Hierarchical Identity-Based Chameleon Hash and Its Applications

Feng Bao[1], Robert H. Deng[2], Xuhua Ding[2], Junzuo Lai[2,⋆], and Yunlei Zhao[3]

[1] Institute for Infocomm Research, Singapore
baofeng@i2r.a-star.edu.sg
[2] Singapore Management University, Singapore
{robertdeng,xhding,junzuolai}@smu.edu.sg
[3] Fudan University, China
ylzhao@fudan.edu.cn

Abstract. At ACNS 2008, Canard et al. introduced the notion of trapdoor sanitizable signature (TSS) based on identity-based chameleon hash (IBCH). Trapdoor sanitizable signatures allow the signer of a message to delegate, at any time, the power of sanitization to possibly several entities who can modify predetermined parts of the message and generate a new signature on the sanitized message without interacting with the original signer. In this paper, we introduce the notion of hierarchical identity-based chameleon hash (HIBCH), which is a hierarchical extension of IBCH. We show that HIBCH can be used to construct other cryptographic primitives, including hierarchical trapdoor sanitizable signature (HTSS) and key-exposure free IBCH. HTSS allows an entity who has the sanitization power for a given signed message, to further delegate its power to its descendants in a controlled manner. Finally, we propose a concrete construction of HIBCH and show that it is t-threshold collusion-resistant.

Keywords: Chameleon Hash, Trapdoor Sanitizable Signature, Hierarchical Identity-Based Chameleon Hash, Hierarchical Trapdoor Sanitizable Signature.

1 Introduction

Chameleon hash was introduced by Krawczyk and Rabin [20] as a tool to construct chameleon signatures. Informally, a chameleon hash function is a trapdoor collision-resistant hash function: without knowledge of the trapdoor, the chameleon hash function is collision-resistant; however, collisions can be easily computed once the trapdoor is known. Similar to undeniable signatures [9], chameleon signatures possess the properties of non-repudiation and *non-transferability* for the signed messages; however, chameleon signatures are non-interactive protocols. In order to provide a recipient with a non-transferable signature, a signer hashes the message to be signed with a recipient's chameleon

⋆ Corresponding author.

J. Lopez and G. Tsudik (Eds.): ACNS 2011, LNCS 6715, pp. 201–219, 2011.

hash function and signs on the resulting digest value. The recipient knows the trapdoor of the chameleon hash function and hence is able to re-use the hash value to obtain a signature on a second message. On the other hand, the signer can prove knowledge of a hash collision, since the original signed message and the claimed signed message have the same hash value. Such a collision can be seen as proof of forgery by the signature recipient, as nobody apart from the recipient has more than a negligible probability of successfully finding a collision. One limitation of the original chameleon signature scheme [20] is that signature forgery results in the signer discovering the recipient's trapdoor information. This deterrent effect of key/trapdoor exposure on forgeries threatens the claims of non-transferability provided by the scheme. In fact, a third party will likely believe claims made by the recipient, because the potential devastating damage to the recipient would result from the forgery of a signature.

Identity-based chameleon hash (IBCH) and identity-based chameleon signature (IBCS), introduced by Ateniese and Medeiros [2], partly addressed the problem of key exposure. In IBCH/IBCS, a unique transaction-specific public key, called *customized identity*, is used to compute the chameleon hash of a transaction. The customized identity is computed by the signer from special strings that describe the transaction, including the signer and recipient information as well as a nonce value or time-stamp. As a result, the trapdoor corresponding to the customized identity is transaction specific and signature forgery only results in the signer recovering the trapdoor information associated with a single transaction.

Based on chameleon hash, Ateniese et al. [1] introduced the notion of sanitizable signature and presented its generic construction. Sanitizable signatures allow a signer to partly delegate signing rights to a semi-trusted party, called a sanitizer. During generation of a signature on a message, the signer chooses a specific sanitizer who can later modify predetermined parts of the message and generate a new signature on the sanitized message without interacting with the signer. The capability of modification renders sanitizable signatures valuable for many applications, such as authenticated multicast, authenticated database outsourcing and secure routing.

At ACNS 2008, Canard et al. [8] introduced the notion of trapdoor sanitizable signature (TSS) and showed its generic construction based on IBCH. TSS allows the signer to delegate the power of sanitization for a specific signed message to possibly several entities. Different from the sanitizable signatures in [1] where the sanitizer is predetermined at the time of signature generation by the signer, the signer in TSS can choose to whom and when it will provide the trapdoor information and therefore, any entity can potentially act as a sanitizer. This property makes a crucial difference from the conventional sanitizer signatures and is essential for applications where the potential sanitizers are not known at the time of signature generation.

In this paper, we introduce the notions of hierarchical identity-based chameleon hash (HIBCH) and hierarchical trapdoor sanitizable signature (HTSS), which are the hierarchical extensions of IBCH and TSS, respectively. We present a generic

construction of HTSS from HIBCH. Like TSS, HTSS allows a signer to delegate the power of sanitization for a specific signed message to any sanitizers at any time. In addition, HTSS allows a sanitizer to further delegate sanitization power to its descendants in an identity hierarchy. This distinguishing feature of cascaded delegation of sanitization powers makes HTSS especially powerful in protecting information flows in distributed settings, such as automated web-service-enabled business processes [26] and tiered multimedia distribution systems [25].

As an example of automated web-service-enabled business processes, let us consider a simple quotation response process, involving an electronic distributor (ED), a transportation company (TC) and an electronic manufacturer (EM). A business document *Quotation* in XML format is transferred between various entities with use of document-styled web services. The quotation response process begins when ED receives a request for quotation from an electronic retailer (ER). ED generates the *Quotation* by providing quotes for each item and the taxes associated and forwards the *Quotation* document to TC via a SOAP message. Upon receipt of the document, TC adds the delivery cost, delivery information and updates the total cost. TC then forwards the document to EM, which inputs additional product information based on the retailer's information and then forwards the *Quotation* to ER. Upon receipt of the document, ER informs ED that the quotation has been received. A basic security requirement of the quotation response process is integrity and authenticity of the *Quotation* document. Such a requirement can be fulfilled readily using HTSS: the document originator ED generates a signature and a trapdoor on the *Quotation* document and forwards them to TC. With the knowledge of the trapdoor, TC is able to perform predetermined modification on the document, such as adding delivery cost, without invaliding the original signature. TC then generates a new trapdoor and forwards the updated document and the trapdoor to EM, which in turn modifies the document based on its input.

