Certificate-Based Signature Schemes without Pairings or Random Oracles

Joseph K. Liu[1], Joonsang Baek[1], Willy Susilo[2], and Jianying Zhou[1]

[1] Cryptography and Security Department
Institute for Infocomm Research, Singapore
{ksliu, jsbaek, jyzhou}@i2r.a-star.edu.sg
[2] Centre for Computer and Information Security (CCISR)
School of Computer Science and Software Engineering
University of Wollongong, Australia
wsusilo@uow.edu.au

Abstract. In this paper, we propose two new certificate-based signature (CBS) schemes with new features and advantages. The first one is very efficient as it does not require any pairing computation and its security can be proven using Discrete Logarithm assumption in the random oracle model. We also propose another scheme whose security can be proven in the standard model without random oracles. To the best of our knowledge, these are the *first* CBS schemes in the literature that have such kind of features.

1 Introduction

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) that provides an unforgeable signature and trusted link between the public key and the identity of the signer. The hierarchical framework is called the public key infrastructure (PKI), which is responsible to issue and manage the certificates (chain). In this case, 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. In the point of view of a verifier, it takes two verification steps for independent signatures. This approach seems inefficient, in particular when the number of users is very large.

Identity-Based cryptography (IBC). Identity-based cryptography (IBC), invented by Shamir [1] in 1984, 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 Key Generator (PKG)". This way, the certificate is

implicitly provided and the explicit authentication of public keys is no longer required. The main disadvantage of identity-based cryptography is an unconditional trust to the PKG. Hence, the PKG can impersonate any user, or decrypt any ciphertexts. 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 [2] introduced the concept of certificate-based encryption (CBE). A CBE scheme combines a public key encryption scheme and an identity based encryption 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 identity based encryption (IBE) [3] scheme to generate the certificate. 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.* [4]. However, one of their proposed schemes was found insecure against key replacement attack, as pointed out by Li *et al.* [5]. They also proposed a concrete scheme. In parallel to their construction, Au *et al.* [6] proposed a certificate-based ring signature scheme. Nevertheless, all the above schemes require pairing operations and can be only proven in the random oracle model. To date, it is unknown whether the existence of pairing based cryptography is essential for the construction of CBS schemes.

We remark that certificateless cryptography [7] is another stream of research, which is to solve the key escrow problem inherited by IBC. Although at the first glance certificateless cryptography shares several similarities with certificate-based cryptography, each notion has its own merit and distinct features.

Our Contributions. In this paper, we propose two CBS schemes. The first scheme does not require any pairing operations, which is regarded as costly operations compared to other operations such as exponentiation. According to the current MIRACL implementation [8], a 512-bit Tate pairing takes 20 ms whereas a 1024-bit prime modular exponentiation takes 8.8 ms. Without pairing, our scheme is more efficient than all of the previous schemes proposed so far [4–6]. This distinct and interesting property enables our scheme to be implemented in some power-constrained devices, such as wireless sensor networks.

In addition, our second scheme can be proven in the standard model without random oracles. All previous schemes mentioned above rely on the random oracle model to prove their security. It is generally believed that cryptographic schemes relaying on the random oracles may not be secure if the underlying random oracles are realized as hash functions in the real world. For some applications that require very high security, it is believed that only those schemes that can be proven in the standard model without random oracles must be employed. Hence our second scheme can be suitable for those particular applications.

2 Preliminaries

2.1 Notations

Pairing . Let e be a bilinear map such that $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ such that it has the following properties:

- $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic multiplicative groups of prime order p.
- each element of $\mathbb{G}$ and $\mathbb{G}_T$ has unique binary representation.
- g is a generator of $\mathbb{G}$.
- (Bilinear) $\forall x, y \in \mathbb{G}$ and a, $b \in \mathbb{Z}_p$, $e(x^a, y^b) = e(x, y)^{ab}$.
- (Non-degenerate) $e(g, g) \neq 1$.

2.2 Mathematical Assumptions

Definition 1 (Discrete Logarithm (DL) Assumption). *Given a group G of prime order q with generator g and elements $A \in G$, the DL problem in G is to output $\alpha \in \mathbb{Z}_q$ such that $A = g^\alpha$.*

An adversary $\mathcal{B}$ has at least an ϵ advantage if

$$\Pr[\mathcal{B}(g, A) = \alpha \mid A = g^\alpha] \geq \epsilon$$

We say that the (ϵ, t)-DL assumption holds in a group G if no algorithm running in time at most t can solve that DL problem in G with advantage at least ϵ.

Definition 2 (Generalized Computational Diffie-Hellman (GCDH) Assumption). *Given a group G of prime order p with generator g and elements $g^a, g^b \in G$ where a, b are selected uniformly at random from $\mathbb{Z}_p^*$, the GCDH problem in G is to output (g^{abc}, g^c), where $g^c \in G$.*

An adversary $\mathcal{B}$ has at least an ϵ advantage if

$$\Pr[\mathcal{B}(g, g^a, g^b) = (g^{abc}, g^c)] \geq \epsilon$$

We say that the (ϵ, t)-GCDH assumption holds in a group G if no algorithm running in time at most t can solve the GCDH problem in G with advantage at least ϵ.

It is a strictly weaker assumption than the Generalized Bilinear Diffie-Hellman (GBDH) assumption [7], which is defined as follow:

Definition 3 (Generalized Bilinear Diffie-Hellman (GBDH) Assumption). *Given a group G of prime order p with generator g and elements $g^a, g^b, g^{b'} \in G$ where a, b, b' are selected uniformly at random from $\mathbb{Z}_p^*$, the GBDH problem in G is to output $(e(g^c, g)^{abb'}, g^c)$, where $g^c \in G$.*

An adversary $\mathcal{B}$ has at least an ϵ advantage if

$$\Pr[\mathcal{B}(g, g^a, g^b, g^{b'}) = (e(g^c, g)^{abb'}, g^c)] \geq \epsilon$$

We say that the (ϵ, t)-GBDH assumption holds in a group G if no algorithm running in time at most t can solve the GBDH problem in G with advantage at least ϵ.

Lemma 1. *The GCDH assumption is strictly weaker than the GBDH assumption.*

Proof. Assume there is a GCDH adversary $\mathcal{A}$. We construct another adversary $\mathcal{B}$ to solve the GBDH problem. Given $(g, g^a, g^b, g^{b'})$ as the GBDH problem instance, $\mathcal{B}$ gives (g, g^a, g^b) to $\mathcal{A}$. $\mathcal{A}$ outputs (g^{abc}, g^c) with probability ϵ and time t. $\mathcal{B}$ outputs $(e(g^{b'}, g^{abc}), g^c)$ as the solution to the GBDH problem. Since $e(g^{b'}, g^{abc}) = e(g^c, g)^{abb'}$, it is a valid solution. Thus $\mathcal{B}$ can solve GBDH problem with time t and probability ϵ.

Definition 4 (Many-DH Assumption [9] (Simplified Version) [3]). *Given a group G of prime order p with generator g and elements $g^a, g^b, g^c, g^{ab}, g^{ac}, g^{bc} \in G$ where a, b, c are selected uniformly at random from $\mathbb{Z}_p^*$, the Many-DH problem in G is to output g^{abc}.*

An adversary $\mathcal{B}$ has at least an ϵ advantage if

$$\Pr[\mathcal{B}(g, g^a, g^b, g^c, g^{ab}, g^{ac}, g^{bc}) = g^{abc}] \geq \epsilon$$

We say that the (ϵ, t)-Many-DH assumption holds in a group G if no algorithm running in time at most t can solve the Many-DH problem in G with advantage at least ϵ.

3 Security Model

Definition 5. *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.*
- *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.*

[3] In the original version presented in [9], the number of input tuples can be as much as $\mathcal{O}(\log k)$ for some security parameter k. Here we simplify it for just enough to our scheme.

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.* [5] which captures key replacement attack in the security of Game 1.

Definition 6 (CBS Game 1 Existential Unforgeability). *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 signing query* (ID, PK, m)*, the challenger generates* σ *by using algorithm* Sign*.*

$\mathcal{A}$ *can replace any user* ID *public key with his own choice, but once it has replaced the public key, it cannot obtain the certificate of the false public key from the challenger. 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*

- σ^* *is a valid signature on the message* m^* *under the public key* PK^* *with user information* ID^**, where* PK^* *might not be the one returned from user-key-gen query.*
- ID^* *has never been submitted to the certification query.*
- (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 7 (CBS Game 2 Existential Unforgeability). *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

- *σ^* is a valid signature on the message m^* under the public key PK^* with user information ID^*.*
- *PK^* is an output from user-key-gen query.*
- *ID^* has never been submitted to corruption query.*
- *(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)* [10] has been formalized in the area of certificateless cryptography.

4 A CBS without Pairing

Our scheme is motivated from Beth's identification scheme [11].

