# Hierarchical Certificateless Signatures

Lei Zhang[1], Qianhong Wu[1,2], Josep Domingo-Ferrer[1], Bo Qin[1,3]

[1]UNESCO Chair in Data Privacy, Department of Computer Engineering and Mathematics, Universitat Rovira i Virgili, Catalonia
[2]Wuhan University, School of Computer, Key Lab. of Aerospace Information Security and Trusted Computing, China
[3]Xi'an University of Technology, School of Science, Department of Maths, China
Email: {lei.zhang,qianhong.wu,josep.domingo,bo.qin}@urv.cat

***Abstract*—Certificateless cryptography eliminates the key escrow problem in identity-based cryptography. Hierarchical cryptography exploits a practical security model to mirror the organizational hierarchy in the real world. In this paper, to incorporate the advantages of both types of cryptosystems, we instantiate hierarchical certificateless cryptography by formalizing the notion of hierarchical certificateless signatures. Furthermore, we propose an HCLS scheme which, under the hardness of the computational Diffie-Hellman (CDH) problem, is proven to be existentially unforgeable against adaptive chosen-message attacks in the random oracle model. As to efficiency, our scheme has constant complexity, regardless of the depth of the hierarchy. Hence, the proposal is secure and scalable for practical applications.**

**Index Terms: Hierarchical cryptography, Hierarchical certificateless signature, CDH assumption.**

## I. Introduction

The digital signature is one of the most important cryptographic primitives. It can provide authenticity, integrity and non-repudiation for various applications. Security systems which match the application environment better can be deployed more easily. Noticing that most organizations are usually organized hierarchically [10], this paper concentrates on hierarchical certificateless signatures.

**Related Work.** When a digital signature scheme is deployed, to let the digital signature system work properly, a user must be bound with his public key. In traditional public key cryptosystems, this binding is provided by a public-key infrastructure (PKI) in which a trusted certificate authority authenticates the user's public key. However, in practice, the management of public-key certificates requires huge efforts in terms of computing time and storage space.

To reduce the certificate management overhead in traditional public key cryptosystems, identity-based cryptography (IBC) was introduced by Shamir in 1984. In an IBC system, each user's public key is his identity in the form of an email account, IP address and so on. A trusted third party called Private Key Generator (PKG) generates the private key for each user by taking as input PKG's master secret key and the user's identity. Since the users' public information can be used as their public keys, there is no need to maintain a complicated management system to administrate the addition, update or withdrawal of certificates. Hence, IBC systems are very efficient for some applications.

It is also known that IBC cryptosystems suffer from two limitations in practice. The first limitation is the single-point problem. In a basic IBC system, there is only a *single* PKG. Although having a single PKG would completely eliminate online lookup, it is undesirable for a large open network because the PKG may become a bottleneck: the PKG needs not only to generate private keys for a large number of users, but also to verify the identities of the users. The second limitation is the so-called secret key escrow problem. In an IBC system, the PKG knows each user's private key and each user must unconditionally trust the PKG. A malicious PKG can forge signatures on behalf of any user without being detected. The above two limitations make IBC systems not scalable and not applicable to large open networks in which it is impractical to realize a third party fully trusted by all the distributed users.

The concept of hierarchical identity-based cryptography (HIBC) was first instantiated in [6]. It efficiently overcomes the single-point problem in IBC schemes. HIBC allows a *root* PKG to distribute the workload by delegating private key generation and identity authentication to *lower-level* PKGs. In this setting, multiple levels of PKGs and users as leaf nodes form a tree-like structure. Similarly to the existing hierarchical PKI model, this hierarchy is consistent with the hierarchical social organization in the real world. Subsequent to the work in [6], Gentry and Silverberg [5] proposed a scalable HIBE scheme with full collusion resistance and chosen-ciphertext security. A hierarchical identity-based signature (HIBS) scheme was proposed in [5] as well. The concepts of HIBE and HIBS have been further investigated and more works can be found in [2], [3], [4], [11], [12].

Although HIBC efficiently overcomes the single-point bottleneck of IBC systems when they are deployed for large organizations, it does not address the secret key escrow problem and cannot be applied to open networks. To mitigate this limitation, Al-Riyami and Paterson introduced certificateless cryptography (CLC). In a CLC scheme, a third party called Key Generation Center (KGC) is employed to help a user generate his private key. Unlike the PKG in IBC, KGC is not allowed to access a user's full private key. Instead, the KGC supplies the user with a partial private key, which is computed from the user's identity and the master key of the KGC. Then the user combines the partial private key with some secret information of his own choice to generate the full private key.

A number of efforts [1], [7], [8], [9], [15], [17] have been devoted to the construction of secure CLC schemes. In [1], Al-Riyami and Paterson proposed a certificateless encryption (CLE) scheme with chosen-ciphertext security. A certificate-

less signature (CLS) scheme was also presented in [1] but no formal proof of security was provided. Subsequently, the security model of CLS schemes was formalized and the security of the Al-Riyami-Paterson CLS scheme was analyzed in this model [9]. The security model of CLS schemes was further developed in [15] and later in [7], [8]. Similarly to the notion of HIBC overcoming the single-point problem, CLC effectively avoids the secret key escrow problem, but it still suffers from the same single-point limitation of basic IBC systems since the single KGC in a CLC system has to help all the users to produce their full private keys.

**Our Contributions.** One may note that neither CLC nor HIBC address the single-point and the secret key escrow problems in IBC systems *simultaneously*. To fill this gap, this paper concentrates on hierarchical certificateless cryptography (HCLC) and, specifically, on hierarchical certificateless signatures (HCLSs). This paper is twofold: the notion of hierarchical certificateless signatures and a provably secure HCLS scheme.