Another application of HTSS is end-to-end content authentication in tiered multimedia distribution systems, where multimedia contents are distributed from a top-tier primary content provider to multiple levels of lower-tier affiliating providers each with its own user groups. An example is a multinational company that has a global headquarter, a number of regional headquarters and many country level offices worldwide. To promote a new product, the company produces a video advertisement for the product and delivers it to all the regional headquarters for processing, which then disseminate the processed video clips to the country level offices. In order to better fit local markets, regional headquarters and country level offices are entitled to derive their own local versions (e. g., adding subtitles in the local language) based on the original advertisement. In scenarios like this, higher-tier content providers may authorize lower-tier content providers performing transcoding operations on the original content, such as content downscaling, content alteration, and content insertion. Apparently, the capability of cascaded delegation of sanitization powers makes HTSS an ideal solution for authenticated content delivery in such environments.

1.1 Our Contributions

In this paper, we make the following contributions:

1. We introduce the notion of HIBCH, which is a hierarchical extension of IBCH. We also introduce the security definitions of HIBCH.
2. We introduce the notion of HTSS, which is a hierarchical extension of TSS. In an HTSS scheme, a signer can delegate at any time the power of sanitization for a specific signed message to an entity; the entity in turn can further delegate its power to its descendants *in a controlled manner* (details are given in Section 4.1). We extend the standard security definitions of TSS for the hierarchical setting and propose a generic construction of HTSS based on HIBCH.
3. We show that a key-exposure free IBCH can be obtained from a two-level HIBCH, though the latter is not key-exposure free. The construction of key-exposure free IBCH from HIBCH with key-exposure is similar to the construction of key-exposure free chameleon hash from IBCH [2,3]. In our construction, forgery only results in recovering the trapdoor information associated to a specific transaction, and therefore offering a partial answer to the key exposure problem of IBCH. We also point out in the full version of the paper a flaw in [11] which was the first full construction of a key-exposure free IBCH.
4. We present a concrete construction of HIBCH, which is resilient against compromise of a threshold number of entities in every level of the underlying hierarchy.

1.2 Related Work

Chameleon hash was introduced by Krawczyk and Rabin [20]. The original construction of chameleon hash [20] suffers from the key exposure problem. The problem was partly addressed by IBCH, which was introduced by Ateniese and Medeiros [2].

Chen et al. [10] presented the first full construction of a key-exposure free chameleon hash, which works in the setting of gap groups with bilinear pairings. Ateniese and Medeiros [3] proposed three key-exposure free chameleon hash functions, two based on RSA and one based on pairings. Other key-exposure free chameleon hash constructions [14,13,12] were proposed subsequently.

Zhang et al. [28] proposed two IBCH schemes from bilinear pairing. Recently, Chen et al. [11] considered the key exposure problem of IBCH and proposed a concrete construction of key-exposure free IBCH. However, in the Appendix, we point out a fault of the construction in [11].

The notion of sanitizable signature was introduced by Ateniese et al. [1]. Santizable signature allows a sanitizer to modify predetermined parts of a signed message and generate new signature on the sanitized message without interacting with the signer. Klonowski and Lauks [19] presented several extensions of sanitizable signature, including limitation of the set of possible modifications of

a single mutable block and limitation of the number of modifications of mutable blocks.

Ateniese et al. [1] identified five security requirements of sanitizable signature schemes, including *unforgeability*, *immutability*, *privacy*, *transparency* and *accountability*. Recently, Brzuska et al. [7] revisited the security requirements for sanitizable signatures and investigated the relationship of the security requirements, showing for example that transparency implies privacy.

Miyazaki et al. [23] also used the notion of sanitizable signature in a slightly different vein. Such sanitizable signature schemes [23,17,22] allow the sanitizer to only *delete* predetermined parts of a signed message.

The notions of incremental cryptography [4] and homomorphic signatures, which encompass transitive [21], redactable [18] and context-extraction signatures [24], are also related to sanitizable signatures. We refer the reader to [1] for details.

Canard et al. [8] introduced the notion of trapdoor sanitizable signatures (TSS), in which the power of sanitization is given to possibly several entities. Based on IBCH, Canard et al. [8] proposed a generic construction of TSS. Recently, Yum et al. [27] presented a generic construction of trapdoor sanitizable signatures from ordinary signature schemes; therefore, one-way functions imply trapdoor sanitizable signatures.

1.3 Organization

The rest of the paper is organized as follows. Some preliminaries are given in Section 2. We introduce the notion and security requirements of HIBCH in Section 3. We introduce the notion of HTSS and propose a generic construction of HTSS from HIBCH in Section 4. In Section 5, we describe the generic construction of key-exposure free IBCH from HIBCH with key exposure. We describe and analysis our concrete HIBCH scheme in Section 6. Finally, we state our conclusion in Section 7.

2 Preliminaries

If L is a positive integer, then $[1, L] = \{1, 2, \ldots, L\}$. If S_1, S_2 are two sets, $S_1 \backslash S_2 = \{x \in S_1 | x \notin S_2\}$. Let $\mathbb{Z}_p$ denote the set $\{0, 1, 2, \ldots, p-1\}$ and $\mathbb{Z}_p^*$ denote $\mathbb{Z}_p \backslash \{0\}$. For a finite set S, $x \xleftarrow{\$} S$ means choosing an element $x \in S$ with a uniform distribution. If $x_1, x_2, \ldots$ are strings, then $x_1 \| x_2 \| \ldots$ denotes their concatenation. If A is a probabilistic algorithm, then $A(x, r)$ is the result of running A on input x and coins r. We denote by $A(x; \mathcal{R})$ the random variable of choosing coins r uniformly at random from $\mathcal{R}$ and outputting $A(x, r)$.

We say that a function $f(\lambda)$ is *negligible* if for every $c > 0$ there exists an λ_c such that $f(\lambda) < 1/\lambda^c$ for all $\lambda > \lambda_c$. We say that two distribution ensembles $\{X(\lambda, z)\}_{\lambda \in N, z \in \{0,1\}^*}$ and $\{Y(\lambda, z)\}_{\lambda \in N, z \in \{0,1\}^*}$ are *computationally indistinguishable*, if for any probabilistic polynomial-time (PPT) algorithm D, and for all sufficiently large λ and any $z \in \{0,1\}^*$, it holds that $|\Pr[D(\lambda, z, X) = 1] - \Pr[D(\lambda, z, Y) = 1]|$ is negligible in λ.

2.1 Bilinear Pairings

Let $\mathbb{G}$ be a cyclic multiplicative group of prime order p and $\mathbb{G}_T$ be a cyclic multiplicative group of the same order p. A bilinear pairing is a map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties:

- Bilinearity: $\forall g_1, g_2 \in \mathbb{G}, \forall a, b \in \mathbb{Z}_p^*$, we have $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$;
- Non-degeneracy: There exist $g_1, g_2 \in \mathbb{G}$ such that $e(g_1, g_2) \neq 1$;
- Computability: There exists an efficient algorithm to compute $e(g_1, g_2)$ for $\forall g_1, g_2 \in \mathbb{G}$.