Setup. Let $\mathbb{G}$ be a multiplicative group with order q. The PKG selects a random generator $g \in \mathbb{G}$ and randomly chooses $x \in_R \mathbb{Z}_q^*$. It sets $X = g^x$. Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ be a cryptographic hash function. The public parameters param and master secret key msk are given by

$$\mathsf{param} = (\mathbb{G}, q, g, X, H) \qquad msk = x$$

UserKeyGen. User selects a secret value $u \in \mathbb{Z}_q^*$ as his secret key usk, and computes his public key PK as (g^u, X^u, π_u) where π_u is the following non-interactive proof-of-knowledge (PoK) [4]:

$$PK\{(u) : U_1 = g^u \wedge U_2 = X^u\}$$

Certify. Let $\tilde{h} = H(PK, ID)$ for user with public key PK and binary string ID which is used to identify the user. To generate a certificate for this user, the CA randomly selects $r \in_R \mathbb{Z}_q^*$, computes

$$R = g^r \qquad s = r^{-1}(\tilde{h} - xR) \bmod q$$

[4] For the details of notation and implementation of PoK, please refer to [12].

The certificate is (R, s). Note that a correctly generated certificate should fulfill the following equality:

$$R^s X^R = g^{\tilde{h}} \tag{1}$$

Sign. To sign a message $m \in \{0,1\}^*$, the signer with public key PK (and user info ID) , certificate (R, s) and secret key u, randomly selects $y \in_R \mathbb{Z}_q^*$, computes

$$Y = R^{-y} \qquad h = H(Y, R, m) \qquad z = y + h\ s\ u \bmod q$$

and outputs (Y, R, z) as the signature σ.

Verify. Given a signature $\sigma = (Y, R, z)$ for a public key PK on a message m, a verifier first checks whether π_u is a valid PoK. If not, output $\perp$. Otherwise computes $h = H(Y, R, m)$, $\tilde{h} = H(PK, ID)$, and checks whether

$$(g^u)^{h\tilde{h}} \stackrel{?}{=} R^z Y (X^u)^{hR} \tag{2}$$

Output valid if it is equal. Otherwise, output $\perp$.

4.1 Security Analysis

Correctness. It is easy to see that the signature scheme is correct, as shown in following:

$$R^z Y (X^u)^{hR} = R^{y+hsu} R^{-y} X^{uhR} = g^{rhsu} g^{uxhR}$$
$$= g^{rhu\left(r^{-1}(\tilde{h}-xR)\right)+uxhR} = g^{hu\tilde{h}-huxR+uxhR} = (g^u)^{h\tilde{h}}$$

Theorem 1 (Unforgeability against Game 1 Adversary). *The CBS scheme without pairing is (ϵ, t)-existential unforgeable against Game 1 adversary (defined in Section 3) with advantage at most ϵ and runs in time at most t, assuming that the (ϵ', t')-DL assumption holds in $\mathbb{G}$, where*

$$\epsilon' = \Big(1 - \frac{q_h(q_e + q_s)}{q}\Big)\Big(1 - \frac{1}{q}\Big)\Big(\frac{1}{q_h}\Big)\epsilon, \qquad t' = t + \mathcal{O}(q_e + q_s)E$$

and q_e, q_s, q_h are the numbers of certification queries, signing queries and hashing queries the adversary is allowed to make and E is the time for an exponentiation operation.

Proof. Here we follow the idea from [13, 14]. Assume there exists a forger $\mathcal{A}$. We construct an algorithm $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve discrete logarithm problem. $\mathcal{B}$ is given a multiplicative group $\mathbb{G}$ with generator g and order q, and a group element $A \in \mathbb{G}$. $\mathcal{B}$ is asked to find $\alpha \in \mathbb{Z}_q$ such that $g^\alpha = A$.

Setup: $\mathcal{B}$ chooses a hash function $H : \{0,1\}^* \to \mathbb{Z}_q$ which behaves like a random oracle. $\mathcal{B}$ is responsible for the simulation of this random oracle. $\mathcal{B}$ assigns $X = A$ and outputs the public parameter $param = (\mathbb{G}, q, g, X, H)$ to $\mathcal{A}$.

User-Key-Gen / Corruption Query: $\mathcal{B}$ generates the secret and public key pair according to the algorithm and stores in the table and outputs the public key. On the corruption query, $\mathcal{B}$ returns the corresponding secret key.

Certification Query: $\mathcal{A}$ is allowed to make certification query for a public key PK with identification string ID. $\mathcal{B}$ simulates the oracle as follow. It randomly chooses $a, b \in_R \mathbb{Z}_q$ and sets

$$R = X^a g^b \qquad s = -a^{-1}R \bmod q \qquad H(PK, ID) = bs \bmod q$$

Note that (R, s) generated in this way satisfies the equation (1) in the Certify algorithm. It is a valid certificate. $\mathcal{B}$ outputs (R, s) as the certificate of PK, ID and store the value of $(R, s, H(PK, ID), PK, ID)$ in the table for consistency. Later if $\mathcal{A}$ queries the H random oracle for PK, ID, $\mathcal{B}$ outputs the same value. If PK, ID is not found in the table, $\mathcal{B}$ executes the certification oracle simulation, stores the value $(R, s, H(PK, ID), PK, ID)$ in the table and outputs $H(PK, ID)$ only.

Signing Query: $\mathcal{A}$ queries the signing oracle for a message m and a public key $PK = (g^u, X^u, \pi_u)$ with identification string ID. $\mathcal{B}$ first checks that whether π_u is a valid PoK for (g^u, X^u). If not , output $\perp$. Else further checks whether PK, ID has been queries for the H random oracle or extraction oracle before. If yes, it just retrieves $(R, s, H(PK, ID), PK, ID)$ from the table. Let $\tilde{h} = H(PK, ID)$. It also randomly generates $h, z \in_R \mathbb{Z}_q$, and sets $Y = \frac{(g^u)^{h\tilde{h}}}{R^z (X^u)^{hR}}$ and assigns the value h to the random oracle query of $H(Y, R, m)$. It outputs the signature (Y, R, z) for the message m and stores the value h, corresponding to $H(Y, R, m)$ in the hash table for consistency. If PK, ID has not been queried to the random oracle or certification oracle, $\mathcal{B}$ executes the simulation of the certification oracle and uses the corresponding certificate to sign the message by the above algorithm.

Output Calculation: Finally the adversary $\mathcal{A}$ outputs a forged signature $\sigma^*_{(1)} = (Y^*, R^*, z^*_{(1)})$ on message m^* and public key PK^* with identification string ID^*. $\mathcal{B}$ rewinds $\mathcal{A}$ to the point it just queries $H(Y^*, R^*, m^*)$ and supplies with a different value (corresponding to the same input value to the hash query). $\mathcal{A}$ outputs another pair of signature $\sigma^*_{(2)} = (Y^*, R^*, z^*_{(2)})$. $\mathcal{B}$ repeats twice and obtains $\sigma^*_{(3)} = (Y^*, R^*, z^*_{(3)})$ and $\sigma^*_{(4)} = (Y^*, R^*, z^*_{(4)})$. Note that Y^* and R^* should be the same every time. We let c_1, c_2, c_3, c_4 be the output of the random oracle queries $H(Y^*, R^*, m^*)$ for the first, second, third and forth time.

We also denote $u, r, x, y \in \mathbb{Z}_q$ such that $g^r = R^*$, $g^x = X$, $g^y = Y^*$ and $PK^* = (g^u, X^u)$. From equation (2), we have

$$c_i u H(PK^*, ID^*) = r z^*_{(i)} + y + x u c_i R^* \bmod q \qquad \text{for } i = 1, 2, 3, 4$$

In these equations, only r, y, x, u are unknown to $\mathcal{B}$. $\mathcal{B}$ solves for these values from the above 4 linear independent equations, and outputs x as the solution of the discrete logarithm problem.

Probability Analysis: The simulation of the random oracle fails if the oracle assignment $H(PK, ID)$ causes inconsistency. It happens with probability at most

q_h/q. Hence the simulation is successful $q_e + q_s$ times (since $H(PK, ID)$ may also be queried in the signing oracle if PK, ID has not been queried in the certification oracle) with probability at least

$$\Big(1 - \frac{q_h}{q}\Big)^{q_e+q_s} \geq 1 - \frac{q_h(q_e + q_s)}{q}$$

Due to the ideal randomness of the random oracle, there exists a query $H(Y^*, R^*, m^*)$ with probability at least $1 - 1/q$. $\mathcal{B}$ guesses it correctly as the point of rewind, with probability at least $1/q_h$. Thus the overall successful probability is

$$\Big(1 - \frac{q_h(q_e + q_s)}{q}\Big)\Big(1 - \frac{1}{q}\Big)\Big(\frac{1}{q_h}\Big)\epsilon$$

The time complexity of the algorithm $\mathcal{B}$ is dominated by the exponentiations performed in the certification and signing queries, which is equal to $t + \mathcal{O}(q_e + q_s)E$ □

Theorem 2 (Unforgeability against Game 2 Adversary). *The CBS scheme without pairing is (ϵ, t)-existential unforgeable against Game 2 adversary (defined in Section 3) with advantage at most ϵ and runs in time at most t, assuming that the (ϵ', t')-DL assumption holds in $\mathbb{G}$, where*

$$\epsilon' = \Big(1 - \frac{q_h q_s}{q}\Big)\Big(1 - \frac{1}{q}\Big)\Big(\frac{1}{q_h}\Big)\Big(\frac{1}{q_u}\Big)\epsilon, \qquad t' = t + \mathcal{O}(q_s)E$$

and q_s, q_h, q_u are the numbers of signing queries, hashing queries and user-key-gen queries the adversary is allowed to make and E is the time for an exponentiation operation.