*The notion of HCLSs.* HCLSs inherit the advantage of HIBSs while solving the escrow problem of HIBSs. Note that a hierarchical certificateless encryption scheme [1] was presented recently as an HCLC primitive but no security formalization has been provided. In this paper, we present the first formalization of HCLSs as a new HCLC primitive. In an HCLS scheme, a *root* KGC distributes the workload by delegating partial private key generation and identity authentication to *lower-level* KGCs. Multiple levels of KGCs and users as leaf nodes form a tree-like structure. This hierarchical design matches the structure of social organizations in the real world very well. We explicitly define the behaviors of attackers against HCLSs. Two types of attackers are distinguished to simulate the malicious KGCs and the collusive users, respectively. Then we define security as existential unforgeability against adaptive chosen-message attacks. We believe this strong security notion is suitable for most applications.

*A provably secure HCLS scheme.* We propose a scalable HCLS scheme in which, to verify the validity of a signature from a signer, a verifier needs only to obtain the public parameters of the signer's *root* KGC, the user's identifying information (a user is uniquely identified by the path from the root to the user as a leaf node in the hierarchical tree) and the corresponding public key list. The verifier does not need an *on-line* lookup of the identities and public keys of the *lower-level* KGCs. The signing cost and the signature size are both constant, independent of the hierarchy depth. As for security, under the CDH assumption, our scheme is provably secure against existential forgery attacks in the random oracle model. Hence, it is secure and practical to many potential applications, *e.g.* to simplify the anonymous certificate management in large-scale vehicular *ad hoc* networks [16], [13].

## II. Definitions

### A. Hierarchical Certificateless Signatures

In an HCLS scheme, the entities include the root KGC, lower-level KGCs and users. Each entity has a unique identity $ID$. These entities form a hierarchical tree structure where the root KGC is at the top and the users at the bottom, with the lower-level KGCs as the nodes between them. In the hierarchy, each entity is uniquely identifiable by the path from the root KGC to the entity itself. Hence, we use an identity tuple $(ID_0, ..., ID_n)$ as the information to identify the position of an entity with $ID_n$ in the hierarchy, where $n$ is the tree depth from the root KGC with $ID_0$ to the entity with $ID_n$.

For $0 \leq i \leq n-1$ and $n \geq 1$, the entity with identity $ID_i$ is the parent node of the entity with identity $ID_{i+1}$. The entity with identity $ID_n$ can be either a lower-level KGC or a user as a leaf node. For clarity, we use $\mathcal{K}_n$ to represent a *lower-level* KGC at level $n$, and $\mathcal{U}_n$ to represent a user at the same level, respectively. Specially, $\mathcal{K}_0$ denotes the *root* KGC with identity $ID_0$, and if $n = 1$ for all the users, the system degenerates to a regular certificateless signature scheme.

An HCLS scheme consists of the following eight polynomial-time (probabilistic or deterministic) algorithms:

- Root-Setup: The *root* KGC $\mathcal{K}_0$ takes as input a security parameter $\lambda$, and returns a *root* private key $s_0$ and a list of public system parameters $params$. The list $params$ will be a common input to the rest of algorithms and for simplicity we omit it in the following specifications.
- Lower-Level-Setup: This is an interactive protocol between $\mathcal{K}_n$ with identifying information $(ID_0, ..., ID_n)$ and its parent $\mathcal{K}_{n-1}$ with identifying information $(ID_0, ..., ID_{n-1})$. It allows lower-level KGCs to be added to the system. At the beginning of this protocol, $\mathcal{K}_n$ selects a secret value and generates its public key; then $\mathcal{K}_n$ sends its identity and public key to $\mathcal{K}_{n-1}$. After receiving the information, $\mathcal{K}_{n-1}$ generates the partial private key for $\mathcal{K}_n$ and forwards it to $\mathcal{K}_n$. If this protocol ends without failing, $\mathcal{K}_n$ obtains its full private key.
- Set-Secret-Value: This algorithm is performed by $\mathcal{U}_n$. On input $\mathcal{U}_n$'s identifying information $(ID_0, ..., ID_n)$ and an ordered public key list $(P_0, ..., P_{n-1})$, it outputs a secret value $s_n$, where $P_i, 1 \leq i \leq n-1$ is the public key of $\mathcal{K}_i$.
- Set-Public-Key: This algorithm is performed by $\mathcal{U}_n$. The input of this algorithm is $\mathcal{U}_n$'s secret value $s_n$, his identifying information $(ID_0, ..., ID_n)$ and the corresponding public key list $(P_0, ..., P_{n-1})$. The output of this algorithm is $\mathcal{U}_n$'s public key $P_n$.
- Partial-Private-Key-Extract: $\mathcal{U}_n$ may request his parent $\mathcal{K}_{n-1}$ to run this algorithm to generate a partial private key for him. The input of this algorithm is $\mathcal{U}_n$'s identifying information $(ID_0, ..., ID_n)$, the corresponding public key list $(P_0, ..., P_n)$ and $\mathcal{K}_{n-1}$'s private key. The output of this algorithm is a partial private key $D_n$ for $\mathcal{U}_n$.
- Set-Private-Key: This algorithm is run by $\mathcal{U}_n$. The input of this algorithm is $\mathcal{U}_n$'s secret value $s_n$, partial private key $D_n$, identifying information $(ID_0, ..., ID_n)$ and the corresponding public key list $(P_0, ..., P_n)$. The output of this algorithm is $\mathcal{U}_n$'s private key $S_n$.
- Sign: This algorithm is run by a user (signer) $\mathcal{U}_n$. It takes as input a message $m$, $\mathcal{U}_n$'s private key $S_n$, identifying

information $(ID_0, ..., ID_n)$ and the corresponding public key list $(P_0, ..., P_n)$, and outputs a signature $\sigma$ on message $m$ under $(ID_0, ..., ID_n, P_0, ..., P_n)$.
- Verify: This algorithm is run by a verifier. It takes as input a message $m$, a purported signature $\sigma$, the signer's identifying information $(ID_0, ..., ID_n)$ and the public key list $(P_0, ..., P_n)$, and outputs $true$ or $false$ to represent that $\sigma$ is valid signature or not, respectively.

