

We define $\mathcal{A}$*'s advantage in this game to be* $Adv(\mathcal{A}) = \Pr[\mathcal{A}$ *wins* $]$*.*

We note that our model does not support security against *Malicious Certifier*. That is, we assume that the certifier generates all public parameters honest, according to the algorithm specified. The adversarial certifier is only given the master secret key, instead of allowing to generate all public parameters. Although malicious certifier has not been discussed in the literature, similar concept of *Malicious Key Generation Centre (KGC)* [2] has been formalized in the area of certificateless cryptography.

3 The Proposed Scheme

3.1 Construction

Setup. Select a pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ where the order of $\mathbb{G}$ is p. Let g be a generator of $\mathbb{G}$. Let $H_1 : \{0,1\}^* \to \mathbb{G}$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_p$ be two collision-resistant cryptographic hash functions. Randomly select $\alpha \in_R \mathbb{Z}_p$ and compute $g_1 = g^\alpha$. The public parameters param are $(e, \mathbb{G}, \mathbb{G}_T, p, g, g_1)$ and the master secret key msk is α.

UserKeyGen. User selects a secret value $x \in \mathbb{Z}_p$ as his secret key usk, and computes his public key PK as $Y = g^x$.

Certify. To construct the certificate for user with public key PK and binary string ID, the CA computes $C = H_1(ID, PK)^\alpha$.

Sign. To sign a message $m \in \{0,1\}^*$, the signer with public key PK (and user information ID) , certificate C and secret key x, compute $\sigma = C^{\frac{1}{x+H_2(m,ID,PK)}}$

Verify. Given a signature σ for a public key PK and user information ID on a message m, a verifier checks whether $e(\sigma, Y \cdot g^{H_2(m,ID,PK)}) \stackrel{?}{=} e(H_1(ID, PK), g_1)$ Output valid if it is equal. Otherwise output $\perp$.

3.2 Analysis

For efficiency, the computation requirement of our scheme is very light. In the signing stage, we just require one exponentiation, which is the minimum among every signature scheme (except for online/offline signature schemes [13]). For verification, we require two pairing operations, which is generally acceptable as verification is usually done in relatively powerful device such as computer or PDA. Our scheme can be implemented in many devices, such as personal computers, PDAs, mobile phones, wireless sensors or smart cards.

For security, our scheme can be proven secure by the following theorems:

Theorem 1 (Game 1). *The CBS scheme proposed in this section is (ϵ, t)-existential unforgeable against Game 1 adversary with advantage at most ϵ and runs in time at most t, assuming that the (ϵ', t')-Weak Modified k-CAA assumption holds in $\mathbb{G}$, where $\epsilon' \geq (1 - \frac{1}{q_k})^{q_e+q_r}(1 - \frac{1}{q_s+1})^{q_s} \frac{1}{(q_s+1)q_k}\epsilon$, $t' = t$ where q_k, q_r, q_e, q_s are the numbers of queries made to the User-key-gen Query, Corruption Query, Certification Query and Signing Query respectively.*

Proof. Assume there exists a Game 1 adversary $\mathcal{A}$. We construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the Weak Modified k-CAA problem with probability at least ϵ' and in time at most t'. We use a similar approach as in [14].

Let g be a generator of $\mathbb{G}$. Let $x, a, b \in \mathbb{Z}_p^*$ be three integers and $h_1, \ldots, h_k \in \mathbb{Z}_p^*$ be k integers. $\mathcal{B}$ is given the problem instance $g, g^x, g^a, g^b, \left(h_1, (g^{ab})^{(x+h_1)^{-1}}\right)$, $\ldots, \left(h_k, (g^{ab})^{(x+h_k)^{-1}}\right)$. $\mathcal{B}$ is asked to output g^{ab} or $\left(h^*, (g^{ab})^{(x+h^*)^{-1}}\right)$ for some $h^* \notin \{h_1, \ldots, h_k\}$.

Setup. $\mathcal{B}$ simulates the environment and the oracles which $\mathcal{A}$ can access. We regard the hash function H_1 and H_2 as random oracles. $\mathcal{B}$ first sets $g_1 = g^a$ and gives the parameter $\mathsf{param} = (e, \mathbb{G}, \mathbb{G}_T, p, g, g_1)$ to $\mathcal{A}$.