Proof. Assume there exists a forger $\mathcal{A}$. We construct an algorithm $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve discrete logarithm problem. $\mathcal{B}$ is given a multiplicative group $\mathbb{G}$ with generator g and order q, and a group element $A \in \mathbb{G}$. $\mathcal{B}$ is asked to find $\alpha \in \mathbb{Z}_q$ such that $g^\alpha = A$.

Setup: $\mathcal{B}$ chooses a hash function $H : \{0,1\}^* \to \mathbb{Z}_q$ which behaves like a random oracle. $\mathcal{B}$ is responsible for the simulation of this random oracle. $\mathcal{B}$ randomly chooses $x \in_R \mathbb{Z}_q$ and sets $X = g^x$. It outputs the public parameter $param = (\mathbb{G}, q, g, X, H)$ to $\mathcal{A}$.

User-Key-Gen Query: $\mathcal{B}$ chooses a particular query ID' and assign the public key $PK' = (A, A^x, \pi')$ where π' can be simulated by the control of the random oracle. For the other queries, $\mathcal{B}$ generates the secret and public key pair according to the algorithm and stores the value in the table.

Corruption Query: If the query is not ID', $\mathcal{B}$ outputs the corresponding secret key from the table. Otherwise $\mathcal{B}$ aborts.

Signing Query: It can be simulated in the same way as in Game 1, which also does not require the knowledge of the secret key.

Output Calculation: Finally the adversary $\mathcal{A}$ outputs a forged signature $\sigma^*_{(1)} = (Y^*, R^*, z^*_{(1)})$ on message m^* and public key PK^* with identification string ID^*. If $PK^* \neq PK'$, $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ rewinds $\mathcal{A}$ to the point it just queries $H(Y^*, R^*, m^*)$ and supplies with a different value (corresponding to the same input value to the hash query). $\mathcal{A}$ outputs another pair of signature $\sigma^*_{(2)} = (Y^*, R^*, z^*_{(2)})$. $\mathcal{B}$ repeats and obtains $\sigma^*_{(3)} = (Y^*, R^*, z^*_{(3)})$. Note that Y^* and R^* should be the same every time. We let c_1, c_2, c_3 be the output of the random oracle queries $H(Y^*, R^*, m^*)$ for the first, second and third.

We also denote $u, r, y \in \mathbb{Z}_q$ such that $g^r = R^*$, $g^y = Y^*$ and $PK^* = (g^u, X^u)$. From equation (2), we have

$$c_i u H(PK^*, ID^*) = r z^*_{(i)} + y + x u c_i R^* \bmod q \qquad \text{for } i = 1, 2, 3$$

In these equations, only r, y, u are unknown to $\mathcal{B}$. $\mathcal{B}$ solves for these values from the above 3 linear independent equations, and outputs u as the solution of the discrete logarithm problem.

Probability Analysis: The simulation of the random oracle fails if the oracle assignment $H(PK, ID)$ causes inconsistency. It happens with probability at most q_h/q. Hence the simulation is successful q_s times with probability at least $\left(1 - \frac{q_h}{q}\right)^{q_s} \geq 1 - \frac{q_h q_s}{q}$ Due to the ideal randomness of the random oracle, there exists a query $H(Y^*, R^*, m^*)$ with probability at least $1 - 1/q$. $\mathcal{B}$ guesses it correctly as the point of rewind, with probability at least $1/q_h$. In addition, $\mathcal{B}$ needs to guess correctly that $PK' = PK^*$, which happens with probability $1/q_u$. Thus the overall successful probability is $\left(1 - \frac{q_h q_s}{q}\right)\left(1 - \frac{1}{q}\right)\left(\frac{1}{q_h}\right)\left(\frac{1}{q_u}\right)\epsilon$ The time complexity of the algorithm $\mathcal{B}$ is dominated by the exponentiations performed in the signing queries, which is equal to $t + \mathcal{O}(q_s)E$ □

5 A CBS without Random Oracles

Our scheme is motivated from the identity-based encryption scheme from Waters [15]. Let $H_u : \{0,1\}^* \to \{0,1\}^{n_u}$ and $H_m : \{0,1\}^* \to \{0,1\}^{n_m}$ be two collision-resistant cryptographic hash functions for some $n_u, n_m \in \mathbb{Z}$.

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 $\mathbb{G}$. Randomly select $\alpha \in_R \mathbb{Z}_p$, $g_2 \in_R \mathbb{G}$ and compute $g_1 = g^\alpha$. Also select randomly the following elements:

$$u', m' \in_R \mathbb{G} \qquad \hat{u}_i \in_R \mathbb{G} \text{ for } i = 1, \ldots, n_u \qquad \hat{m}_i \in_R \mathbb{G} \text{ for } i = 1, \ldots, n_m$$

Let $\hat{U} = \{\hat{u}_i\}, \hat{M} = \{\hat{m}_i\}$. The public parameters param are $(e, \mathbb{G}, \mathbb{G}_T, p, g, g_1, g_2, u', \hat{U}, m', \hat{M})$ and the master secret key msk is g_2^α.

UserKeyGen. User selects a secret value $x \in \mathbb{Z}_p$ as his secret key usk, and computes his public key PK as $(\mathsf{pk}^{(1)}, \mathsf{pk}^{(2)}) = (g^x, g_1^x)$.

Certify. Let $\mathfrak{u} = H_u(PK, ID)$ for user with public key PK and binary string ID which is used to identify the user. Let $\mathfrak{u}[i]$ be the i-th bit of $\mathfrak{u}$. Define $\mathcal{U} \subset \{1, \ldots, n_u\}$ to be the set of indicies such that $\mathfrak{u}[i] = 1$.

To construct the certificate, the CA randomly selects $r_u \in_R \mathbb{Z}_p$ and computes

$$\left(g_2^{\alpha}\big(U\big)^{r_u}, g^{r_u}\right) = (\mathsf{cert}^{(1)}, \mathsf{cert}^{(2)}) \qquad \text{where } U = u' \prod_{i \in \mathcal{U}} \hat{u}_i$$

Sign. To sign a message $m \in \{0,1\}^*$, the signer with identity PK (and user information ID) , certificate $(\mathsf{cert}^{(1)}, \mathsf{cert}^{(2)})$ and secret key usk, compute $\mathfrak{m} = H_m(m)$. Let $\mathfrak{m}[i]$ be the i-th bit of $\mathfrak{m}$ and $\mathcal{M} \subset \{1, \ldots, n_m\}$ be the set of indicies i such that $\mathfrak{m}[i] = 1$. Randomly select $r_\pi, r_\mathfrak{m} \in_R \mathbb{Z}_p$, compute $\mathfrak{u} = H_u(PK, \mathsf{userinfo})$, $U = u' \prod_{i \in \mathcal{U}} \hat{u}_i$ and

$$\sigma = \left(\Big(\mathsf{cert}^{(1)}\Big)^{usk}\Big(U\Big)^{r_\pi}\Big(m' \prod_{i \in \mathcal{M}} \hat{m}_i\Big)^{r_\mathfrak{m}} \ , \ \Big(\mathsf{cert}^{(2)}\Big)^{usk} g^{r_\pi} \ , \ g^{r_\mathfrak{m}}\right) \ = (V, R_\pi, R_\mathfrak{m})$$

Verify. Given a signature $\sigma = (V, R_\pi, R_\mathfrak{m})$ for a public key PK and user information ID on a message m, a verifier first checks whether $e(g^x, g_1) = e(g_1^x, g)$. If not, outputs $\perp$. Otherwise computes $\mathfrak{m} = H_m(m)$, $\mathfrak{u} = H_u(PK, ID)$, $U = u' \prod_{i \in \mathcal{U}} \hat{u}_i$ and checks whether

$$e(V, g) \stackrel{?}{=} e(g_2, g_1^x)\ e(U, R_\pi)\ e(m' \prod_{i \in \mathcal{M}} \hat{m}_i, R_\mathfrak{m})$$

Output valid if it is equal. Otherwise output $\perp$.