2.2 Identity-Based Chameleon Hash

A identity-based chameleon hash (IBCH) scheme [2] is a tuple of algorithms described as follows:

Setup takes as input a security parameter λ. It generates a public/private key pair (pk, sk), publishes pk and keeps sk secret. This algorithm is run by a trusted party, called private key generator (PKG).

$$(pk, sk) \leftarrow \mathsf{Setup}(\lambda).$$

$\mathsf{Extract}$ takes as input sk and an identity ID. It outputs the trapdoor information sk_{ID} associated with the identity. This algorithm is run by PKG.

$$sk_{\mathsf{ID}} \leftarrow \mathsf{Extract}(sk, \mathsf{ID}).$$

Hash takes as input pk, an identity ID and a message m. It chooses a randomness r and outputs a hash value h.

$$h \leftarrow \mathsf{Hash}(pk, \mathsf{ID}, m, r).$$

Forge takes as input an identity ID, the trapdoor information sk_{ID} associated with ID, the hash value h on a message m with r, and a new message m'. It outputs a value r'.

$$r' \leftarrow \mathsf{Forge}(sk_{\mathsf{ID}}, \mathsf{ID}, m, r, h, m').$$

For correctness, it requires that

$$\mathsf{Hash}(pk, \mathsf{ID}, m, r) = h = \mathsf{Hash}(pk, \mathsf{ID}, m', r' = \mathsf{Forge}(sk_{\mathsf{ID}}, \mathsf{ID}, m, r, h, m')) \text{ and } m' \neq m.$$

The security of an IBCH scheme consists of two requirements: *resistance to collision forgery under active attacks* and *semantic security*. In the following and throughout the rest of the paper, we use $\mathcal{A}$ to denote an adversary which can be any probabilistic polynomial-time algorithm.

Resistance to collision forgery under active attacks: The IBCH scheme is secure against (existential) collision forgery under active attacks, if for any

PPT adversary $\mathcal{A}$, for all sufficiently large λ, any $(pk, sk) \leftarrow \mathsf{Setup}(\lambda)$, the probability that $\mathcal{A}(\lambda, pk)$ outputs $(\mathsf{ID}, m, r, m', r')$, satisfying $\mathsf{Hash}(pk, \mathsf{ID}, m, r) = \mathsf{Hash}(pk, \mathsf{ID}, m', r')$ and $m' \neq m$, is negligible. $\mathcal{A}$ is allowed to query an oracle $\mathcal{O}_{IBCH}^{\mathsf{Extract}}$ [1] on adaptively chosen identities other than ID.

Semantic security: The IBCH scheme is said to be semantically secure if, for all sufficiently large λ, any $(pk, sk) \leftarrow \mathsf{Setup}(\lambda)$, any target identity ID and all pairs of messages m and m', the distribution ensembles $\{\mathsf{Hash}(pk, \mathsf{ID}, m; \mathcal{R})\}_{\lambda, pk, \mathsf{ID}, m, m'}$ and $\{\mathsf{Hash}(pk, \mathsf{ID}, m'; \mathcal{R})\}_{\lambda, pk, \mathsf{ID}, m, m'}$ are computationally indistinguishable.

3 Hierarchical Identity-Based Chameleon Hash

Like an IBCH scheme, a hierarchical identity-based chameleon hash (HIBCH) scheme consists of four algorithms: Setup, $\mathsf{Extract}$, Hash and Forge. In HIBCH, however, identities are organized into a hierarchy, where an identity at depth k of the hierarchy is represented as a vector of dimension k, and the trapdoor information for an identity is generated by its parent. Concretely, an ℓ-HIBCH scheme consists of the following algorithms:

Setup takes as input a security parameter λ and the maximum hierarchy depth ℓ that is polynomial in λ. It generates a public/private key pair (pk, sk), publishes pk and keeps sk secret. This algorithm is run by PKG.

$$(pk, sk) \leftarrow \mathsf{Setup}(\lambda, \ell).$$

$\mathsf{Extract}$ takes as an identity $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_k)$ at depth $k \leq \ell$, and the trapdoor information $sk_{\mathsf{ID}_{|k-1}}$ of the parent identity $\mathsf{ID}_{|k-1} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_{k-1})$ at depth $k-1$. It outputs the trapdoor information sk_{ID} for identity ID.

$$sk_{\mathsf{ID}} \leftarrow \mathsf{Extract}(sk_{\mathsf{ID}_{|k-1}}, \mathsf{ID}).$$

Note that, if $k = 1$, the trapdoor information $sk_{\mathsf{ID}_{|k-1}}$ of the identity $\mathsf{ID}_{|k-1}$ is sk. Running $\mathsf{Extract}$ algorithm recursively, an identity $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_k)$, using its trapdoor information sk_{ID}, can generate trapdoor information for all its descendants. So, we also can denote this algorithm as

$$sk_{\mathsf{ID}'} \leftarrow \mathsf{Extract}(sk_{\mathsf{ID}}, \mathsf{ID}'),$$

where ID' is a descendant of ID.

Hash takes as input pk, an identity ID and a message m. It chooses a randomness r and outputs a hash value h.

$$h \leftarrow \mathsf{Hash}(pk, \mathsf{ID}, m, r).$$

[1] When the adversary $\mathcal{A}$ queries $\mathcal{O}_{IBCH}^{\mathsf{Extract}}$ on an identity ID', the simulator gives the trapdoor information $sk_{\mathsf{ID}'}$ associated with ID' to $\mathcal{A}$.

Forge takes as input an identity ID, the trapdoor information sk_{ID} associated with ID, the hash value h on a message m with randomness r, and a new message m'. It outputs r'.

$$r' \leftarrow \mathsf{Forge}(sk_{\mathsf{ID}}, \mathsf{ID}, m, r, h, m').$$

For correctness, it requires that

$$\mathsf{Hash}(pk, \mathsf{ID}, m, r) = h = \mathsf{Hash}(pk, \mathsf{ID}, m', r' = \mathsf{Forge}(sk_{\mathsf{ID}}, \mathsf{ID}, m, r, h, m')) \text{ and } m' \neq m.$$

We now introduce the security requirements of HIBCH, including *resistance to collision forgery under active attacks*, *semantic security* and *forgery indistinguishability*. The security requirements of *resistance to collision forgery under active attacks* and *semantic security* are extended from the security requirements of IBCH. For *forgery indistinguishability*, informally, it requires that an adversary be not able to decide whether (m, r, h) is a forgery or not.