Oracles Simulation. $\mathcal{B}$ simulates all oracles as follow:

(User-key-gen (and H_1 random oracle) Query .) At the beginning $\mathcal{B}$ first randomly chooses a number $t \in \{1, \ldots, q_k\}$. $\mathcal{B}$ also maintains a list L_U which stores the secret/public key pair of the corresponding identity. $\mathcal{A}$ can query this oracle by submitting an identity ID_i. If the public key pair is already created, $\mathcal{B}$ retrieves them from the list L_U. Otherwise,

1. If $i \neq t$, $\mathcal{B}$ chooses $d_i, x_i \in_R \mathbb{Z}_p^*$ and sets $PK_i = g^{x_i}$, $H_1(ID_i, PK_i) = g^{d_i}$. The corresponding certificate is $C_{ID_i} = H_1(ID_i, PK_i)^a = (g^a)^{d_i}$ and the user secret key is x_i. $\mathcal{B}$ stores (d_i, x_i, PK_i, ID_i) to the list L_U.
2. If $i = t$, $\mathcal{B}$ sets $PK_i = g^x$ and $H_1(ID_i, PK_i) = g^b$. In this case, the certificate and the user secret key are unknown to $\mathcal{B}$.

(Corruption Query.) Given an identity ID_i, $\mathcal{B}$ outputs $\perp$ if $ID_i = ID_t$ or ID_i has not been created. Otherwise, $\mathcal{B}$ returns x_i from the list L_U.

(Certification Query.) Given an identity ID_i, $\mathcal{B}$ outputs $\perp$ if $ID_i = ID_t$ or ID_i has not been created. Otherwise, $\mathcal{B}$ returns g^{ad_i} where d_i is retrieved from the list L_U.

(Key-replace Query.) Given an identity ID_i, a new public key pair PK_i, usk_i, $\mathcal{B}$ checks whether PK_i is the corresponding public key of usk_i. If yes, $\mathcal{B}$ replaces the new public key pair into the list L_U. Otherwise, outputs $\perp$.

(Signing (and H_2 random oracle) Query.) We first describe the simulation of H_2 oracle. On the i-th input $\omega_i = (m_\ell, ID_j, PK_k)$, $\mathcal{B}$ first checks if $ID_j = ID_t$ and $PK_k = PK_t$. Here PK_t is the original public key. We divide into two cases:

1. If $ID_j = ID_t$ and $PK_k = PK_t$, then $\mathcal{B}$ flips a biased coin which outputs a value $c_i = 1$ with probability ξ, and $c_i = 0$ with probability $1 - \xi$. The value of ξ will be optimized later.
   - If $c_i = 1$, $\mathcal{B}$ randomly chooses $h'_i \in_R \mathbb{Z}_p^*$ where $h'_i \notin \{h_1, \ldots, h_k\}$ and returns h'_i to $\mathcal{A}$ as the value of $H_2(\omega_i)$.
   - If $c_i = 0$, $\mathcal{B}$ randomly chooses $h''_i \in_R \{h_1, \ldots, h_k\}$ as the output of $H_2(\omega_i)$, where h''_i must be a fresh value which has not been assigned as an output of H_2 queries before.
2. Otherwise, $\mathcal{B}$ randomly chooses $\mu_i \in_R \mathbb{Z}_p^*$ as the output.

In all cases, $\mathcal{B}$ records $(\omega_i, h'_i, c_i), (\omega_i, h''_i, c_i)$ or (ω_i, μ_i) to another list L_{H_2}.

For each signing query on an input (m_ℓ, ID_j, PK_j), output $\perp$ if ID_j has not been created. Otherwise, we assume that $\mathcal{A}$ has already queried the random oracle H_2 on the input $\omega_i = (m_\ell, ID_j, PK_j)$. We divide into two cases:

1. If $ID_j \neq ID_t$, $\mathcal{B}$ uses the the user secret key and certificate of ID_j from the list L_k and $\mu_i = H_2(\omega_i)$ from the list L_{H_2} to generate a valid signature.
2. If $ID_j = ID_t$, $\mathcal{B}$ first checks the list L_{H_2}.
   - If $c_i = 1$, $\mathcal{B}$ reports *failure* and terminates the simulation.
   - Otherwise, that is $c_i = 0$ and $h''_i = H_2(m_\ell, ID_t, PK_t)$ is on the list L_{H_2}. For simplicity, we assume $h''_i = h_i \in \{h_1, \ldots, h_k\}$. $\mathcal{B}$ outputs $\sigma_i = (g^{ab})^{(x+h_i)^{-1}}$ as the signature. It is a valid signature because $e(\sigma_i, PK_t \cdot g^{h_i}) = e((g^{ab})^{(x+h_i)^{-1}}, g^x \cdot g^{h_i}) = e(g^{ab}, g) = e(g^b, g^a) = e(H_1(ID_t, PK_t), g_1)$