5.1 Security Analysis

Correctness. The correctness of the scheme is as follows.

$$\begin{aligned} e(V, g) &= e\Big(g_2^{\alpha x}\ U^{r_u x}\ U^{r_\pi}\ \Big(m' \prod_{i \in \mathcal{M}} \hat{m}_i\Big)^{r_m}\ ,\ g\Big) \\ &= e(g_2^x, g^\alpha)\ e(U^{r_u x + r_\pi}, g)\ e\Big(\big(m' \prod_{i \in \mathcal{M}} \hat{m}_i\big)^{r_m}, g\Big) \\ &= e(g_2, g_1^x)\ e(U, g^{r_u x + r_\pi})\ e(m' \prod_{i \in \mathcal{M}} \hat{m}_i, g^{r_\mathfrak{m}}) \\ &= e(g_2, g_1^x)\ e(U, R_\pi)\ e(m' \prod_{i \in \mathcal{M}} \hat{m}_i, R_\mathfrak{m}) \end{aligned}$$

Theorem 3 (Unforgeability against Game 1 Adversary). *The CBS scheme without random oracles is (ϵ, t)-existential unforgeable against Game 1 adversary (defined in Section 3) with advantage at most ϵ and runs in time at most t, assuming that the (ϵ', t')-GCDH assumption holds in $\mathbb{G}$, where*

$$\epsilon' \geq \frac{\epsilon}{16(q_e + q_s)(n_u + 1)q_s(n_m + 1)} \ , \ t' = t + O\Big(\big(q_e n_u + q_s(n_u + n_m)\big)\rho + (q_e + q_s)\tau\Big)$$

where q_e is the number of queries made to the Certification Query, q_s is the number of queries made to the Signing Query, and ρ and τ are the time for a multiplication and an exponentiation in $\mathbb{G}$ respectively.

The proof is given in Appendix A.

Theorem 4 (Unforgeability against Game 2 Adversary). *The CBS scheme without random oracles is (ϵ, t)-existential unforgeable against Game 2 adversary (defined in Section 3) with advantage at most ϵ and runs in time at most t, assuming that the (ϵ', t')-Many-DH assumption holds in $\mathbb{G}$, where*

$$\epsilon' \geq \frac{\epsilon}{16q_s(n_u+1)q_s(n_m+1)q_k} \ , \ t' = t + O\Big(\big(q_s(n_u+n_m)\big)\rho + (q_k+q_s)\tau\Big)$$

where q_s is the number of queries made to the Signing Queries, q_k is the number of queries made to the User-key-gen Queries and ρ and τ are the time for a multiplication and an exponentiation in $\mathbb{G}$ respectively.

The proof is given in Appendix B.

6 Concluding Remarks

In this paper, we proposed two certificate-based signature schemes. The first one does not require any pairing operations. Thus, it is very efficient and particularly suitable to be implemented in some power-constrained devices, such as wireless sensor networks. The second one does not require random oracles for proving its security. It may be suitable for applications that require a high level of security with regard to the fact that cryptographic schemes using the random oracles may not be secure if the random oracles are replaced by conventional hash functions in reality.

References

1. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: CRYPTO '84. Vol. 196 of LNCS., Springer (1984) 47–53
2. Gentry, C.: Certificate-based encryption and the certificate revocation problem. In: EUROCRYPT '03. Vol. 2656 of LNCS, Springer-Verlag (2003) 272–293
3. Boneh, D., Franklin, M.K.: Identity-Based Encryption from the Weil Pairing. In: CRYPTO '01. Vol. 2139 of LNCS., Springer (2001) 213–229
4. Kang, B.G., Park, J.H., Hahn, S.G.: A certificate-based signature scheme. In: Ct-RSA '04. Vol. 2964 of LNCS., Springer (2004) 99–111
5. Li, J., Huang, X., Mu, Y., Susilo, W., Wu, Q.: Certificate-based signature: Security model and efficient construction. In: EuroPKI '07. Vol. 4582 of LNCS., Springer (2007) 110–125
6. Au, M., Liu, J., Susilo, W., Yuen, T.: Certificate based (linkable) ring signature. In: ISPEC '07. Vol. 4464 of LNCS., Springer (2007) 79–92
7. Al-Riyami, S.S., Paterson, K.: Certificateless public key cryptography. In: ASIACRYPT '03. Vol. 2894 of LNCS., Springer (2003) 452–473
8. MIRACL: http://www.shamus.ie/.
9. Lysyanskaya, A.: Unique signatures and verifiable random functions from the DH-DDH separation. In: CRYPTO '04. Vol. 2442 of LNCS., Springer (2002) 597–612

10. Au, M., Chen, J., Liu, J., Mu, Y., Wong, D., Yang, G.: Malicious KGC attacks in certificateless cryptography. In: ASIACCS 2007, ACM Press (2007) 302–311
11. Beth, T.: Efficient Zero-Knowledged Identification Scheme for Smart Cards. In: EUROCRYPT '88. Vol. 330 of LNCS., Springer (1988) 77–86
12. Camenisch, J., Stadler, M.: Efficient Group Signature Schemes for Large Groups (Extended Abstract). CRYPTO '97. Vol. 1294 of LNCS., Springer (1997) 410–424
13. Bellare, M., Namprempre, C., Neven, G.: Security Proofs for Identity-Based Identification and Signature Schemes. In: EUROCRYPT '04. Vol. 3027 of LNCS., Springer (2004) 268–286
14. Bellare, M., Namprempre, C., Neven, G.: Security Proofs for Identity-Based Identification and Signature Schemes (Full version). Cryptology ePrint Archive, Report 2004/252 (2004) http://eprint.iacr.org/.
15. Waters, B.: Efficient identity-based encryption without random oracles. In: EUROCRYPT '05. Vol. 3494 of LNCS., Springer (2005) 114–127
16. Paterson, K., Schuldt, J.: Efficient identity-based signatures secure in the standard model. In: ACISP '06. Vol. 4058 of LNCS., Springer (2006) 195–206

A Proof of Theorem 3

Proof. Assume there exists a Game 1 adversary $\mathcal{A}$. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the GCDH problem with probability at least ϵ' and in time at most t'. We use a similar approach as in [16].

$\mathcal{B}$ is given a problem instance as follow: Given a group $\mathbb{G}$, a generator $g \in \mathbb{G}$, two elements $g^a, g^b \in \mathbb{G}$. It is asked to output two elements $g^{abc}, g^c \in \mathbb{G}$. In order to use $\mathcal{A}$ to solve for the problem, $\mathcal{B}$ needs to simulate a challenger and all oracles for $\mathcal{A}$. $\mathcal{B}$ does it in the following way.

Setup. Let $l_u = 2(q_e + q_s)$ and $l_m = 2q_s$. $\mathcal{B}$ randomly selects two integers k_u and k_m such that $0 \leq k_u \leq n_u$ and $0 \leq k_m \leq n_m$. Also assume that $l_u(n_u + 1) < p$ and $l_m(n_m + 1) < p$ for the given values of q_e, q_s, n_u and n_m. It randomly selects the following integers:

$$
\begin{aligned}
& x' \in_R \mathbb{Z}_{l_u} \ ; \ z' \in_R \mathbb{Z}_{l_m} \ ; \ y', w' \in_R \mathbb{Z}_p \\
& \hat{x}_i \in_R \mathbb{Z}_{l_u}, \text{ for } i = 1, \ldots, n_u \\
& \hat{z}_i \in_R \mathbb{Z}_{l_m}, \text{ for } i = 1, \ldots, n_m \\
& \hat{y}_i \in_R \mathbb{Z}_p, \text{ for } i = 1, \ldots, n_u \\
& \hat{w}_i \in_R \mathbb{Z}_p, \text{ for } i = 1, \ldots, n_m \\
& \text{Let } \hat{X} = \{\hat{x}_i\}, \hat{Z} = \{\hat{z}_i\}, \hat{Y} = \{\hat{y}_i\}, \hat{W} = \{\hat{w}_i\}
\end{aligned}
$$

We further define the following functions for binary strings $\mathfrak{u}$ and $\mathfrak{m}$ where $\mathfrak{u} = H_u(ID)$ for an identity ID and $\mathfrak{m} = H_m(m)$ for a message m, as follow:

$$
F(\mathfrak{u}) = x' + \sum_{i \in \mathcal{U}_j} \hat{x}_i - l_u k_u \qquad J(\mathfrak{u}) = y' + \sum_{i \in \mathcal{U}_j} \hat{y}_i
$$

$$
K(\mathfrak{m}) = z' + \sum_{i \in \mathcal{M}} \hat{z}_i - l_m k_m \qquad L(\mathfrak{m}) = w' + \sum_{i \in \mathcal{M}} \hat{w}_i
$$

$\mathcal{B}$ constructs a set of public parameters as follow:

$$g_1 = g^a, \qquad g_2 = g^b \qquad u' = g_2^{-l_u k_u + x'} g^{y'}, \qquad \hat{u}_i = g_2^{\hat{x}_i} g^{\hat{y}_i} \quad \text{for} \quad 1 \leq i \leq n_u$$
$$m' = g_2^{-l_m k_m + z'} g^{w'}, \qquad \hat{m}_i = g_2^{\hat{z}_i} g^{\hat{w}_i} \quad \text{for} \quad 1 \leq i \leq n_m$$

Note that the master secret will be $g_2^\alpha = g_2^a = g^{ab}$ and we have the following equations:

$$U = u' \prod_{i \in \mathcal{U}} \hat{u}_i = g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})} \qquad \text{and} \qquad m' \prod_{i \in \mathcal{M}} \hat{m}_i = g_2^{K(\mathfrak{m})} g^{L(\mathfrak{m})}$$

All public parameters are passed to $\mathcal{A}$.

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

(User-key-gen / Corruption Query.) $\mathcal{B}$ simulates thess oracle queries in a nature way, by running algorithm UserKeyGen and maintains a list L. It retrieves the secret key from L for the corruption query.