Resistance to collision forgery under active attacks: The HIBCH scheme is secure against (existential) collision forgery under active attacks, if for any PPT adversary $\mathcal{A}$, for any sufficiently large λ, any hierarchy depth ℓ that is polynomial in λ, any $(pk, sk) \leftarrow \mathsf{Setup}(\lambda, \ell)$, the probability that $\mathcal{A}(\lambda, pk)$ outputs $(\mathsf{ID}, m, r, m', r')$, satisfying $\mathsf{Hash}(pk, \mathsf{ID}, m, r) = \mathsf{Hash}(pk, \mathsf{ID}, m', r')$ and $m' \neq m$, is negligible. $\mathcal{A}$ is allowed to query an oracle $\mathcal{O}^{\mathsf{Extract}}_{HIBCH}$[2] on adaptively chosen identities other than ID or an ancestor of ID.

We say that an HIBCH scheme is t-threshold resistant to collision forgery under active attacks (or t-threshold collusion-resistant for simplicity.), if $\mathcal{A}$ issues at most t queries to its $\mathcal{O}^{\mathsf{Extract}}_{HIBCH}$ oracle on identities at each depth of the hierarchy.

Semantic security: The HIBCH scheme is said to be semantically secure if, for all sufficiently large λ, any hierarchy depth ℓ that is polynomial in λ, any $(pk, sk) \leftarrow \mathsf{Setup}(\lambda, \ell)$, all identities ID and all pairs of messages m and m', the distribution ensembles $\{\mathsf{Hash}(pk, \mathsf{ID}, m; \mathcal{R})\}_{\lambda, pk, \mathsf{ID}, m, m'}$ and $\{\mathsf{Hash}(pk, \mathsf{ID}, m'; \mathcal{R})\}_{\lambda, pk, \mathsf{ID}, m, m'}$ are computationally indistinguishable.

Forgery indistinguishability: The HIBCH scheme is said to be forgery-indistinguishable if, for all sufficiently large λ, any hierarchy depth ℓ that is polynomial in λ, any $(pk, sk) \leftarrow \mathsf{Setup}(\lambda, \ell)$, all identities ID and all pairs of messages m and m', the following distribution ensembles are computationally indistinguishable:

$$\begin{aligned}
\mathcal{D}_{\mathsf{Forge}} &= \{(m', \hat{r}, h) | r \xleftarrow{\$} \mathcal{R}, h \leftarrow \mathsf{Hash}(pk, \mathsf{ID}, m, r), sk_{\mathsf{ID}} \leftarrow \mathsf{Extract}(sk, \mathsf{ID}), \\
&\qquad \hat{r} \leftarrow \mathsf{Forge}(sk_{\mathsf{ID}}, \mathsf{ID}, m, r, h, m')\}_{\lambda, pk, \mathsf{ID}}, \\
\mathcal{D}_{\mathsf{Hash}} &= \{(m', r', h') | r' \xleftarrow{\$} \mathcal{R}, h' \leftarrow \mathsf{Hash}(pk, \mathsf{ID}, m', r')\}_{\lambda, pk, \mathsf{ID}}.
\end{aligned}$$

[2] When the adversary $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{Extract}}_{HIBCH}$ on an identity ID', the simulator gives the trapdoor information $sk_{\mathsf{ID}'}$ associated with ID' to $\mathcal{A}$.

4 Hierarchical Trapdoor Sanitizable Signature and Its Construction from HIBCH

In this section, we first introduce the notion of hierarchical trapdoor sanitizable signature (HTSS), which is a hierarchical extension of TSS, and extend the standard security definitions of TSS for the hierarchical setting. Then, we propose a generic construction of HTSS from HIBCH. The construction is similar to the construction of TSS from IBCH [8].

4.1 Hierarchical Trapdoor Sanitizable Signature

Informally, in an HTSS scheme, an identity associated with an entity who has the power of sanitization for a given signed message, can delegate its rights to its descendant identities in a controlled manner. In the following, to simplify the description, we will use the terms identity and entity interchangeably.

In the sequel we assume that each signed message $m = m_1 \| \cdots \| m_L$ is partitioned into L blocks, where L is an positive integer. We define a hierarchical sanitizable description $\mathtt{ADM}$ of m using a tree. Each leaf node of the tree is labeled by a distinct block index $i \in [1, L]$, which indicates the block is sanitizable. Each node of the tree is associated with an identity. The identity of a node at depth k is a $k+1$-dimensional vector, and the first k components of the identity is inherited from its parent. The identity of the root node is computed from special strings, which may include the signer and recipient information as well as some nonce or time-stamp. We say that an identity ID matches $\mathtt{ADM}$ if an internal node of the tree is associated with the identity ID.

A pictorial depiction of a hierarchical sanitizable description $\mathtt{ADM}$ is given in Figure 1, where $L = 8$. We can obtain from the $\mathtt{ADM}$ that the set of indices $I = \{1, 4, 5, 8\}$ that are sanitizable, and identities ID_0, $(\mathsf{ID}_0, \mathsf{ID}_1^1)$, $(\mathsf{ID}_0, \mathsf{ID}_1^2)$, $(\mathsf{ID}_0, \mathsf{ID}_1^1, \mathsf{ID}_2^1)$ and $(\mathsf{ID}_0, \mathsf{ID}_1^1, \mathsf{ID}_2^2)$ match the $\mathtt{ADM}$.

Like a TSS scheme, an HTSS scheme consists of five algorithms: KeyGen, Sign, $\mathsf{Trapdoor}$, $\mathsf{Sanitize}$ and Verify. In HTSS, however, Sign algorithm takes as input a hierarchical sanitizable description $\mathtt{ADM}$, not only a set of the indices $I \subseteq [1, L]$

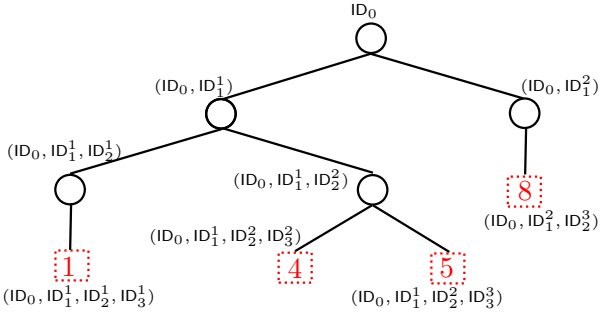

Fig. 1. An example of hierarchical sanitizable description

that are sanitizable in TSS, and the trapdoor information for an identity is generated by its parent, which runs Trapdoor algorithm. Concretely, an HTSS scheme consists of the following algorithms:

KeyGen takes as input a security parameter λ and the maximum hierarchy depth ℓ of hierarchical sanitizable descriptions. It generates a public/private key pair (pk, sk), publishes pk and keeps sk secret.

$$(pk, sk) \leftarrow \mathsf{KeyGen}(\lambda, \ell).$$

Sign takes as input a message $m = m_1 \| \cdots \| m_L$, a hierarchial sanitizable description $\mathtt{ADM}$ and sk. It outputs a signature σ on the message m.

$$\sigma \leftarrow \mathsf{Sign}(m, \mathtt{ADM}, sk).$$

Trapdoor takes as input a message m, a valid signature σ on m, a hierarchial sanitizable description $\mathtt{ADM}$, the trapdoor $sk_{\mathsf{ID}_{|k-1}}$ of the identity $\mathsf{ID}_{|k-1} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_{k-1})$, and a child identity $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_k)$. It outputs a trapdoor sk_{ID} associated with ID.

$$sk_{\mathsf{ID}} \leftarrow \mathsf{Trapdoor}(m, \mathtt{ADM}, \sigma, \mathsf{ID}, sk_{\mathsf{ID}_{|k-1}}).$$

Note that, if $k = 1$, the trapdoor $sk_{\mathsf{ID}_{|k-1}}$ of the identity $\mathsf{ID}_{|k-1}$ is sk. Running Trapdoor algorithm recursively, an identity $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_k)$ with its trapdoor sk_{ID} can generate the trapdoors for all its descendants.

Sanitize takes as input pk, a message m, a valid signature σ on m, a hierarchial sanitizable description $\mathtt{ADM}$, a trapdoor sk_{ID} associated with identity ID, a new message m'. It outputs a new signature σ' on m'.

$$\sigma' \leftarrow \mathsf{Sanitize}(pk, m, \mathtt{ADM}, \sigma, m', \mathsf{ID}, sk_{\mathsf{ID}}).$$

Verify takes as input pk, a message m, a putative signature σ and a hierarchial sanitizable description $\mathtt{ADM}$. It outputs 1 if the signature σ on m is valid and 0 otherwise.

$$0/1 \leftarrow \mathsf{Verify}(pk, m, \mathtt{ADM}, \sigma).$$

For an HTSS scheme the usual correctness properties should hold, saying that genuinely signed or sanitized messages are accepted. Formally, for correctness, an HTSS scheme must satisfy the following condition. For any security parameter λ and maximum hierarchy depth ℓ of hierarchical sanitizable descriptions, any message $m = m_1 \| \cdots \| m_L$, any hierarchial sanitizable description $\mathtt{ADM}$, any identity $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_k)$ and the trapdoor $sk_{\mathsf{ID}_{|k-1}}$ of the parent identity $\mathsf{ID}_{|k-1} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_{k-1})$, letting $(pk, sk) \leftarrow \mathsf{KeyGen}(\lambda, \ell)$, $\sigma \leftarrow \mathsf{Sign}(m, \mathtt{ADM}, sk)$, $sk_{\mathsf{ID}} \leftarrow \mathsf{Trapdoor}(m, \mathtt{ADM}, \sigma, \mathsf{ID}, sk_{\mathsf{ID}_{|k-1}})$, $\sigma' \leftarrow \mathsf{Sanitize}(pk, m, \mathtt{ADM}, \sigma, m', \mathsf{ID}, sk_{\mathsf{ID}})$,

1. $\mathsf{Verify}(pk, m, \mathtt{ADM}, \sigma) = 1$.
2. $\mathsf{Verify}(pk, m', \mathtt{ADM}, \sigma') = 1$.

The security requirements of an HTSS scheme include *unforgeability* and *indistinguishability*, which are extended from the security requirements of TSS. Informally, unforgeability requires that an outsider be not able to forger a signature on the original or the sanitized message, and indistinguishability requires that an outsider be not able to decide whether a message has been sanitized or not.

Unforgeability: An HTSS scheme is existential unforgeable under adaptive chosen message attacks, if for all sufficiently large λ, any hierarchy depth ℓ that is polynomial in λ, any $(pk, sk) \leftarrow \mathsf{Setup}(\lambda, \ell)$, any PPT adversary $\mathcal{A}(\lambda, pk)$, after issuing $\mathcal{O}_{HTSS}^{\mathsf{Sign}}$, $\mathcal{O}_{HTSS}^{\mathsf{Trapdoor}}$ and $\mathcal{O}_{HTSS}^{\mathsf{Sanitize}}$[3] oracle queries adaptively, with only negligible probability, can output $(m^*, \mathtt{ADM}^*, \sigma^*)$ such that:

1. $\mathsf{Verify}(pk, m^*, \mathtt{ADM}^*, \sigma^*) = 1$;
2. $\mathcal{A}$ never queries $\mathcal{O}_{HTSS}^{\mathsf{Sign}}$ oracle on $(m^*, \cdot)$;
3. (m^*, σ^*) does not come from $\mathcal{O}_{HTSS}^{\mathsf{Sanitize}}$ oracle, i. e., $\mathcal{A}$ never queries $\mathcal{O}_{HTSS}^{\mathsf{Sanitize}}$ oracle on $(m, \cdot, \sigma, m^*, \cdot)$;
4. $\mathcal{A}$ never queries $\mathcal{O}_{HTSS}^{\mathsf{Trapdoor}}$ oracle on $(m, \mathtt{ADM}, \sigma, \mathsf{ID})$ such that $m_i = m_i^*$ for all $i \notin I$, where I is extracted from $\mathtt{ADM}$ and is a set of indices $I \subseteq [1, L]$ that are sanitizable.