### B. Security Model

Before defining the security of HCLS schemes, we first review the adversaries in CLC. A secure CLS scheme should resist the attacks from two types of adversaries, known as 'Type I adversary' and 'Type II adversary'. In [1], these two types of adversaries are respectively defined as follows:

- Type I adversary is an attacker (colluding with users) who does not have the knowledge of KGC's private key, but he can replace the public key of any user with a value of his choice.
- Type II adversary is an attacker (colluding with KGC) who owns KGC's private key but cannot perform any public key replacement.

In [7], [8], Type II adversary is enhanced by additionally allowing it to replace the public key of any user except the target one. That is, the attacker can collude with the KGC and the users except the target user. We observe that this type of attack is also possible in our hierarchical setting. Hence, we respectively define the Type I adversary and the Type II adversary as follows:

- Type I adversary $\mathcal{A}_I$ does not have the knowledge of the *root* KGC's private key or the partial private key (or private key) of any *lower-level* KGC, but he can replace the public key of any entity except the root KGC with a value of his choice.
- Type II adversary $\mathcal{A}_{II}$ knows the private key of any KGC (including the *root* and all *lower-level* KGCs), and can replace the public key of any *lower-level* KGC and user except $\mathcal{A}_{II}$'s target user.

Now we present the security model for HCLS schemes. The model is defined by the following two games played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}_I$ or $\mathcal{A}_{II}$.

Game I (for *Type I Adversary*)

This game is played between a challenger $\mathcal{C}$ and a Type I adversary $\mathcal{A}_I$, and comprises three phases.

*Phase I-1*: $\mathcal{C}$ first runs Root-Setup to generate the *root* private key and the system parameter list $params$. Then $\mathcal{C}$ sends $params$ to $\mathcal{A}_I$ but keeps the *root* private key.

*Phase I-2*: The adversary can access the following oracles which are controlled by $\mathcal{C}$:

- Lower-Level-Setup Oracle: On input $\mathcal{K}_n$'s identifying information $(ID_0, ..., ID_n)$, this oracle outputs the secret value $s_n$ and public key $P_n$ for $\mathcal{K}_n$. Here, we require that the public key of $\mathcal{K}_i$ has been already generated for $1 \leq i \leq n-1$. In other words, the attacker can only query $(ID_0, ..., ID_n)$ after it queries $(ID_0, ..., ID_i)$ for $1 \leq i \leq n-1$.
- Public-Key Oracle: On input $\mathcal{U}_n$'s identifying information $(ID_0, ..., ID_n)$, this oracle outputs the public key $P_n$ of user $\mathcal{U}_n$.
- Partial-Private-Key Oracle: On input $\mathcal{U}_n$'s identifying information $(ID_0, ..., ID_n)$, and the corresponding public key list $(P_0, ..., P_n)$, this oracle outputs the partial private key $D_n$ of $\mathcal{U}_n$.
- Secret-Value Oracle: On input $\mathcal{U}_n$'s identifying information $(ID_0, ..., ID_n)$, this oracle outputs the secret value $s_n$ of $\mathcal{U}_n$.
- Public-Key-Replacement Oracle: On input $(ID_0, ..., ID_n, P'_n)$, this oracle sets $P'_n$ as the new public key of the entity (either a user $\mathcal{U}_n$ or a KGC $\mathcal{K}_n$) whose identifying information is $(ID_0, ..., ID_n)$. Here $P'_n$ is an element chosen from the public key space by the adversary.
- Sign Oracle: On input a tuple $(ID_0, ..., ID_n, P_0, ..., P_n, m)$, this oracle outputs a valid signature $\sigma$ on message $m$, where $(ID_0, ..., ID_n)$ is the identifying information of a user $\mathcal{U}_n$ and $(P_0, ..., P_n)$ is the corresponding public key list chosen by the adversary.

At this phase, $\mathcal{A}_I$ can adaptively query the above Public-Key Oracle, Partial-Private-Key Oracle, Secret-Value Oracle, Public-Key-Replacement Oracle and Sign Oracle. However, before querying those oracles, we require that the public keys corresponding to the identities in the identifying information have been already generated. That is, the hierarchy of the lower-level KGCs has been fixed before the attacker queries the rest of oracles. This requirement is reasonable because, in practice, the hierarchy is formed before the users join the system or generate signatures.

*Phase I-3*: Finally, $\mathcal{A}_I$ outputs a tuple $(m^*, \sigma^*, ID_0^*, ..., ID_n^*, P_0^*, ..., P_n^*)$, in which $(ID_0^*, ..., ID_n^*)$ is the identifying information of the target user $\mathcal{U}_n^*$, $(P_0^*, ..., P_n^*)$ is the public key list chosen by $\mathcal{A}_I$, $ID_0^* = ID_0$ and $P_0^* = P_0$. We say that $\mathcal{A}_I$ wins Game I if the above tuple satisfies the following requirements: (1) $\sigma^*$ is a valid signature on $m^*$ under $(ID_0^*, ..., ID_n^*, P_0^*, ..., P_n^*)$; (2) $(ID_0^*, ..., ID_n^*, P_0^*, ..., P_n^*)$ has never been submitted to the Partial-Private-Key Oracle; (3) $(ID_0^*, ..., ID_n^*, P_0^*, .., P_n^*, m^*)$ has never been submitted to the Sign Oracle.

Game II (for *Type II Adversary*)

This game is played between a challenger $\mathcal{C}$ and a type II adversary $\mathcal{A}_{II}$. The game comprises three phases as follows.

*Phase II-1*: $\mathcal{C}$ runs Root-Setup to generate the *root* private key and the system parameter list $params$, then $\mathcal{C}$ sends $params$ and *root* private key to $\mathcal{A}_{II}$.