(Certification Query.) Upon receiving a query for a certificate of a public key PK, $\mathcal{B}$ retrieves corresponding user information ID from L, and computes $\mathfrak{u} = H_u(PK, ID)$. Although $\mathcal{B}$ does not know the master secret, it still can construct the certificate by assuming $F(\mathfrak{u}) \neq 0 \bmod p$. It randomly chooses $r_u \in_R \mathbb{Z}_p$ and computes

$$(\mathsf{cert}^{(1)}, \mathsf{cert}^{(2)}) = \left(g_1^{-\frac{J(\mathfrak{u})}{F(\mathfrak{u})}} \big(U\big)^{r_u}, g_1^{-\frac{1}{F(\mathfrak{u})}} g^{r_u} \right)$$

By letting $\tilde{r}_u = r_u - \frac{a}{F(\mathfrak{u})}$, it can be verifier that cert is a valid certificate, shown as follow:

$$\begin{aligned}
\mathsf{cert}^{(1)} &= g_1^{-\frac{J(\mathfrak{u})}{F(\mathfrak{u})}} \big(U\big)^{r_u} = g_1^{-\frac{J(\mathfrak{u})}{F(\mathfrak{u})}} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{r_u} = g^{-\frac{aJ(\mathfrak{u})}{F(\mathfrak{u})}} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{r_u} \\
&= g^{-\frac{aJ(\mathfrak{u})}{F(\mathfrak{u})}} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{\frac{a}{F(\mathfrak{u})}} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{-\frac{a}{F(\mathfrak{u})}} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{r_u} \\
&= g^{-\frac{aJ(\mathfrak{u})}{F(\mathfrak{u})}} g^{ab} g^{\frac{aJ(\mathfrak{u})}{F(\mathfrak{u})}} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{\tilde{r}_u} \\
&= g^{ab} (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{\tilde{r}_u} = g_2^a (g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})})^{\tilde{r}_u} = g_2^a \big(U\big)^{\tilde{r}_u}
\end{aligned}$$

and

$$\mathsf{cert}^{(2)} = g_1^{-\frac{1}{F(\mathfrak{u})}} g^{r_u} = g^{r_u - \frac{a}{F(\mathfrak{u})}} = g^{\tilde{r}_u}$$

To the adversary, all certificates given by $\mathcal{B}$ are indistinguishable from the those generated by the true challenger.

If $F(\mathfrak{u}) = 0 \bmod p$, since the above computation cannot be performed (division by 0), the simulator aborts. To make it simple, the simulator will abort if $F(\mathfrak{u}) = 0 \bmod l_u$. The equivalency can be observed as follow. From the assumption $l_u(n_u + 1) < p$, it implies $0 \leq l_u k_u < p$ and $0 \leq x' + \sum_{i \in \mathcal{U}_j} \hat{x}_i < p$ ($\because x' < l_u, \hat{x}_i < l_u, |\mathcal{U}| \leq n_u$). We have $-p < F(\mathfrak{u}) < p$ which implies if $F(\mathfrak{u}) = 0 \bmod p$ then $F(\mathfrak{u}) \bmod l_u$. Hence, $F(\mathfrak{u}) \neq 0 \bmod l_u$ implies $F(\mathfrak{u}) \neq 0 \bmod p$. Thus the former condition will be sufficient to ensure that a private key can be computed without abort.

(Signing Query.) For a given query of a signature on a public key PK with user information ID and a message m, $\mathcal{B}$ first computes $\mathfrak{u} = H_u(PK, ID)$ and $\mathfrak{m} = H_m(m)$. If the public key has been replaced with public key $(\mathsf{pk}^{(1)}, \mathsf{pk}^{(2)})$ (that is, the corresponding secret key is not in the list L), it computes the signature in the following way. Assume $K(\mathfrak{m}) \neq 0 \bmod l_m$. Using the argument mentioned above, it implies $K(\mathfrak{m}) \neq 0 \bmod p$ provided that $l_m(n_m + 1) < p$. The signature can be constructed by first randomly selecting $r_\pi, r_\mathfrak{m} \in_R \mathbb{Z}_p$, and computing

$$\sigma = \left(\big(U\big)^{r_\pi} \big(\mathsf{pk}^{(2)}\big)^{-\frac{L(\mathfrak{m})}{K(\mathfrak{m})}} \Big(m' \prod_{i \in \mathcal{M}} \hat{m}_i \Big)^{r_\mathfrak{m}} , g^{r_\pi} \ , \ \big(\mathsf{pk}^{(2)}\big)^{-\frac{1}{K(\mathfrak{m})}} g^{r_\mathfrak{m}} \right)$$
$$= \left(g_2^{ax} \big(U\big)^{r_\pi} \Big(m' \prod_{i \in \mathcal{M}} \hat{m}_i \Big)^{\tilde{r}_m} , \ g^{r_\pi} \ , \ g^{\tilde{r}_m} \right) = \big(V, R_\pi, R_m\big)$$

where $\tilde{r}_m = r_\mathfrak{m} - \frac{ax}{K(\mathfrak{m})}$. If $K(\mathfrak{m}) = 0 \bmod l_m$, the simulator aborts.

The correctness can be shown as follow:

$$V = \big(U\big)^{r_\pi} \big(\mathsf{pk}^{(2)}\big)^{-\frac{L(\mathfrak{m})}{K(\mathfrak{m})}} \Big(m' \prod_{i \in \mathcal{M}} \hat{m}_i \Big)^{r_m} = \big(U\big)^{r_\pi} g_1^{-\frac{L(\mathfrak{m})x}{K(\mathfrak{m})}} (g_2^{K(\mathfrak{m})} g^{L(\mathfrak{m})})^{r_m}$$
$$= \big(U\big)^{r_\pi} g^{-\frac{axL(\mathfrak{m})}{K(\mathfrak{m})}} (g_2^{K(\mathfrak{m})} g^{L(\mathfrak{m})})^{r_m}$$
$$= \big(U\big)^{r_\pi} g^{-\frac{axL(\mathfrak{m})}{K(\mathfrak{m})}} (g_2^{K(\mathfrak{m})} g^{L(\mathfrak{m})})^{\frac{ax}{K(\mathfrak{m})}} (g_2^{K(\mathfrak{m})} g^{L(\mathfrak{m})})^{-\frac{ax}{K(\mathfrak{m})}} (g_2^{K(\mathfrak{m})} g^{L(\mathfrak{m})})^{r_m}$$
$$= \big(U\big)^{r_\pi} g^{-\frac{axL(\mathfrak{m})}{K(\mathfrak{m})}} g^{abx} g^{\frac{axL(\mathfrak{m})}{K(\mathfrak{m})}} (g_2^{K(\mathfrak{m})} g^{L(\mathfrak{m})})^{\tilde{r}_m} = \big(U\big)^{r_\pi} g^{abx} (g_2^{K(\mathfrak{m})} g^{L(\mathfrak{m})})^{\tilde{r}_m}$$
$$= \big(U\big)^{r_\pi} g_2^{ax} (g_2^{K(\mathfrak{m})} g^{L(\mathfrak{m})})^{\tilde{r}_m} = \big(U\big)^{r_\pi} \big(g_2^a\big)^x \Big(m' \prod_{i \in \mathcal{M}} \hat{m}_i \Big)^{\tilde{r}_m}$$

and $R_m = \big(\mathsf{pk}^{(2)}\big)^{-\frac{1}{K(\mathfrak{m})}} g^{r_m} = g^{r_m - \frac{ax}{K(\mathfrak{m})}} = g^{\tilde{r}_m}$. The signature generated in this way is indistinguishable to the real one.

If the public key has not been replaced, it retrieves the corresponding secret key from L. If $F(\mathfrak{u}) \neq 0 \bmod l_u$, $\mathcal{B}$ constructs a certificate as in the certificate query, then it just uses the Sign algorithm to create a signature on ID, PK and m.

If $F(\mathfrak{u}) = 0 \bmod l_u$, $\mathcal{B}$ tries to construct the signature in a similar way as above (the case that the public key has been replaced). Assume $K(\mathfrak{m}) \neq 0 \bmod l_m$. Using the argument mentioned above, it implies $K(\mathfrak{m}) \neq 0 \bmod p$ provided that $l_m(n_m+1) < p$. The signature can be constructed by first randomly selecting $r_\pi, r_\mathfrak{m} \in_R \mathbb{Z}_p$, getting the secret key x from L and computing

$$\sigma = \left(\big(U\big)^{r_\pi} g_1^{-\frac{L(\mathfrak{m})}{K(\mathfrak{m})}x} \Big(m' \prod_{i \in \mathcal{M}} \hat{m}_i \Big)^{r_\mathfrak{m} x} , g^{r_\pi} \ , \ g_1^{-\frac{x}{K(\mathfrak{m})}} g^{r_\mathfrak{m} x} \right)$$
$$= \left(g_2^{ax} \big(U\big)^{r_\pi} \Big(m' \prod_{i \in \mathcal{M}} \hat{m}_i \Big)^{\tilde{r}_m} , \ g^{r_\pi} \ , \ g^{\tilde{r}_m} \right)$$

where $\tilde{r}_m = r_{\mathfrak{m}} x - \frac{a}{K(\mathfrak{m})} x$. If $K(\mathfrak{m}) = 0 \bmod l_m$, the simulator aborts.