Indistinguishability: Indistinguishability of an HTSS scheme demands that the output distributions of Sign algorithm and $\mathsf{Sanitize}$ algorithm be computationally indistinguishable. In other words, for all sufficiently large λ, any hierarchy depth ℓ that is polynomial in λ, any $(pk, sk) \leftarrow \mathsf{Setup}(\lambda, \ell)$, any hierarchial sanitizable description $\mathtt{ADM}$, all message pairs m, m' such that $m_i = m'_i$ for all $i \notin I$, where I is extracted from $\mathtt{ADM}$ and is a set of indices $I \subseteq [1, L]$ that are sanitizable, any identity $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_k)$ that matches $\mathtt{ADM}$ and the trapdoor $sk_{\mathsf{ID}_{|k-1}}$ of the parent identity $\mathsf{ID}_{|k-1} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_{k-1})$, the following distribution ensembles $\mathcal{D}_{\mathsf{Sanitize}}$ and $\mathcal{D}_{\mathsf{Sign}}$ are computationally indistinguishable:

$$\begin{aligned}\mathcal{D}_{\mathsf{Sanitize}} &= \{(m', \hat{\sigma}) | \sigma \leftarrow \mathsf{Sign}(m, \mathtt{ADM}, sk), sk_{\mathsf{ID}} \leftarrow \mathsf{Trapdoor}(m, \mathtt{ADM}, \sigma, \mathsf{ID}, sk_{\mathsf{ID}_{|k-1}}),\\ &\qquad \hat{\sigma} \leftarrow \mathsf{Sanitize}(pk, m, \mathtt{ADM}, \sigma, m', \mathsf{ID}, sk_{\mathsf{ID}})\}_{\lambda, pk, \mathsf{ID}, \mathtt{ADM}},\\ \mathcal{D}_{\mathsf{Sign}} &= \{(m', \sigma') | \sigma' \leftarrow \mathsf{Sign}(m', \mathtt{ADM}, sk)\}_{\lambda, pk, \mathsf{ID}, \mathtt{ADM}}.\end{aligned}$$

4.2 Generic Construction of HTSS from HIBCH

In our construction, to sign a message $m = m_1 \| \cdots \| m_L$, the signer first sets $\tilde{m} = \tilde{m}_1 \| \cdots \| \tilde{m}_L$, where $\tilde{m}_i = m_i$ if $i \notin I$ and otherwise, $\tilde{m}_i = h_i = \mathsf{HIBCH.Hash}(pk, \mathsf{ID}^{(i)}, m_i, r_i)$. The set of indices $I \subseteq [1, L]$ that are sanitizable and the identity $\mathsf{ID}^{(i)}$ associated with the sanitizable block index $i \in I$ are given in the hierarchial sanitizable description $\mathtt{ADM}$. Then, the signer signs the message

[3] When the adversary $\mathcal{A}$ queries $\mathcal{O}_{HTSS}^{\mathsf{Sign}}$ on a message $(m, \mathtt{ADM})$, the simulator gives a valid signature $\sigma = \mathsf{Sign}(m, \mathtt{ADM}, sk)$ on m to $\mathcal{A}$. When the adversary $\mathcal{A}$ queries $\mathcal{O}_{HTSS}^{\mathsf{Trapdoor}}$ on $(m, \mathtt{ADM}, \sigma, \mathsf{ID})$, the simulator gives the corresponding trapdoor sk_{ID} to $\mathcal{A}$. When the adversary $\mathcal{A}$ queries $\mathcal{O}_{HTSS}^{\mathsf{Sanitize}}$ on $(m, \mathtt{ADM}, \sigma, m', \mathsf{ID})$, the simulator gives a valid signature σ' on m' to $\mathcal{A}$.

$\tilde{m}$ using a conventional signature scheme. Obviously, an entity with the trapdoor associated with $\mathsf{ID}^{(i)}$ or an ancestor of $\mathsf{ID}^{(i)}$ can modify m_i and generate a new signature on the sanitized message.

Given a conventional signature scheme $\Sigma = (\Sigma.\mathsf{KeyGen}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Verify})$ and an HIBCH scheme $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Extract}, \Pi.\mathsf{Hash}, \Pi.\mathsf{Forge})$, we define the 5-tuple algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Trapdoor}, \mathsf{Sanitize}, \mathsf{Verify})$ of an HTSS scheme as follows:

KeyGen Given a security parameter λ and the maximum hierarchy depth ℓ of hierarchical sanitizable descriptions, it first runs

$$(pk_\Sigma, sk_\Sigma) \leftarrow \Sigma.\mathsf{KeyGen}(\lambda),\ (pk_\Pi, sk_\Pi) \leftarrow \Pi.\mathsf{Setup}(\lambda, \ell + 1).$$

Then, it sets the public key $pk = (pk_\Sigma, pk_\Pi)$ and the private key $sk = (sk_\Sigma, sk_\Pi)$. Finally, it publishes pk and keeps sk secret.

Sign Given a message $m = m_1 \| \cdots \| m_L$, a hierarchial sanitizable description $\mathtt{ADM}$ and $sk = (sk_\Sigma, sk_\Pi)$, it first extracts a set of indices $I \subseteq [1, L]$ that are sanitizable from $\mathtt{ADM}$. Then, it proceeds as follows.

1. For all $i \in [1, L] \backslash I$, it sets $\tilde{m}_i = m_i$.
2. For all $i \in I$, let $\mathsf{ID}^{(i)}$ be the identity of the leaf node of $\mathtt{ADM}$ labeled by the block index i, it chooses a randomness r_i uniformly, and computes $h_i = \Pi.\mathsf{Hash}(pk_\Pi, \mathsf{ID}^{(i)}, m_i, r_i)$ and sets $\tilde{m}_i = h_i$. Let r be the concatenation of all random values r_i, $i \in I$.
3. It sets $\tilde{m} = \tilde{m}_1 \| \cdots \| \tilde{m}_L$ and runs
$$\tilde{\sigma} \leftarrow \Sigma.\mathsf{Sign}(\tilde{m}, sk_\Sigma).$$
4. Finally, it sets $\sigma = \tilde{\sigma} \| r$ and outputs the signature σ on m.

$\mathsf{Trapdoor}$ Given a message m, a valid signature σ on m, a hierarchial sanitizable description $\mathtt{ADM}$, an identity $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_k)$, and the trapdoor $sk_{\mathsf{ID}_{|k-1}}$ of the parent identity $\mathsf{ID}_{|k-1} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_{k-1})$, it first checks whether ID matches $\mathtt{ADM}$. If not, it outputs $\perp$, denoted an error. Otherwise, it runs

$$sk_{\mathsf{ID}} \leftarrow \Pi.\mathsf{Extract}(sk_{\mathsf{ID}_{|k-1}}, \mathsf{ID}),$$

and outputs the trapdoor sk_{ID} associated with ID.

Note that, if $k = 1$, the trapdoor $sk_{\mathsf{ID}_{|k-1}}$ of the identity $\mathsf{ID}_{|k-1}$ is sk_Π.

$\mathsf{Sanitize}$ Given $pk = (pk_\Sigma, pk_\Pi)$, a message $m = m_1 \| \cdots \| m_L$, a valid signature $\sigma = \tilde{\sigma} \| r$ on m, a hierarchial sanitizable description $\mathtt{ADM}$, a trapdoor sk_{ID} associated with identity ID, a new message $m' = m'_1 \| \cdots \| m'_L$, it proceeds as follows.