*Phase II-2*: Similarly to Game I, $\mathcal{A}_{II}$ can query the following oracles as defined in Game I: Lower-Level-Setup Oracle, Public-Key Oracle, Partial-Private-Key Oracle, Secret-Value Oracle, Public-Key-Replacement Oracle and Sign Oracle. Furthermore, we additionally allow $\mathcal{A}_{II}$ to query Reveal-KGC Oracle which is used to reveal the partial private keys of *lower-level* KGCs:

- Reveal-KGC Oracle: On input $\mathcal{K}_n$'s identifying information $(ID_0, ..., ID_n)$ and the corresponding public key list $(P_0, ..., P_n)$, this oracle returns the partial private key $D_n$ of $\mathcal{K}_n$.

*Phase II-3*: Finally, $\mathcal{A}_{II}$ outputs a tuple $(m^*, \sigma^*, ID_0^*, ..., ID_n^*, P_0^*, ..., P_n^*)$, in which $(ID_0^*, ..., ID_n^*)$ is the identifying information of the target user $\mathcal{U}_n^*$ and $(P_0^*, ..., P_n^*)$ is the corresponding public key list, $ID_0^* = ID_0$ and $P_0^* = P_0$. We say that $\mathcal{A}_{II}$ wins Game II if the above tuple satisfies the following requirements: (1) $\sigma^*$ is a valid signature on $m^*$ under $(ID_0^*, ..., ID_n^*, P_0^*, ..., P_n^*)$; (2) $P_n^*$ is the original public key of the target user $\mathcal{U}_n^*$. That is, $(ID_0^*, ..., ID_n^*, P_n')$ has never been submitted to the Public-Key-Replacement Oracle, where $P_n'$ denotes any public key except $P_n^*$; (3) $\mathcal{A}_{II}$ has never requested the secret value of the user possessing public key $P_n^*$; (4) $(ID_0^*, ..., ID_n^*, P_0^*, ..., P_n^*, m^*)$ has never been submitted to the Sign Oracle.

**Definition 1.** An HCLS scheme is existentially unforgeable under adaptive chosen-message attacks iff the probability of success of $\mathcal{A}_I$ in Game I and $\mathcal{A}_{II}$ in Game II is negligible.

## III. An HCLS Scheme

### A. Bilinear Maps

Let $\mathbb{G}_1$ be an additive group of prime order $q$ and $\mathbb{G}_2$ be a multiplicative group of the same order. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$ be a bilinear map with the following properties [14]: (1) Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1, a, b \in \mathbb{Z}_q^*$; (2) Non-degeneracy: There exists $P \in \mathbb{G}_1$ such that $e(P, P) \neq 1$; (3) Computability: There exists an efficient algorithm to compute $e(P, Q)$ for any $P, Q \in \mathbb{G}_1$.

**Computational Diffle-Hellman (CDH) Problem** in $\mathbb{G}_1$: Given a generator $P$ of $\mathbb{G}_1$ and $(aP, bP)$ for unknown $a, b \in \mathbb{Z}_q^*$, compute $abP$.

The CDH assumption states that any polynomial-time algorithm has only negligible success probability of solving the CDH problem.