Output Calculation. If $\mathcal{B}$ does not abort, $\mathcal{A}$ will return a public key PK^* with user information ID^* and a message m^* with a forged signature $\sigma^* = (V, R_\pi, R_{\mathfrak{m}})$ on ID^*, the current public key PK^* and m^* with probability at least ϵ. $\mathcal{B}$ checks whether the following conditions are fulfilled:

1. $F(\mathfrak{u}^*) = 0 \bmod p$, where $\mathfrak{u}^* = H_u(PK^*, ID^*)$.
2. $K(\mathfrak{m}^*) = 0 \bmod p$, where $\mathfrak{m}^* = H_m(m^*)$.

If not all the above conditions are fulfilled, $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ computes and outputs

$$\begin{aligned}
\frac{V}{R_\pi^{J(\mathfrak{u}^*)} R_{\mathfrak{m}}^{L(\mathfrak{m}^*)}} &= \frac{g_2^{ax} \left(U\right)^{r_\pi} \left(m' \prod_{i \in \mathcal{M}} \hat{m}_i\right)^{r_{\mathfrak{m}}}}{g^{J(\mathfrak{u}^*) r_\pi} g^{L(\mathfrak{m}^*) r_{\mathfrak{m}}}} \\
&= \frac{g_2^{ax} \left(g_2^{F(\mathfrak{u}^*)} g^{J(\mathfrak{u}^*)}\right)^{r_\pi} \left(g_2^{K(\mathfrak{m}^*)} g^{L(\mathfrak{m}^*)}\right)^{r_{\mathfrak{m}}}}{g^{J(\mathfrak{u}^*) r_\pi} g^{L(\mathfrak{m}^*) r_{\mathfrak{m}}}} \\
&= g_2^{ax} = g^{abx}
\end{aligned}$$

$\mathcal{B}$ outputs $(g^{abx}, \mathsf{pk}^{(1)}) = (g^{abx}, g^x)$ as the solution to the GCDH problem instance.

Probability Analysis. For the simulation to complete without aborting, we require the following conditions fulfilled:

1. Certification Queries on an identity ID have $F(\mathfrak{u}) \neq 0 \bmod l_u$, where $\mathfrak{u} = H_u(PK, ID)$.
2. Signing Queries (ID, PK, m) will either have $F(\mathfrak{u}) \neq 0 \bmod l_u$, or $K(\mathfrak{m}) \neq 0 \bmod l_m$ where $\mathfrak{m} = H_m(m)$, if the public key PK has not been replaced. Otherwise, it requires $K(\mathfrak{m}) \neq 0 \bmod l_m$.
3. $F(\mathfrak{u}^*) = 0 \bmod l_u$ and $K(\mathfrak{m}^*) = 0 \bmod l_m$.

In order to make the analysis simpler, we will bound the probability of a subcase of this event.

Let $\mathfrak{u}_1, \ldots, \mathfrak{u}_{q_I}$ be the output of the hash function H_u appearing in either Certification Queries or in Signing Queries not involving any of the challenge identity ID^*, and let $\mathfrak{m}_1, \ldots, \mathfrak{m}_{q_M}$ be the output of the hash function H_m in the sign queries involving the challenge list. We have $q_I \leq q_e + q_s$ and $q_M \leq q_s$. We also define the events A_i, A^*, B_ℓ, B^* as follow:

$$A_i : F(\mathfrak{u}_i) \neq 0 \bmod l_u \qquad \text{where } i = 1, \ldots, q_I \qquad A^* : F(\mathfrak{u}^*) = 0 \bmod p$$
$$B_\ell : K(\mathfrak{m}_\ell) \neq 0 \bmod l_m \qquad \text{where } \ell = 1, \ldots, q_M \qquad B^* : K(\mathfrak{m}^*) = 0 \bmod p$$

The probability of $\mathcal{B}$ not aborting is:

$$\Pr[\mathsf{not\ abort}] \geq \Pr\left[\left(\bigwedge_{i=1}^{q_I} A_i \wedge A^*\right) \wedge \left(\bigwedge_{\ell=1}^{q_M} B_\ell \wedge B^*\right)\right]$$

Note that the events $\left(\bigwedge_{i=1}^{q_I} A_i \wedge A^*\right)$ and $\left(\bigwedge_{\ell=1}^{q_M} B_\ell \wedge B^*\right)$ are independent.

The assumption $l_u(n_u + 1) < p$ implies if $F(\mathfrak{u}) = 0 \bmod p$ then $F(\mathfrak{u}) = 0 \bmod l_u$. In addition, it also implies that if $F(\mathfrak{u}) = 0 \bmod l_u$, there will be a unique choice of k_u with $0 \leq k_u \leq n_u$ such that $F(\mathfrak{u}) = 0 \bmod p$. Since k_u, x' and $\hat{X}$ are randomly chosen,

$$\begin{aligned}\Pr[A^*] &= \Pr[F(\mathfrak{u}^*) = 0 \bmod p \wedge F(\mathfrak{u}^*) = 0 \bmod l_u] \\ &= \Pr[F(\mathfrak{u}^*) = 0 \bmod l_u] \Pr[F(\mathfrak{u}^*) = 0 \bmod p \mid F(\mathfrak{u}^*) = 0 \bmod l_u] \\ &= \frac{1}{l_u} \frac{1}{n_u + 1}\end{aligned}$$

On the other hand, we have:

$$\Pr\left[\bigwedge_{i=1}^{q_I} A_i | A^*\right] = 1 - \Pr\left[\bigvee_{i=1}^{q_I} \overline{A_i} \mid A^*\right] \geq 1 - \sum_{i=1}^{q_I} \Pr[\overline{A_i} \mid A^*]$$

where $\overline{A_i}$ denote the event $F(\mathfrak{u}_i) = 0 \bmod l_u$.

Also note that the events $F(\mathfrak{u}_{i_1}) = 0 \bmod l_u$ and $F(\mathfrak{u}_{i_2}) = 0 \bmod l_u$ are independent, where $i_1 \neq i_2$, since the outputs of $F(\mathfrak{u}_{i_1})$ and $F(\mathfrak{u}_{i_2})$ will differ in at least one randomly chosen value. Also since the events A_i and A^* are independent for any i, we have $\Pr[\overline{A_i}|A^*] = 1/l_u$ and

$$\begin{aligned}\Pr\left[\bigwedge_{i=1}^{q_I} A_i \wedge A^*\right] &= \Pr[A^*] \Pr\left[\bigwedge_{i=1}^{q_I} A_i | A^*\right] \\ &= \frac{1}{l_u(n_u + 1)}\left(1 - \frac{q_I}{l_u}\right) \\ &\geq \frac{1}{l_u(n_u + 1)}\left(1 - \frac{q_e + q_s}{l_u}\right) \\ &= \frac{1}{2(q_e + q_s)(n_u + 1)}\left(1 - \frac{1}{2}\right) (\text{ by setting } l_u = 2(q_e + q_s) \ \ ) \\ &= \frac{1}{4(q_e + q_S)(n_u + 1)}\end{aligned}$$

Using similar analysis technique for signing queries we can have:

$$\Pr\left[\bigwedge_{\ell=1}^{q_M} B_\ell \wedge B^*\right] \geq \frac{1}{4q_s(n_m + 1)}$$

By combining the above result, we have

$$\begin{aligned}\Pr[\mathsf{not\ abort}] &\geq \Pr\left[\left(\bigwedge_{i=1}^{q_I} A_i \wedge A^*\right) \wedge \left(\bigwedge_{\ell=1}^{q_M} B_\ell \wedge B^*\right)\right] \\ &\geq \frac{1}{16(q_e + q_s)(n_u + 1)q_s(n_m + 1)}\end{aligned}$$

If the simulation does not abort, $\mathcal{A}$ will produce a forged signature with probability at least ϵ. Thus $\mathcal{B}$ can solve for the GCDH problem instance with probability

$$\epsilon' \geq \frac{\epsilon}{16(q_e + q_s)(n_u + 1)q_s(n_m + 1)}$$

Time Complexity Analysis. The time complexity of $\mathcal{B}$ is dominated by the exponentiation and multiplication operations for large values of n_u and n_m performed in the partial secret key extraction and signing queries.

There are $O(n_u)$ and $O(n_u + n_m)$ multiplications and $O(1)$ and $O(1)$ exponentiations in the certification query and singing query respectively. The time complexity of $\mathcal{B}$ is $t + O\Big(\big(q_e n_u + q_s(n_u + n_m)\big)\rho + (q_e + q_s)\tau\Big)$ □

B Proof of Theorem 4

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 Many-DH problem with probability at least ϵ' and in time at most t'.