Forgery. After all queries, $\mathcal{A}$ outputs $(ID^*, PK^*, m^*, \sigma^*)$. If $\mathcal{A}$ wins, we have the following conditions satisfied:

1. ID^* has not been submitted to certification query.
2. (ID^*, PK^*, m^*) has not been submitted to signing query.
3. σ^* is a valid signature on (ID^*, PK^*, m^*). That is,

$$
\begin{aligned}
e(\sigma^*, PK^* \cdot g^{h^*}) &= e(H_1(ID^*, PK^*), g_1) \\
e(\sigma^*, g^{x^*} \cdot g^{h^*}) &= e(g^b, g^a) \\
e(\sigma^*, g) &= e((g^{ab})^{(x^*+h^*)^{-1}}, g) \\
\Rightarrow \quad \sigma^* &= (g^{ab})^{(x^*+h^*)^{-1}}
\end{aligned}
$$

where $h^* = H_2(m^*, ID^*, PK^*)$ and $PK^* = g^{x^*}$.

4. PK^* is either the original public key generated by the User-key-gen oracle, or a new public key inputted by $\mathcal{A}$ to the key-replace oracle successfully. Note that in the latter case, $\mathcal{B}$ also knows the corresponding secret key.

If $ID^* \neq ID_t$, then $\mathcal{B}$ outputs *failure* and terminates the simulation. Otherwise, $\mathcal{B}$ checks the list L_{H_2}. If $c^* = 0$, $\mathcal{B}$ outputs *failure* and terminates the simulation. Otherwise, that is, $c^* = 1$ and $h^* \notin \{h_1, \ldots, h_k\}$. We divide into two cases:

1. PK^* is the original public key generated by user-key-gen oracle. In this case, $\mathcal{B}$ outputs a new pair $(h^*, \sigma^*) = (h^*, (g^{ab})^{(x+h^*)^{-1}})$ which is the solution of the modified k-CAA problem.
2. PK^* is a new public key inputted by $\mathcal{A}$ to the key-replace oracle successfully. In this case, $\mathcal{B}$ retrieves the corresponding secret key x^* from the list L_k. $\mathcal{B}$ computes $(\sigma^*)^{(x^*+h^*)} = \left((g^{ab})^{(x^*+h^*)^{-1}}\right)^{(x^*+h^*)} = g^{ab}$ as the solution of the Weak Modified k-CAA problem.

Probability Analysis. $\mathcal{B}$ succeeds if: (1) E_1: $\mathcal{B}$ does not abort during simulation; (2) E_2: $\mathcal{A}$ wins; and (3) E_3: $ID^* = ID_t$ and $c^* = 1$.

The advantage of $\mathcal{B}$ is $\epsilon' = \Pr[E_1 \wedge E_2 \wedge E_3] = \Pr[E_1] \cdot \Pr[E_2|E_1] \cdot \Pr[E_3|E_1 \wedge E_2]$ If E_1 happens, then:

- $\mathcal{B}$ does not output *failure* during the simulation of the certification query. This happens with probability $(1 - \frac{1}{q_k})^{q_e}$.
- $\mathcal{B}$ does not output *failure* during the simulation of the corruption query. This happens with probability $(1 - \frac{1}{q_k})^{q_r}$.
- $\mathcal{B}$ does not output *failure* during the simulation of the signing query. This happens with probability $(1 - \frac{1}{q_k}\xi)^{q_s} \geq (1 - \xi)^{q_s}$.