### B. The Proposal

- Root-Setup: On input $1^\lambda$, the root KGC $\mathcal{K}_0$, chooses $\mathbb{G}_1, \mathbb{G}_2, e$ as defined in Section III-A, choose a generator $P \in \mathbb{G}_1$, a *root* private key $s_0 \in \mathbb{Z}_q^*$ and sets $P_0 = s_0 P$ as the public key of $\mathcal{K}_0$, chooses hash functions $H_1, ..., H_4 : \{0,1\}^* \longrightarrow \mathbb{G}_1$. The system parameters are $params = (\mathbb{G}_1, \mathbb{G}_2, e, P, P_0, H_1, ..., H_4, ID_0)$, where $ID_0$ is the identity of $\mathcal{K}_0$.
- Lower-Level-Setup: For $n \geq 1$, let $\mathcal{K}_n$'s identifying information be $(ID_0, ..., ID_n)$ and $P_i, 1 \leq i \leq n-1$ be the public key of $\mathcal{K}_i$. When $\mathcal{K}_n$ joins the system, it runs this protocol with its parent $\mathcal{K}_{n-1}$.
  1) $\mathcal{K}_n$ chooses a random secret value $s_n \in \mathbb{Z}_q^*$, sets $P_n = s_n P$ as its public key, computes $Q_n = H_1(ID_0, ..., ID_n, P_0, ..., P_n)$ and sends $(ID_n, P_n)$ to $\mathcal{K}_{n-1}$.
  2) Let $\mathcal{K}_{n-1}$ possess private key $(s_{n-1}, D_{n-1})$. When receiving $(ID_n, P_n)$ from $\mathcal{K}_n$, $\mathcal{K}_{n-1}$ computes $Q_n = H_1(ID_0, ..., ID_n, P_0, ..., P_n)$ and $D_n = D_{n-1} + s_{n-1}Q_n$, sends $D_n$ to $\mathcal{K}_n$ through a secure channel. Here, for consistency, we define $D_0 = 0$.
  3) $\mathcal{K}_n$ sets its private key as $(s_n, D_n)$.
- Set-Secret-Value: On input $\mathcal{U}_n$'s identifying information $(ID_0, ..., ID_n)$ and the corresponding public key list $(P_0, ..., P_{n-1})$, $\mathcal{U}_n$ randomly selects $s_n \in \mathbb{Z}_q^*$ and outputs $s_n$ as his secret value.
- Set-Public-Key: On input $\mathcal{U}_n$'s secret value $s_n$, identifying information $(ID_0, ..., ID_n)$ and the corresponding public key list $(P_0, ..., P_{n-1})$, $\mathcal{U}_n$ outputs $P_n = s_n P$ as his public key.
- Partial-Private-Key-Extract: On input $\mathcal{U}_n$'s identifying information $(ID_0, ..., ID_n)$ and the corresponding public key list $(P_0, ..., P_n)$, $\mathcal{K}_{n-1}$ computes the partial private key for $\mathcal{U}_n$ as follows:
  1) Compute $Q_n = H_1(ID_0, ..., ID_n, P_0, ..., P_n)$.
  2) Select a random $x' \in \mathbb{Z}_q^*$ and compute $R' = x'P$.
  3) Compute $E = H_2(ID_0, ..., ID_n, P_0, ..., P_n)$.
  4) Compute $D_n' = D_{n-1} + s_{n-1}Q_n + x'E$.
  5) Output the partial private key $D_n = (R', D_n')$.
- Set-Private-Key: On input $\mathcal{U}_n$'s secret value $s_n$, partial private key $D_n$, identifying information $(ID_0, ..., ID_n)$ and the public key list $(P_0, ..., P_n)$, $\mathcal{U}_n$ sets $S_n = (s_n, D_n)$ as his private key.
- Sign: Let the signer $\mathcal{U}_n$ have identifying information $(ID_0, ..., ID_n)$, public keys $(P_0, ..., P_n)$ and private key $S_n = (s_n, D_n)$, where $D_n = (R', D_n')$. To sign a message $m \in \mathcal{M}$, $\mathcal{U}_n$ uses the private key $S_n$ and performs the following steps of which the first two can be pre-computed:
  1) Compute $E = H_2(ID_0, ..., ID_n, P_0, ..., P_n)$ and $Q_i = H_1(ID_0, ..., ID_i, P_0, ..., P_i)$ for $1 \leq i \leq n$.
  2) Compute $R = R' + xP, W = D_n' + xE$, $U = yP$ for randomly chosen values $x, y \in \mathbb{Z}_q^*$.
  3) Compute $F = H_3(ID_0, ..., ID_n, P_0, ..., P_n, m)$, $T = H_4(ID_0, ..., ID_n, P_0, ..., P_n, m)$.
  4) Compute $V = W + s_n F + yT$.
  5) Output $\sigma = \{R, U, V\}$ as the signature on $m$.
- Verify: To verify a signature $\sigma = \{R, U, V\}$ on message $m$ under $(ID_0, ..., ID_n, P_0, ..., P_n)$[1], the verifier performs the following steps:
  1) Compute $E = H_2(ID_0, ..., ID_n, P_0, ..., P_n)$, $F = H_3(ID_0, ..., ID_n, P_0, ..., P_n, m), T = H_4(ID_0, ..., ID_n, P_0, ..., P_n, m)$ and $Q_i = H_1(ID_0, ..., ID_i, P_0, ..., P_i)$ for $1 \leq i \leq n$.
  2) Check whether $e(V, P) \stackrel{?}{=} e(R, E)e(U, T)e(P_n, F)\prod_{i=1}^{n} e(P_{i-1}, Q_i)$ holds with equality. Output *true* if the equality holds; otherwise, output *false*.

[1]We notice that the identifying information and the corresponding public key list may need to be provided to a signature verifier along with the signature itself. However, the signature itself consists of only three group elements.

## IV. Security Analysis

**Theorem 1.** *If there exists a Type I adversary who has an advantage $\epsilon$ in forging a signature of our HCLS scheme in an attack modeled by the Game I of Section II-B, within a time span $\tau$; and who can make at most $q_{H_1}$ (resp. $q_{H_2}, q_{H_3}, q_{H_4}, q_L, q_{PK}, q_{PPK}$, $q_{SK}, q_{PKR}$ and $q_S$) times $H_1$ (resp. $H_2, H_3, H_4$,* Lower-Level-Setup Oracle*,* Public-Key Oracle*,* Partial-Private-Key Oracle*,* Secret-Key Oracle*,* Public-Key-Replacement Oracle *and* Sign Oracle*) queries, then the CDH problem in $\mathbb{G}_1$ can be solved within time $\tau + \mathcal{O}(q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_L + q_{PK} + \max\{n\}q_{PPK} + \max\{n\}q_S)\tau_{\mathbb{G}_1}$ and with probability $\epsilon' \geq (1 - \frac{3}{q_{PPK}+q_S+3})^{q_{PPK}+q_S}(\frac{3}{q_{PPK}+q_S+3})^3\epsilon$, where $\tau_{\mathbb{G}_1}$ is the time to compute a scalar multiplication in $\mathbb{G}_1$.*

Due to the page limitation, the proof will be presented in the full version of this paper.

**Theorem 2.** *If there exists a Type II adversary who has an advantage $\epsilon$ in forging a signature of our HCLS scheme in an attack modeled by the Game II of Section II-B within a time span $\tau$, then the CDH problem in $\mathbb{G}_1$ can be solved within time $\tau + \mathcal{O}(q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_L + q_{PK} + \max\{n\}(q_{PPK} + q_{RK} + q_S))\tau_{\mathbb{G}_1}$ and with probability $\epsilon' \geq (1 - \frac{2}{q_{SK}+q_{PKR}+q_S+2})^{q_{SK}+q_{PKR}+q_S}(\frac{2}{q_{SK}+q_{PKR}+q_S+2})^2\epsilon$.*

*Proof.* Let $\mathcal{C}$ be a CDH challenger who is given an instance $(P, aP, bP)$ of the CDH problem in $\mathbb{G}_1$ and wants to compute $abP$. Let $\mathcal{A}_{II}$ be a forger who can break the proposed HCLS scheme under an adaptive chosen-message attack. We show how $\mathcal{C}$ can use $\mathcal{A}_{II}$ to solve the CDH problem.