$\mathcal{B}$ is given a problem instance as follow: Given a group $\mathbb{G}$, a generator $g \in \mathbb{G}$, 6 elements $g^a, g^b, g^x, g^{ab}, g^{ax}, g^{bx} \in \mathbb{G}$. It is asked to output an element $g^{abx} \in \mathbb{G}$. In order to use $\mathcal{A}$ to solve for the problem, $\mathcal{B}$ needs to simulate a challenger and all oracles for $\mathcal{A}$. $\mathcal{B}$ does it in the following way.

Setup. Let $l_u = 2(q_e + q_s)$ and $l_m = 2q_s$. $\mathcal{B}$ randomly selects two integers k_u and k_m such that $0 \leq k_u \leq n_u$ and $0 \leq k_m \leq n_m$. Also assume that $l_u(n_u + 1) < p$ and $l_m(n_m + 1) < p$ for the given values of q_s, n_u and n_m. It randomly selects the following integers:

$$\begin{aligned}
&x' \in_R \mathbb{Z}_{l_u} \ ; \ z' \in_R \mathbb{Z}_{l_m} \ ; \ y', w' \in_R \mathbb{Z}_p \\
&\hat{x}_i \in_R \mathbb{Z}_{l_u}, \text{ for } i = 1, \ldots, n_u \\
&\hat{z}_i \in_R \mathbb{Z}_{l_m}, \text{ for } i = 1, \ldots, n_m \\
&\hat{y}_i \in_R \mathbb{Z}_p, \text{ for } i = 1, \ldots, n_u \\
&\hat{w}_i \in_R \mathbb{Z}_p, \text{ for } i = 1, \ldots, n_m \\
&\text{Let } \hat{X} = \{\hat{x}_i\}, \hat{Z} = \{\hat{z}_i\}, \hat{Y} = \{\hat{y}_i\}, \hat{W} = \{\hat{w}_i\}
\end{aligned}$$

We further define the following functions for binary strings $\mathfrak{u}$ and $\mathfrak{m}$ where $\mathfrak{u} = H_u(PK, ID)$ for a public key PK with user identity ID and $\mathfrak{m} = H_m(m)$ for a message m, as follow:

$$F(\mathfrak{u}) = x' + \sum_{i \in \mathcal{U}_j} \hat{x}_i - l_u k_u \qquad J(\mathfrak{u}) = y' + \sum_{i \in \mathcal{U}_j} \hat{y}_i$$

$$K(\mathfrak{m}) = z' + \sum_{i \in \mathcal{M}} \hat{z}_i - l_m k_m \qquad L(\mathfrak{m}) = w' + \sum_{i \in \mathcal{M}} \hat{w}_i$$

$\mathcal{B}$ constructs a set of public parameters as follow:

$$
\begin{aligned}
& g_1 = g^a, \qquad g_2 = g^b, \qquad (g_2)^\alpha = g^{ab} \\
& \mathsf{pk}^{*(1)} = g^x, \qquad \mathsf{pk}^{*(2)} = g^{ax}, \qquad u' = g_2^{-l_u k_u + x'} g^{y'} \\
& \hat{u}_i = g_2^{\hat{x}_i} g^{\hat{y}_i} \quad \text{for} \quad 1 \le i \le n_u, \\
& m' = g_2^{-l_m k_m + z'} g^{w'}, \qquad \hat{m}_i = g_2^{\hat{z}_i} g^{\hat{w}_i} \quad \text{for} \quad 1 \le i \le n_m
\end{aligned}
$$

for a randomly chosen public key PK^* (with the corresponding identity information ID^*), and we have the following equations:

$$
U = u' \prod_{i \in \mathcal{U}} \hat{u}_i = g_2^{F(\mathfrak{u})} g^{J(\mathfrak{u})} \qquad \text{and} \qquad m' \prod_{i \in \mathcal{M}} \hat{m}_i = g_2^{K(\mathfrak{m})} g^{L(\mathfrak{m})}
$$

All public parameters and master secret key $g_2^\alpha = g^{ab}$ are passed to $\mathcal{A}$.

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

(User-key-gen Queries.) $\mathcal{B}$ keeps the list L of user secret-public key. It first puts the public key of the identity ID^* into L. Upon receiving a query for a public key of an identity ID, $\mathcal{B}$ looks up its database L to find out the corresponding entry. If it does not exits, $\mathcal{B}$ runs UserKeyGen to generate a secret and public key pair. It stores the key pair in its database and returns the public key as the query output.

(Corruption Queries.) If the adversary asks for the secret key of ID^*, $\mathcal{B}$ aborts. Otherwise, it returns the corresponding entry from L.

(Signing Queries) For a given query of a signature on public key PK with the corresponding identity information ID and a message m, $\mathcal{B}$ first checks if the identity is equal to ID^*. If yes, computes the signature in the following way. Assume $K(\mathfrak{m}) \neq 0 \bmod l_m$. Using the argument mentioned above, it implies $K(\mathfrak{m}) \neq 0 \bmod p$ provided that $l_m(n_m+1) < p$. The signature can be constructed by first randomly selecting $r_\pi, r_\mathfrak{m} \in_R \mathbb{Z}_p$, and computing

$$
\begin{aligned}
\sigma &= \left((U)^{r_\pi} \left(\mathsf{pk}^{*(2)}\right)^{-\frac{L(\mathfrak{m})}{K(\mathfrak{m})}} \left(m' \prod_{i \in \mathcal{M}} \hat{m}_i \right)^{r_\mathfrak{m}} , g^{r_\pi} \ , \ \left(\mathsf{pk}_{\mathsf{ID}^*}^{(2)}\right)^{-\frac{1}{K(\mathfrak{m})}} g^{r_\mathfrak{m}} \right) \\
&= \left(g_2^{ax} (U)^{r_\pi} \left(m' \prod_{i \in \mathcal{M}} \hat{m}_i \right)^{\tilde{r}_m} , \ g^{r_\pi} \ , \ g^{\tilde{r}_m} \right) = \left(V, R_\pi, R_m \right)
\end{aligned}
$$

where $\tilde{r}_m = r_\mathfrak{m} - \frac{ax}{K(\mathfrak{m})}$. If $K(\mathfrak{m}) = 0 \bmod l_m$, the simulator aborts.

The correctness can be shown as follow:

$$\begin{aligned}
V &= \big(U\big)^{r_\pi}\big(\mathsf{pk}^{*(2)}\big)^{-\frac{L(\mathfrak{m})}{K(\mathfrak{m})}}\Big(m'\prod_{i\in\mathcal{M}}\hat{m}_i\Big)^{r_m}\\
&= \big(U\big)^{r_\pi} g_1^{-\frac{L(\mathfrak{m})x}{K(\mathfrak{m})}}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{r_m}\\
&= \big(U\big)^{r_\pi} g^{-\frac{axL(\mathfrak{m})}{K(\mathfrak{m})}}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{r_m}\\
&= \big(U\big)^{r_\pi} g^{-\frac{axL(\mathfrak{m})}{K(\mathfrak{m})}}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{\frac{ax}{K(\mathfrak{m})}}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{-\frac{ax}{K(\mathfrak{m})}}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{r_m}\\
&= \big(U\big)^{r_\pi} g^{-\frac{axL(\mathfrak{m})}{K(\mathfrak{m})}} g^{abx} g^{\frac{axL(\mathfrak{m})}{K(\mathfrak{m})}}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{\tilde{r}_m}\\
&= \big(U\big)^{r_\pi} g^{abx}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{\tilde{r}_m}\\
&= \big(U\big)^{r_\pi} g_2^{ax}(g_2^{K(\mathfrak{m})}g^{L(\mathfrak{m})})^{\tilde{r}_m}\\
&= \big(U\big)^{r_\pi}\big(g_2^{a}\big)^{x}\Big(m'\prod_{i\in\mathcal{M}}\hat{m}_i\Big)^{\tilde{r}_m}
\end{aligned}$$

and $R_m = \big(\mathsf{pk}^{*(2)}\big)^{-\frac{1}{K(\mathfrak{m})}} g^{r_m} = g^{r_m - \frac{ax}{K(\mathfrak{m})}} = g^{\tilde{r}_m}$. The signature generated in this way is indistinguishable to the real one.

If it is not equal to ID^*, it computes $\mathfrak{u} = H_u(PK, ID)$ and $\mathfrak{m} = H_m(m)$.

If $F(\mathfrak{u}) \neq 0 \bmod l_u$, $\mathcal{B}$ just construct a certificate using the knowledge of msk, then it checks from L whether the secret key of ID has been created or not. If it has not been created, run the $\mathsf{UserKeyGen}$ algorithm and stores the secret / public key pair in L. If it has been created, it just uses the Sign algorithm to create a signature on PK, ID and m.