1. Let $I' = \{i \in [1, L] | m_i \neq m'_i\}$. It extracts a set of indices $I \subseteq [1, L]$ that are sanitizable from $\mathtt{ADM}$. Then, it checks whether $I' \subseteq I$. If not, it outputs $\perp$, denoted an error.
2. It checks whether ID matches $\mathtt{ADM}$. If not, it outputs $\perp$.

3. For all $i \in I'$, let $\mathsf{ID}^{(i)}$ be the identity of the leaf node of $\mathtt{ADM}$ labeled by the block index i, it checks whether $\mathsf{ID}^{(i)}$ is a descendant of ID. If not, it outputs $\perp$. Otherwise, it runs

$$sk_{\mathsf{ID}^{(i)}} \leftarrow \Pi.\mathsf{Extract}(sk_{\mathsf{ID}}, \mathsf{ID}^{(i)}),$$

to obtain the trapdoor $sk_{\mathsf{ID}^{(i)}}$ associated with $\mathsf{ID}^{(i)}$.
4. It retrieves $\{r_i | i \in I\}$ from the signature $\sigma = \tilde{\sigma} \| r$.
5. For all $i \in I'$, it computes $h_i \leftarrow \Pi.\mathsf{Hash}(pk_\Pi, \mathsf{ID}^{(i)}, m_i, r_i)$ and

$$r'_i \leftarrow \Pi.\mathsf{Forge}(sk_{\mathsf{ID}^{(i)}}, \mathsf{ID}^{(i)}, m_i, r_i, h_i, m'_i).$$

6. For all $i \in I \backslash I'$, it sets $r'_i = r_i$. Let r' be the concatenation of all random values r'_i, $i \in I$.
7. It sets $\sigma' = \tilde{\sigma} \| r'$ and outputs the new signature σ' on m'.

Verify Given $pk = (pk_\Sigma, pk_\Pi)$, a message $m = m_1 \| \cdots \| m_L$, a putative signature $\sigma = \tilde{\sigma} \| r$ and a hierarchial sanitizable description $\mathtt{ADM}$, it proceeds as follows.

1. It extracts a set of indices $I \subseteq [1, L]$ that are sanitizable from $\mathtt{ADM}$ and retrieves $\{r_i | i \in I\}$ from the signature $\sigma = \tilde{\sigma} \| r$.
2. For all $i \in [1, L] \backslash I$, it sets $\tilde{m}_i = m_i$.
3. For all $i \in I$, let $\mathsf{ID}^{(i)}$ be the identity of the leaf node of $\mathtt{ADM}$ labeled by the block index i, it computes $h_i = \Pi.\mathsf{Hash}(pk_\Pi, \mathsf{ID}^{(i)}, m_i, r_i)$ and sets $\tilde{m}_i = h_i$.
4. It sets $\tilde{m} = \tilde{m}_1 \| \cdots \| \tilde{m}_L$ and outputs $\Sigma.\mathsf{Verify}(pk_\Sigma, \tilde{m}, \tilde{\sigma})$.

It is obvious that the above HTSS scheme satisfies correctness. We now state the security theorems of the above HTSS scheme, including *unforgeability* and *indistinguishability*. The proofs of the security theorems are similar to those in [8] and will be given in the full version of the paper.

Theorem 1 (Unforgeability). *If the signature scheme Σ is existential unforgeable under adaptive chosen message attacks [16] and the HIBCH scheme Π is resistant to collision forgery under active attacks, the above construction of HTSS is existential unforgeable under adaptive chosen message attacks.*

Theorem 2 (Indistinguishability). *If the HIBCH scheme Π is forgery indistinguishable, the following distributions $\mathcal{D}_{\mathsf{Sanitize}}$ and $\mathcal{D}_{\mathsf{Sign}}$ are indistinguishable for all sufficiently large λ, any hierarchy depth ℓ that is polynomial in λ, any $(pk, sk) \leftarrow \mathsf{Setup}(\lambda, \ell)$, any hierarchial sanitizable description $\mathtt{ADM}$, messages m, m' such that $m_i = m'_i$ for all $i \notin I$, where I is extracted from $\mathtt{ADM}$ and is a set of indices $I \subseteq [1, L]$ that are sanitizable, any identity $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_k)$ that matches $\mathtt{ADM}$ and the trapdoor $sk_{\mathsf{ID}_{|k-1}}$ of the parent identity $\mathsf{ID}_{|k-1} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_{k-1})$:*

$$\begin{aligned}
\mathcal{D}_{\mathsf{Sanitize}} &= \{(m', \hat{\sigma}) | \sigma \leftarrow \mathsf{Sign}(m, \mathtt{ADM}, sk), sk_{\mathsf{ID}} \leftarrow \mathsf{Trapdoor}(m, \mathtt{ADM}, \sigma, \mathsf{ID}, sk_{\mathsf{ID}_{|k-1}}), \\
&\qquad \hat{\sigma} \leftarrow \mathsf{Sanitize}(pk, m, \mathtt{ADM}, \sigma, m', \mathsf{ID}, sk_{\mathsf{ID}})\}_{\lambda, pk, \mathsf{ID}, \mathtt{ADM}}, \\
\mathcal{D}_{\mathsf{Sign}} &= \{(m', \sigma') | \sigma' \leftarrow \mathsf{Sign}(m', \mathtt{ADM}, sk)\}_{\lambda, pk, \mathsf{ID}, \mathtt{ADM}}.
\end{aligned}$$

5 Key-Exposure Free IBCH from HIBCH

As mentioned in [2,3], a key-exposure free chameleon hash scheme can be obtained from an IBCH scheme. In the construction of key-exposure free chameleon hash from IBCH, each transaction uses a different public key (corresponding to a different private key), so that a forgery only results in the user recovering the trapdoor information associated with a single transaction. The transaction-specific public key, called *customized identity*, is computed from special strings that describe the transaction. Based on the same idea, we show that a 2-HIBCH can be used to construct a key-exposure free IBCH scheme.

In this section, we first review the notion of key exposure freeness. Then, we describe the generic construction of key-exposure free IBCH from a two-level HIBCH formally.

Key Exposure Freeness: An identity-based chameleon hash scheme is key-exposure free if, for any PPT adversary $\mathcal{A}$, for all sufficiently large λ, any $(pk, sk) \leftarrow \mathsf{Setup}(\lambda)$, the probability that, $\mathcal{A}(\lambda, pk)$ outputs $(\mathsf{ID}, \mathcal{L}, m, r, m', r')$, satisfying $\mathsf{Hash}(pk, \mathsf{ID}, \mathcal{L}, m, r) = \mathsf{Hash}(pk, \mathsf{ID}, \mathcal{L}, m', r')$ and $m' \neq m$, is negligible. $\mathcal{A}$ is allowed to query $\mathcal{O}_{IBCH}^{\mathsf{Forge}}$ [4] oracle on the adaptively chosen tuples $(\mathsf{ID}, \mathcal{L}_i, m_i, r_i, m_i')$, except that $\mathcal{L}_i$ must be different from the target *customized identity* $\mathcal{L}$.