*Phase II-1*: Firstly, $\mathcal{C}$ selects $s_0 \in \mathbb{Z}_q^*$, sets $P_0 = s_0P$, then selects the system parameters $\mathsf{params} = (\mathbb{G}_1, \mathbb{G}_2, e, P, P_0, H_1, H_2, H_3, H_4, ID_0)$, and then gives $s_0$ and $params$ to $\mathcal{A}_{II}$. In the sequel, we treat $H_1, H_2, H_3$ and $H_4$ as random oracles which are controlled by $\mathcal{C}$.

*Phase II-2*: $\mathcal{C}$ simulates the random oracles as well as the oracles defined in our Game II as follows:

$H_1$ Oracle: $\mathcal{C}$ maintains an initially empty list $H_1^{list}$. On input $(ID_0, ..., ID_n, P_0, ..., P_n)$, the same answer from $H_1^{list}$ will be given if the request has been asked before. Otherwise, $\mathcal{C}$ picks a random $\alpha_n \in \mathbb{Z}_q^*$, computes $Q_n = \alpha_nP$, returns $Q_n$ as the answer and adds $(ID_0, ..., ID_n, P_0, ..., P_n, \alpha_n, Q_n)$ to $H_1^{list}$.

$H_2$ Oracle: $\mathcal{C}$ keeps an initially empty list $H_2^{list}$. On input $(ID_0, ..., ID_n, P_0, ..., P_n)$, the same answer from $H_2^{list}$ will be given if the request has been asked before. Otherwise, $\mathcal{C}$ randomly selects $\beta \in \mathbb{Z}_q^*$, sets $E = \beta P$, returns $E$ as the answer and adds $(ID_0, ..., ID_n, P_0, ..., P_n, \beta, E)$ to $H_2^{list}$.

$H_3$ Oracle: $\mathcal{C}$ keeps an initially empty list $H_3^{list}$. On input $(ID_0, ..., ID_n, P_0, ..., P_n, m)$, the same answer from $H_3^{list}$ will be given if the request has been asked before. Otherwise, $\mathcal{C}$ randomly selects $\gamma \in \mathbb{Z}_q^*$, flips a coin $coin_{H_3} \in \{0, 1\}$ that yields 1 with probability $\delta$ and 0 with probability $1 - \delta$. If $coin_{H_3} = 0$, $\mathcal{C}$ computes $F = \gamma P$; else sets $F = \gamma aP$. Finally, $\mathcal{C}$ adds $(ID_0, ...ID_n, P_0, ..., P_n, m, \pi, F, coin_{H_3})$ to $H_3^{list}$ and returns $F$ as the answer.

$H_4$ Oracle: $\mathcal{C}$ keeps an initially empty list $H_4^{list}$. On input $(ID_0, ..., ID_n, P_0, ..., P_n, m)$, the same answer from $H_4^{list}$ will be given if the request has been asked before. Otherwise, $\mathcal{C}$ randomly selects $\pi \in \mathbb{Z}_q^*$ and computes $T = \pi P$. Finally, $\mathcal{C}$ adds $(ID_0, ..., ID_n, P_0, ..., P_n, m, \pi, T)$ to $H_4^{list}$ and returns $T$ as the answer.

Lower-Level-Setup Oracle: $\mathcal{C}$ keeps an initially empty list $KGC^{list}$. On input a KGC's identifying information $(ID_0, ..., ID_n)$, $\mathcal{C}$ does the following:

- If $(ID_0, ..., ID_n)$ already appears in $KGC^{list}$ in a tuple $(ID_0, ..., ID_n, ...; s_0, ..., s_n, ...; P_0, ..., P_n, ...)$, respond with $(s_n, P_n)$.
- Else if $n = 1$, select $s_1 \in \mathbb{Z}_q^*$, compute $P_1 = s_1P$, add $(ID_0, ID_1; s_0, s_1; P_0, P_1)$ to $KGC^{list}$ and return $(s_1, P_1)$ as the answer.
- Else, recover $(ID_0, ..., ID_{n-1}; s_0, ..., s_{n-1}; P_0, ..., P_{n-1})$ from $KGC^{list}$, then randomly select $s_n \in \mathbb{Z}_q^*$, compute $P_n = s_nP$, add $(ID_0, ..., ID_n; s_0, ..., s_n; P_0, ..., P_n)$ to $KGC^{list}$, remove $(ID_0, ..., ID_{n-1}; s_0, ..., s_{n-1}; P_0, ..., P_{n-1})$ from $KGC^{list}$ and respond with $(s_n, P_n)$.

Public-Key Oracle: $\mathcal{C}$ keeps an initially empty list $User^{list}$. On input $(ID_0, ..., ID_n)$, $\mathcal{C}$ does the following:

- If $(ID_0, ..., ID_n, P_n, s_n, coin_{PK})$ is in $User^{list}$, return $P_n$ as the answer.
- Else flip a coin $coin_{PK} \in \{0, 1\}$ that yields 1 with probability $\delta$ and 0 with probability $1-\delta$, randomly select $s_n \in \mathbb{Z}_q^*$ and do the following:
  - If $coin_{PK} = 0$, compute $P_n = s_nP$ and add $(ID_0, ..., ID_n, P_n, s_n, coin_{PK})$ to $User^{list}$ and return $P_n$ as the answer.
  - Else, set $P_n = s_nbP$, return $P_n$ as the answer and add $(ID_0, ..., ID_n, P_n, s_n, coin_{PK})$ to $User^{list}$.

Partial-Private-Key Oracle: On input $(ID_0, ..., ID_n, P_0, ..., P_n)$, $\mathcal{C}$ does the following:

1) For $1 \leq i \leq n$, submit $(ID_0, ..., ID_i, P_0, ..., P_i)$ to the $H_1$ Oracle to generate the tuple $(ID_0, ...ID_i, P_0, ..., P_i, \alpha_i, Q_i)$.
2) Submit $(ID_0, ..., ID_n, P_0, ..., P_n)$ to the $H_2$ Oracle and find the tuple $(ID_0, ..., ID_n, P_0, ..., P_n, \beta, E)$ in $H_2^{list}$.
3) Randomly select $x \in \mathbb{Z}_q^*$, set $R_n = xP$, $D_n = \sum_{i=1}^{n} \alpha_iP_{i-1} + xE$, and return $(R_n, D_n)$ as the answer.