Now we have $\Pr[E_1] \geq (1 - \frac{1}{q_k})^{q_e+q_r}(1 - \xi)^{q_s}$ In addition, $\Pr[E_2|E_1] = \epsilon$ and $\Pr[E_3|E_1 \wedge E_2] = \frac{\xi}{q_k}$ Combine together, we have $\epsilon' \geq (1 - \frac{1}{q_k})^{q_e+q_r}(1 - \xi)^{q_s}\frac{\xi}{q_k}\epsilon$ The function $(1 - \xi)^{q_s}\xi$ is maximized when $\xi = \frac{1}{q_s+1}$. Finally, we have $\epsilon' \geq (1 - \frac{1}{q_k})^{q_e+q_r}(1 - \frac{1}{q_s+1})^{q_s}\frac{1}{(q_s+1)q_k}\epsilon$ The time $\mathcal{B}$ has used in the simulation is approximately the same as $\mathcal{A}$. □

Theorem 2 (Game 2). *The CBS scheme proposed in this section is (ϵ, t)-existential unforgeable against Game 2 adversary with advantage at most ϵ and runs in time at most t, assuming that the (ϵ', t') k-CAA assumption holds in $\mathbb{G}$, where $\epsilon' \geq (1 - \frac{1}{q_k})^{q_r}(1 - \frac{1}{q_s+1})^{q_s}\frac{1}{(q_s+1)q_k}\epsilon$, $t' = t$ where q_k, q_r, q_s are the numbers of queries made to the User-key-gen Query, Corruption Query and Signing Query respectively.*

Proof. Assume there exists a Game 2 adversary $\mathcal{A}$. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the k-CAA problem with probability at least ϵ' and in time at most t'. We use a similar approach as in [14].

Let g be a generator of $\mathbb{G}$. Let $x, h_1, \ldots, h_k \in \mathbb{Z}_p^*$ be $k+1$ integers. $\mathcal{B}$ is given the problem instance $g, g^x, \left(h_1, g^{(x+h_1)^{-1}}\right), \ldots, \left(h_k, g^{(x+h_k)^{-1}}\right)$. $\mathcal{B}$ is asked to output $\left(h^*, g^{(x+h^*)^{-1}}\right)$ for some $h^* \notin \{h_1, \ldots, h_k\}$.

Setup. $\mathcal{B}$ simulates the environment and the oracles which $\mathcal{A}$ can access. We regard the hash function H_1 and H_2 as random oracles. $\mathcal{B}$ first randomly chooses $s \in_R \mathbb{Z}_p^*$ and sets $g_1 = g^s$ and gives the parameter $\mathsf{param} = (e, \mathbb{G}, \mathbb{G}_T, p, g, g_1)$, together with the master secret key s to $\mathcal{A}$.

Oracles Simulation. $\mathcal{B}$ simulates all oracles as follow. Note that different from Game 1, there is no Certification Query, as $\mathcal{A}$ has already obtained the master secret key.

(User-key-gen (and H_1 random oracle) Query .) At the beginning $\mathcal{B}$ first randomly chooses a number $t \in \{1, \ldots, q_k\}$. $\mathcal{B}$ also maintains a list L_U which stores the secret/public key pair of the corresponding identity. $\mathcal{A}$ can query this oracle by submitting an identity ID_i. If the public key pair is already created, $\mathcal{B}$ retrieves them from the list L_U. Otherwise,

1. If $i \neq t$, $\mathcal{B}$ chooses $d_i, x_i \in_R \mathbb{Z}_p^*$ and sets $PK_i = g^{x_i}$, $H_1(ID_i, PK_i) = g^{d_i}$. The corresponding certificate is $C_{ID_i} = H_1(ID_i, PK_i)^s = (g^s)^{d_i}$ and the user secret key is x_i. $\mathcal{B}$ stores (d_i, x_i, PK_i, ID_i) to the list L_U.
2. If $i = t$, $\mathcal{B}$ randomly chooses $d_t \in_R \mathbb{Z}_p^*$, sets $PK_i = g^x$ and $H_1(ID_i, PK_i) = g^{d_t}$. In this case, the user secret key is unknown to $\mathcal{B}$.

(Corruption Query.) Given an identity ID_i, $\mathcal{B}$ outputs $\perp$ if $ID_i = ID_t$ or ID_i has not been created. Otherwise, $\mathcal{B}$ returns x_i from the list L_U.