If $F(\mathfrak{u}) = 0 \bmod l_u$, $\mathcal{B}$ tries to construct the signature in a similar way as above (the case that the public key has been replaced). Assume $K(\mathfrak{m}) \neq 0 \bmod l_m$. Using the argument mentioned above, it implies $K(\mathfrak{m}) \neq 0 \bmod p$ provided that $l_m(n_m+1) < p$. The signature can be constructed by first randomly selecting $r_\pi, r_\mathfrak{m} \in_R \mathbb{Z}_p$, getting the secret key x from L (if it has not been created, run $\mathsf{UserKeyGen}$ algorithm first) and computing

$$\begin{aligned}
\sigma &= \left(\big(U\big)^{r_\pi} g_1^{-\frac{L(\mathfrak{m})}{K(\mathfrak{m})}x}\left(m'\prod_{i\in\mathcal{M}}\hat{m}_i\right)^{r_\mathfrak{m} x}, g^{r_\pi}\ ,\ g_1^{-\frac{x}{K(\mathfrak{m})}} g^{r_\mathfrak{m} x}\right)\\
&= \left(g_2^{ax}\big(U\big)^{r_\pi}\left(m'\prod_{i\in\mathcal{M}}\hat{m}_i\right)^{\tilde{r}_m}\ ,\ g^{r_\pi}\ ,\ g^{\tilde{r}_m}\right)
\end{aligned}$$

where $\tilde{r}_m = r_\mathfrak{m} x - \frac{a}{K(\mathfrak{m})}x$. If $K(\mathfrak{m}) = 0 \bmod l_m$, the simulator aborts.

Output Calculation. If $\mathcal{B}$ does not abort, $\mathcal{A}$ will return a public key PK^* with the corresponding identity ID^* and a message m^* with a forged signature $\sigma^* = (V, R_\pi, R_\mathfrak{m})$ on ID^*, the current public key PK^* and m^* with probability at least ϵ. $\mathcal{B}$ checks whether the following conditions are fulfilled:

1. $F(\mathfrak{u}^*) = 0 \bmod p$, where $\mathfrak{u}^* = H_u(PK^*, ID^*)$.
2. $K(\mathfrak{m}^*) = 0 \bmod p$, where $\mathfrak{m}^* = H_m(m^*)$.

If not all the above conditions are fulfilled, $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ computes and outputs

$$\begin{aligned}\frac{V}{R_\pi^{J(\mathfrak{u}^*)} R_{\mathfrak{m}}^{L(\mathfrak{m}^*)}} &= \frac{g_2^{ax}(U)^{r_\pi}\left(m' \prod_{i\in\mathcal{M}} \hat{m}_i\right)^{r_{\mathfrak{m}}}}{g^{J(\mathfrak{u}^*)r_\pi} g^{L(\mathfrak{m}^*)r_{\mathfrak{m}}}} \\ &= \frac{g_2^{ax}\left(g_2^{F(\mathfrak{u}^*)} g^{J(\mathfrak{u}^*)}\right)^{r_\pi}\left(g_2^{K(\mathfrak{m}^*)} g^{L(\mathfrak{m}^*)}\right)^{r_{\mathfrak{m}}}}{g^{J(\mathfrak{u}^*)r_\pi} g^{L(\mathfrak{m}^*)r_{\mathfrak{m}}}} \\ &= g_2^{ax} = g^{abx}\end{aligned}$$

$\mathcal{B}$ outputs g^{abx} as the solution to the Many-DH problem instance.

Probability Analysis. For the simulation to complete without aborting, we require the following conditions fulfilled:

1. Signing queries (ID, PK, m) will either have $F(\mathfrak{u}) \neq 0 \bmod l_u$, or $K(\mathfrak{m}) \neq 0 \bmod l_m$ where $\mathfrak{m} = H_m(m)$, if $ID \neq ID^*$. Otherwise, it requires $K(\mathfrak{m}) \neq 0 \bmod l_m$.
2. $F(\mathfrak{u}^*) = 0 \bmod l_u$ and $K(\mathfrak{m}^*) = 0 \bmod l_m$.

In addition, in order to get the desired result, it is required that $\mathcal{A}$ has chosen PK^* with ID^* for the signature forgery.

To make the analysis simpler, we will bound the probability of a subcase of this event.

Let $\mathfrak{u}_1, \ldots, \mathfrak{u}_{q_I}$ be the output of the hash function H_u appearing in Signing queries not involving any of the challenge identity ID^*, and let $\mathfrak{m}_1, \ldots, \mathfrak{m}_{q_M}$ be the output of the hash function H_m in the sign queries involving the challenge list. We have $q_I \leq q_s \leq q_e + q_s$ and $q_M \leq q_s$. We also define the events A_i, A^*, B_ℓ, B^* as follow:

$$\begin{array}{llll} A_i : F(\mathfrak{u}_i) \neq 0 \bmod l_u & \text{where } i = 1, \ldots, q_I & A^* : F(\mathfrak{u}^*) = 0 \bmod p \\ B_\ell : K(\mathfrak{m}_\ell) \neq 0 \bmod l_m & \text{where } \ell = 1, \ldots, q_M & B^* : K(\mathfrak{m}^*) = 0 \bmod p \end{array}$$

The probability of $\mathcal{B}$ not aborting is:

$$\Pr[\text{not abort}] \geq \Pr\left[\left(\bigwedge_{i=1}^{q_I} A_i \wedge A^*\right) \wedge \left(\bigwedge_{\ell=1}^{q_M} B_\ell \wedge B^*\right)\right]$$

Note that the events $\left(\bigwedge_{i=1}^{q_I} A_i \wedge A^*\right)$ and $\left(\bigwedge_{\ell=1}^{q_M} B_\ell \wedge B^*\right)$ are independent.

The assumption $l_u(n_u + 1) < p$ implies if $F(\mathfrak{u} = 0 \bmod p$ then $F(\mathfrak{u}) = 0 \bmod l_u$. In addition, it also implies that if $F(\mathfrak{u}) = 0 \bmod l_u$, there will be a unique choice of k_u with $0 \leq k_u \leq n_u$ such that $F(\mathfrak{u}) = 0 \bmod p$. Since k_u, x' and $\hat{X}$ are randomly chosen,

$$\begin{aligned}\Pr[A^*] &= \Pr[F(\mathfrak{u}^*) = 0 \bmod p \wedge F(\mathfrak{u}^*) = 0 \bmod l_u] \\ &= \Pr[F(\mathfrak{u}^*) = 0 \bmod l_u] \Pr[F(\mathfrak{u}^*) = 0 \bmod p \mid F(\mathfrak{u}^*) = 0 \bmod l_u] \\ &= \frac{1}{l_u}\frac{1}{n_u + 1}\end{aligned}$$

On the other hand, we have:

$$\Pr\left[\bigwedge_{i=1}^{q_I} A_i | A^*\right] = 1 - \Pr\left[\bigvee_{i=1}^{q_I} \overline{A_i} \mid A^*\right] \geq 1 - \sum_{i=1}^{q_I} \Pr[\overline{A_i} \mid A^*]$$

where $\overline{A_i}$ denote the event $F(\mathtt{u}_i) = 0 \bmod l_u$.

Also note that the events $F(\mathtt{u}_{i_1}) = 0 \bmod l_u$ and $F(\mathtt{u}_{i_2}) = 0 \bmod l_u$ are independent, where $i_1 \neq i_2$, since the outputs of $F(\mathtt{u}_{i_1})$ and $F(\mathtt{u}_{i_2})$ will differ in at least one randomly chosen value. Also since the events A_i and A^* are independent for any i, we have $\Pr[\overline{A_i}|A^*] = 1/l_u$ and

$$\begin{aligned}
\Pr\left[\bigwedge_{i=1}^{q_I} A_i \wedge A^*\right] &= \Pr[A^*] \Pr\left[\bigwedge_{i=1}^{q_I} A_i | A^*\right] \\
&= \frac{1}{l_u(n_u+1)}\left(1 - \frac{q_I}{l_u}\right) \\
&\geq \frac{1}{l_u(n_u+1)}\left(1 - \frac{q_e+q_s}{l_u}\right) \\
&= \frac{1}{2(q_e+q_s)(n_u+1)}\left(1 - \frac{1}{2}\right) (\text{ by setting } l_u = 2(q_e+q_s) \ \ ) \\
&= \frac{1}{4(q_e+q_S)(n_u+1)}
\end{aligned}$$

Using similar analysis technique for signing queries we can have:

$$\Pr\left[\bigwedge_{\ell=1}^{q_M} B_\ell \wedge B^*\right] \geq \frac{1}{4q_s(n_m+1)}$$

By combining the above result, we have

$$\begin{aligned}
\Pr[\mathsf{not\ abort}] &\geq \Pr\left[\left(\bigwedge_{i=1}^{q_I} A_i \wedge A^*\right) \wedge \left(\bigwedge_{\ell=1}^{q_M} B_\ell \wedge B^*\right)\right] \\
&\geq \frac{1}{16(q_e+q_s)(n_u+1)q_s(n_m+1)}
\end{aligned}$$

If the simulation does not abort, $\mathcal{A}$ will produce a forged signature with probability at least ϵ. In addition, $\mathcal{B}$ needs to guess which identity $\mathcal{A}$ is going to forge the signature, and assign the problem instance element as the public key of this identity. The probability of guessing correctly is $1/q_k$. Thus $\mathcal{B}$ can solve for the Many-DH problem instance with probability

$$\epsilon' \geq \frac{\epsilon}{16(q_e+q_s)(n_u+1)q_s(n_m+1)q_k}$$

Time Complexity Analysis. It is similar to the proof of Game 1 Adversary except the removal of the certificate query in Game 2 Adversary. We skip here.

□