Secret-Value Oracle: On input $(ID_0, ..., ID_n)$, $\mathcal{C}$ first submits $(ID_0, ..., ID_n)$ to the Public-Key Oracle then recovers the tuple $(ID_0, ..., ID_n, P_n, s_n, coin_{PK})$ from $User^{list}$. If $coin_{PK} = 0$, $\mathcal{C}$ returns $s_n$ as the answer; else $\mathcal{C}$ aborts.

Public-Key-Replacement Oracle: On input $(ID_0, ..., ID_n, P'_n)$, this oracle does the following:

- If $ID_n$ is the identity of a user, submit $(ID_0, ..., ID_n)$ to the Public-Key Oracle, recover $(ID_0, ..., ID_n, P_n, s_n, coin_{PK})$ from $User^{list}$. If $coin_{PK} = 1$ and $P'_n \neq s_nP$, abort; otherwise, set $P_n = P'_n$.

- Else $ID_n$ is the identity of a KGC, submit $(ID_0, ..., ID_n)$ to the Lower-Level-Setup Oracle, recover from $KGC^{list}$ all the tuples $(ID_0, ..., ID_n, ...; s_0, ..., s_n, ...; P_0, ..., P_n, ...)$ which contain $ID_0, ..., ID_n$. Set $P_n = P'_n$ in all the tuples.

Reveal-KGC Oracle: On input $(ID_0, ..., ID_n, P_0, ..., P_n)$, $\mathcal{C}$ first submits $(ID_0, ..., ID_i, P_0, ..., P_i)$ to the $H_1$ Oracle and recovers $(ID_0, ..., ID_i, P_0, ..., P_i, \alpha_i, Q_i)$ from $H_1^{list}$ for $1 \leq i \leq n$, then computes and outputs $D_n = \sum_{i=1}^{n} \alpha_i P_{i-1}$.

Sign Oracle: On input $(ID_0, ..., ID_n, P_0, ..., P_n, m)$, $\mathcal{C}$ first does the following:

1) For $1 \leq i \leq n$, submit $(ID_0, ..., ID_i, P_0, ..., P_i)$ to the $H_1$ Oracle and recover $(ID_0, ..., ID_i, P_0, ..., P_i, \alpha_i, Q_i)$ from $H_1^{list}$.
2) Submit $(ID_0, ..., ID_n, P_0, ..., P_n)$ to the $H_2$ Oracle and recover $(ID_0, ..., ID_n, P_0, ..., P_n, \beta, E)$ from $H_2^{list}$.
3) Submit $(ID_0, ..., ID_n, P_0, ..., P_n, m)$ to both $H_3$ Oracle and $H_4$ Oracle, and recover $(ID_0, ..., ID_n, P_0, ..., P_n, m, \gamma, F, coin_{H_3})$ from $H_3^{list}$, $(ID_0, ..., ID_n, P_0, ..., P_n, m, \pi, T)$ from $H_4^{list}$.

Then, if $coin_{H_3} = 0$, $\mathcal{C}$ randomly selects $R, U \in \mathbb{G}_1$, computes $V = \beta R + \pi U + \gamma P_n + \sum_{i=1}^{n} \alpha_i P_{i-1}$ and outputs the signature $(R, U, V)$; otherwise, it aborts.

*Phase II-3*: In this phase, $\mathcal{A}_{II}$ outputs a tuple $(m^*, \sigma^*, ID_0^*, ..., ID_n^*, P_0^*, ..., P_n^*)$, where $\sigma^* = (R^*, U^*, V^*)$. It requires that $\sigma^*$ is a valid signature on $m^*$ under $(ID_0^*, ..., ID_n^*, P_0^*, ..., P_n^*)$.

$\mathcal{C}$ aborts if any of following events is not satisfied:

1) *Event 3*: For $1 \leq i \leq n$, $(ID_0^*, ..., ID_i^*, P_0^*, ..., P_i^*, \alpha_i^*, Q_i^*)$ is in $H_1^{list}$, $(ID_0^*, ..., ID_n^*, P_0^*, ..., P_n^*, \beta^*, E^*)$ is in $H_2^{list}$, $(ID_0^*, ..., ID_n^*, P_0^*, ..., P_n^*, m^*, \gamma^*, F^*, coin_{H_3}^*)$ is in $H_3^{list}$, and $(ID_0^*, ..., ID_n^*, P_0^*, ..., P_n^*, m^*, \pi^*, T^*)$ is in $H_4^{list}$.
2) *Event 4*: $coin_{H_3}^* = coin_{PK}^* = 1$, where $coin_{PK}^*$ is in the tuple $(ID_0^*, ..., ID_n^*, P_0^*, ..., P_n^*, m^*, \gamma^*, F^*, coin_{H_3}^*)$ in $User^{list}$.

If $\mathcal{C}$ does not abort, we have $E^* = \beta^* P, F^* = \gamma^* aP, T^* = \pi^* P$ and for $1 \leq i \leq n$, $Q_i^* = \alpha_i^* P$, $P_n^* = s_n^* bP$. Since $e(V^*, P) = e(R^*, E^*) e(U^*, T^*) e(P_n^*, F^*) \prod_{i=1}^{n} e(P_{i-1}^*, Q_i^*)$, $\mathcal{C}$ can output $abP = (s_n^* \gamma^*)^{-1} (V^* - (\beta^* R^* + \pi^* U^* + \sum_{i=1}^{n} \alpha_i^* P_{i-1}^*))$ as the solution of the CDH problem.