(Signing (and H_2 random oracle) Query.) We first describe the simulation of H_2 random oracle. On the i-th input $\omega_i = (m_\ell, ID_j, PK_k)$, $\mathcal{B}$ first checks if $ID_j = ID_t$ and $PK_k = PK_t$. We divide into two cases:

1. If $ID_j = ID_t$ and $PK_k = PK_t$, then $\mathcal{B}$ flips a biased coin which outputs a value $c_i = 1$ with probability ξ, and $c_i = 0$ with probability $1 - \xi$. The value of ξ will be optimized later.
   - If $c_i = 1$, $\mathcal{B}$ randomly chooses $h'_i \in_R \mathbb{Z}_p^*$ where $h'_i \notin \{h_1, \ldots, h_k\}$ and returns h'_i to $\mathcal{A}$ as the value of $H_2(\omega_i)$.
   - If $c_i = 0$, $\mathcal{B}$ randomly chooses $h''_i \in_R \{h_1, \ldots, h_k\}$ as the output of $H_2(\omega_i)$, where h''_i must be a fresh value which has not been assigned as an output of H_2 queries before.
2. Otherwise, $\mathcal{B}$ randomly chooses $\mu_i \in_R \mathbb{Z}_p^*$ as the output.

In all cases, $\mathcal{B}$ records $(\omega_i, h'_i, c_i), (\omega_i, h''_i, c_i)$ or (ω_i, μ_i) to another list L_{H_2}.

For each signing query on an input (m_ℓ, ID_j, PK_j), output $\perp$ if ID_j has not been created. Otherwise, we assume that $\mathcal{A}$ has already queried the random oracle H_2 on the input $\omega_i = (m_\ell, ID_j, PK_j)$. We divide into two cases:

1. If $ID_j \neq ID_t$, $\mathcal{B}$ uses the the user secret key and certificate of ID_j from the list L_k and $\mu_i = H_2(\omega_i)$ from the list L_{H_2} to generate a valid signature.

2. If $ID_j = ID_t$, $\mathcal{B}$ first checks the list L_{H_2}.
   - If $c_i = 1$, $\mathcal{B}$ reports *failure* and terminates the simulation.
   - Otherwise, that is $c_i = 0$ and $h''_i = H_2(m_\ell, ID_t, PK_t)$ is on the list L_{H_2}. For simplicity, we assume $h''_i = h_i \in \{h_1, \ldots, h_k\}$. $\mathcal{B}$ outputs $\sigma_i = (g^{sd_t})^{(x+h_i)^{-1}}$ as the signature. It is a valid signature because $e(\sigma_i, PK_t \cdot g^{h_i}) = e((g^{sd_t})^{(x+h_i)^{-1}}, g^x \cdot g^{h_i}) = e(g^{sd_t}, g) = e(g^{d_t}, g^s) = e(H_1(ID_t, PK_t), g_1)$

Forgery. After all queries, $\mathcal{A}$ outputs $(ID^*, PK^*, m^*, \sigma^*)$. If $\mathcal{A}$ wins, we have the following conditions satisfied:

1. PK^* has not been submitted to corruption query.
2. (ID^*, PK^*, m^*) has not been submitted to signing query.
3. σ^* is a valid signature on (ID^*, PK^*, m^*). That is,

$$\begin{aligned} e(\sigma^*, PK^* \cdot g^{h^*}) &= e(H_1(ID^*, PK^*), g_1) \\ e(\sigma^*, g^x \cdot g^{h^*}) &= e(g^{d_t}, g^s) \\ e(\sigma^*, g) &= e((g^{sd_t})^{(x+h^*)^{-1}}, g) \\ \Rightarrow \quad \sigma^* &= (g^{sd_t})^{(x+h^*)^{-1}} \end{aligned}$$

   where $h^* = H_2(m^*, ID^*, PK^*)$ and $PK^* = g^{x^*}$.

If $ID^* \neq ID_t$, then $\mathcal{B}$ outputs *failure* and terminates the simulation. Otherwise, $\mathcal{B}$ checks the list L_{H_2}. If $c^* = 0$, $\mathcal{B}$ outputs *failure* and terminates the simulation. Otherwise, that is, $c^* = 1$ and $h^* \notin \{h_1, \ldots, h_k\}$. $\mathcal{B}$ computes $\phi = (\sigma^*)^{(sd_t)^{-1}} = g^{(x+h^*)^{-1}}$ and outputs (h^*, ϕ) as the solution of the problem instance.

Probability Analysis. $\mathcal{B}$ succeeds if: (1) E_1: $\mathcal{B}$ does not abort during simulation; (2) E_2: $\mathcal{A}$ wins; and (3) E_3: $ID^* = ID_t$ and $c^* = 1$.