We further show that $\mathcal{C}$ solves the given instance of the CDH problem with probability at least $\epsilon'$. First, we analyze the three events needed for $\mathcal{C}$ to succeed: (1) $\mathcal{E}1$: $\mathcal{C}$ does not abort as a result of any of $\mathcal{A}_{II}$'s Secret-Key Oracle, Public-Key-Replacement Oracle and Sign Oracle queries; (2) $\mathcal{E}2$: $\sigma^*$ is a valid signature on $m^*$ under $(ID_0^*, ..., ID_n^*, P_0^*, ..., P_n^*)$; (3) $\mathcal{E}3$: Both *Event 3* and *Event 4* happen.

$\mathcal{C}$ succeeds if all of these events happen. Hence, we have that $\epsilon' = \Pr[\mathcal{E}1 \wedge \mathcal{E}2 \wedge \mathcal{E}3] = \Pr[\mathcal{E}1] \Pr[\mathcal{E}2|\mathcal{E}1] \Pr[\mathcal{E}3|\mathcal{E}1 \wedge \mathcal{E}2]$. We have $\epsilon' = \Pr[\mathcal{E}1 \wedge \mathcal{E}2 \wedge \mathcal{E}3] \geq (1-\delta)^{q_{SK}+q_{PKR}+q_S} \delta^2 \epsilon$. When $\delta = \frac{2}{q_{SK}+q_{PKR}+q_S+2}$, we have $\epsilon' \geq (1 - \frac{2}{q_{SK}+q_{PKR}+q_S+2})^{q_{SK}+q_{PKR}+q_S} (\frac{2}{q_{SK}+q_{PKR}+q_S+2})^2 \epsilon$. □

## V. Conclusion

We modeled HCLS systems and presented a concrete fully scalable HCLS scheme. We illustrated that our scheme is secure under the CDH assumption.

*Acknowledgments and disclaimer*

This work is partly supported by the Spanish Government under projects TSI2007-65406-C03-01 "E-AEGIS", TIN2009-11689 "RIPUP", "eVerification" TSI-020100-2009-720, SeCloud TSI-020302-2010-153 and CONSOLIDER INGENIO 2010 CSD2007-00004 "ARES", by the Nature Science Foundation of China through projects 60970114, 60970115, 60970116 and 61003214, and by the Fundamental Research Funds for the Central Universities of China through project 3103004. The third author is partially supported as an ICREA-Acadèmia researcher by the Catalan Government. The authors are with the UNESCO Chair in Data Privacy, but this paper does not necessarily reflect the position of UNESCO nor does it commit that organization.

## References

[1] S. Al-Riyami and K. Paterson, Certificateless public key cryptography. ASIACRYPT 2003, LNCS 2894, pp. 452-473, 2003.
[2] D. Boneh, X. Boyen, and E. Goh, Hierarchical identity based encryption with constant size ciphertext. EUROCRYPT 2005, LNCS 3494, pp. 440-456, 2005.
[3] X. Boyen and B. Waters, Anonymous hierarchical identity-based encryption (without random oracles). CRYPTO 2006, LNCS 4117, pp. 290-307, 2006.
[4] R. Canetti, S. Halevi, and J. Katz, A forward-secure public-key encryption scheme. EUROCRYPT 2003, LNCS 2656, pp. 255-271, 2003.
[5] C. Gentry and A. Silverberg, Hierarchical ID-based cryptography. ASIACRYPT 2002, LNCS 2501, pp. 548-566, 2002.
[6] J. Horwitz and B. Lynn, Towards hierarchical identity-based encryption. EUROCRYPT 2002, LNCS 2332, pp. 466-481, 2002.
[7] B. Hu, D. Wong, Z. Zhang and X. Deng, Key replacement attack against a generic construction of certificateless signature. ACISP 2006, LNCS 4058, pp. 235-346, 2006.
[8] X. Huang, Y. Mu, W. Susilo, D. Wong, and W. Wu, Certificateless signature revisited. ACISP 2007, LNCS 4586, pp. 308-322, 2007.
[9] X. Huang, W. Susilo, Y. Mu and F. Zhang, On the security of a certificateless signature scheme. CANS 2005, LNCS 3810, pp. 13-25, 2005.
[10] J. Lane, P. Heus and T. Mulcahy, Data access in a cyberworld: making use of cyberinfrastructure. Transactions on Data Privacy, 1(1): 2-16, 2008.
[11] H. Lim and K. Paterson, Multi-key hierarchical identity-based signatures. Cryptography and Coding 2007, LNCS 4887, pp. 384-402, 2007.
[12] Y. Ren and D. Gu, Secure hierarchical identity based encryption scheme in the standard model. INDOCRYPT 2008, LNCS 5365, pp. 104-115, 2008.
[13] Q. Wu, J. Domingo-Ferrer and Ú. González-Nicolás, Balanced Trustworthiness, Safety and Privacy in Vehicle-to-Vehicle Communications. IEEE Transactions on Vehicular Technology, 59(2): 559-573, 2010.
[14] Q. Wu, Y. Mu, W. Susilo, B. Qin and J. Domingo-Ferrer. Asymmetric Group Key Agreement. Eurocrypt 2009, LNCS 5479, pp. 153-170. Springer-Verlag, 2009.
[15] Z. Zhang, D. Wong, J. Xu and D. Feng, Certificateless public-key signature: security model and efficient construction. ACNS 2006, LNCS 3989, pp. 293-308, 2006.
[16] L. Zhang, Q. Wu, A. Solanas and J. Domingo-Ferrer, A Scalable Robust Authentication Protocol for Secure Vehicular Communications. IEEE Transactions on Vehicular Technology, 59(4), 1606-1617, 2010.
[17] L. Zhang, F. Zhang, Q. Wu and J. Domingo-Ferrer, Simulatable certificateless two-party authenticated key agreement protocol. Information Sciences, 180(6): 1020-1030, 2010.