The advantage of $\mathcal{B}$ is $\epsilon' = \Pr[E_1 \wedge E_2 \wedge E_3] = \Pr[E_1] \cdot \Pr[E_2|E_1] \cdot \Pr[E_3|E_1 \wedge E_2]$ If E_1 happens, then:

- $\mathcal{B}$ does not output *failure* during the simulation of the corruption query. This happens with probability $(1 - \frac{1}{q_k})^{q_r}$.
- $\mathcal{B}$ does not output *failure* during the simulation of the signing query. This happens with probability $(1 - \frac{1}{q_k}\xi)^{q_s} \geq (1 - \xi)^{q_s}$.

Now we have $\Pr[E_1] \geq (1 - \frac{1}{q_k})^{q_r}(1 - \xi)^{q_s}$ In addition, $\Pr[E_2|E_1] = \epsilon$ and $\Pr[E_3|E_1 \wedge E_2] = \frac{\xi}{q_k}$ Combine together, we have $\epsilon' \geq (1 - \frac{1}{q_k})^{q_r}(1 - \xi)^{q_s}\frac{\xi}{q_k}\epsilon$ The function $(1 - \xi)^{q_s}\xi$ is maximized when $\xi = \frac{1}{q_s+1}$. Finally, we have $\epsilon' \geq (1 - \frac{1}{q_k})^{q_r}(1 - \frac{1}{q_s+1})^{q_s}\frac{1}{(q_s+1)q_k}\epsilon$ The time $\mathcal{B}$ has used in the simulation is approximately the same as $\mathcal{A}$. □

4 Conclusion

We have proposed a short CBS scheme. It is the shortest CBS scheme in the literature. The signature size and public key are just one group element respectively. It is even shorter than any PKI based signature, which require at least one group element for the signature size, public key and certificate respectively. Our scheme can be proven secure in the random oracle model using the weak modified k-CAA assumption. The computation requirement is very light, can be implemented in many devices easily. We believe our scheme can be very useful in many different applications such as wireless sensor networks and mesh networks, wherever space efficiency and level of trust are the major concerns.

References

1. S. S. Al-Riyami and K. Paterson. Certificateless public key cryptography. In *ASIACRYPT '03*, volume 2894 of *LNCS*, pages 452–473. Springer-Verlag, 2003.
2. M. Au, J. Chen, J. Liu, Y. Mu, D. Wong, and G. Yang. Malicious KGC attacks in certificateless cryptography. In *ASIACCS 2007*, pages 302–311. ACM Press, 2007. Also available at http://eprint.iacr.org/2006/255.
3. M. Au, J. Liu, W. Susilo, and T. Yuen. Certificate based (linkable) ring signature. In *ISPEC '07*, volume 4464 of *LNCS*, pages 79–92. Springer-Verlag, 2007.
4. D. Boneh and M. K. Franklin. Identity-Based Encryption from the Weil Pairing. In *CRYPTO '01*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001.
5. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001.
6. C. Gentry. Certificate-based encryption and the certificate revocation problem. In *EUROCRYPT '03*, pages 272–293. Springer-Verlag, 2003. LNCS No. 2656.
7. M. Girault. Self-certified public keys. In *EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 490–497. Springer, 1991.
8. F. Hess. Efficient identity based signature schemes based on pairings. In *Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer, 2002.
9. B. G. Kang, J. H. Park, and S. G. Hahn. A certificate-based signature scheme. In *CR-RSA '04*, volume 2964 of *LNCS*, pages 99–111. Springer, 2004.
10. J. Li, X. Huang, Y. Mu, W. Susilo, and Q. Wu. Certificate-based signature: Security model and efficient construction. In *EuroPKI '07*, volume 4582 of *LNCS*, pages 110–125. Springer, 2007.
11. J. K. Liu, J. Baek, W. Susilo, and J. Zhou. Certificate-based signature schemes without pairings or random oracles. In *ISC*, volume 5222 of *Lecture Notes in Computer Science*, pages 285–297. Springer, 2008.
12. S. Mitsunari, R. Sakai, and M. Kasahara. A new traitor tracing. *IEICE Transactions*, E85-A(2):481–484, 2002.
13. A. Shamir and Y. Tauman. Improved online/offline signature schemes. In *Proc. CRYPTO '01*, volume 2139 of *Lecture Notes in Computer Science*, pages 355–367. Springer-Verlag, 2001.
14. R. Tso, X. Yi, and X. Huang. Efficient and short certificateless signature. In *CANS*, volume 5339 of *Lecture Notes in Computer Science*, pages 64–79. Springer, 2008.