Now, given an HIBCH scheme $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Extract}, \Pi.\mathsf{Hash}, \Pi.\mathsf{Forge})$, we define the 4-tuple algorithms $(\mathsf{Setup}, \mathsf{Extract}, \mathsf{Hash}, \mathsf{Forge})$ of an IBCH scheme as follows:

Setup Given a security parameter λ, PKG first runs

$$(pk, sk) \leftarrow \Pi.\mathsf{Setup}(\lambda, 2).$$

Then, it publishes the public key pk and keeps the private key sk secret.

$\mathsf{Extract}$ Given the private key sk and an identity ID, it first runs

$$sk_{\mathsf{ID}} \leftarrow \Pi.\mathsf{Extract}(sk, \mathsf{ID}).$$

Then, it outputs the trapdoor information sk_{ID} associated with the identity.

Hash Given the public key pk, an identity ID and a message m, it first computes the *customized identity* $\mathcal{L}$ for this transaction, and chooses a randomness r. Then, it sets a 2-level identity $\widetilde{\mathsf{ID}} = (\mathsf{ID}, \mathcal{L})$ and runs

$$h \leftarrow \Pi.\mathsf{Hash}(pk, \widetilde{\mathsf{ID}}, m, r).$$

Finally, it outputs the hash value h.

Forge Given an identity ID, the trapdoor information sk_{ID} association with ID, the hash value h on a message m with *customized identity* $\mathcal{L}$ and randomness r, and a new message m', it first sets a 2-level identity $\widetilde{\mathsf{ID}} = (\mathsf{ID}, \mathcal{L})$ and runs

[4] When the adversary $\mathcal{A}$ queries $\mathcal{O}_{IBCH}^{\mathsf{Forge}}$ on $(\mathsf{ID}, \mathcal{L}_i, m_i, r_i, m_i')$, the simulator gives the randomness r_i' to $\mathcal{A}$ such that $\mathsf{Hash}(pk, \mathsf{ID}, \mathcal{L}_i, m_i, r_i) = \mathsf{Hash}(pk, \mathsf{ID}, \mathcal{L}_i, m_i', r_i')$.

$$sk_{\widetilde{\mathsf{ID}}} \leftarrow \Pi.\mathsf{Extract}(sk_{\mathsf{ID}}, \widetilde{\mathsf{ID}}).$$

Then it runs

$$r' \leftarrow \Pi.\mathsf{Forge}(sk_{\widetilde{\mathsf{ID}}}, \widetilde{\mathsf{ID}}, m, r, h, m'),$$

and outputs r'.

It is obvious that, if the HIBCH scheme Π satisfies correctness, the above IBCH scheme also satisfies correctness, and if the HIBCH scheme Π is resistant to collision forgery under active attacks and semantically secure, so is the above IBCH scheme. Next, we prove that the above IBCH scheme is key-exposure free.

Theorem 3. *If the HIBCH scheme Π is resistant to collision forgery under active attacks, the above IBCH scheme is key-exposure free.*

Proof. To prove this theorem, we will show that, given pk, if a PPT adversary $\mathcal{A}$ can output $(\mathsf{ID}, \mathcal{L}, m, r, m', r')$ such that $\mathsf{Hash}(pk, \mathsf{ID}, \mathcal{L}, m, r) = \mathsf{Hash}(pk, \mathsf{ID}, \mathcal{L}, m', r')$ and $m' \neq m$, we can construct another algorithm $\mathcal{B}$, which is a forger against the HIBCH scheme Π.

Given a public key pk of the HIBCH scheme Π, using $\mathcal{A}$ as a sub-routine, $\mathcal{B}$ simulates a forger against the HIBCH scheme Π. First, $\mathcal{B}$ sends pk to $\mathcal{A}$. When $\mathcal{A}$ issues $\mathcal{O}_{IBCH}^{\mathsf{Forge}}$ oracle queries on the adaptively chosen tuples $(\mathsf{ID}, \mathcal{L}_i, m_i, r_i, m_i')$, $\mathcal{B}$ sets $\widetilde{\mathsf{ID}}_i = (\mathsf{ID}, \mathcal{L}_i)$ and queries its $\mathcal{O}_{HIBCH}^{\mathsf{Extract}}$ oracle on $\widetilde{\mathsf{ID}}_i$ to obtain the trapdoor information $sk_{\widetilde{\mathsf{ID}}_i}$; then $\mathcal{B}$ computes $\Pi.\mathsf{Hash}(pk, \widetilde{\mathsf{ID}}_i, m_i, r_i) = h_i$ and $\Pi.\mathsf{Forge}(sk_{\widetilde{\mathsf{ID}}_i}, \widetilde{\mathsf{ID}}_i, m_i, r_i, h_i, m_i') = r_i'$ and sends r_i' to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs $(\mathsf{ID}, \mathcal{L}, m, r, m', r')$ such that $\mathsf{Hash}(pk, \mathsf{ID}, \mathcal{L}, m, r) = \mathsf{Hash}(pk, \mathsf{ID}, \mathcal{L}, m', r')$. $\mathcal{B}$ also outputs $(\widetilde{\mathsf{ID}} = (\mathsf{ID}, \mathcal{L}), m, r, m', r')$, which is a collision against the HIBCH scheme Π.

6 Construction of HIBCH

In this section, based on multivariate polynomials, we propose a concrete construction of HIBCH, which is t-threshold collusion-resistant. Blundo et al. [6] first used multivariate polynomials to construct key distribution schemes, and Gennaro et al. [15] extended their schemes for hierarchical systems.

In our construction, the private key of the HIBCH scheme is a random multivariate polynomial $f(x_1, \ldots, x_\ell)$, where the degree of x_i is a threshold parameter t. The trapdoor information of an identity $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_k)$ is $f(\mathsf{ID}_1, \ldots, \mathsf{ID}_{k-1}, \mathsf{ID}_k, x_{k+1}, \ldots, x_\ell)$, which can be derived from his parent's trapdoor information $f(\mathsf{ID}_1, \ldots, \mathsf{ID}_{k-1}, x_k, \ldots, x_\ell)$. Blundo et al. [6] proved that, if an adversary colludes with at most t entities in each depth of the hierarchy, the multivariate polynomial $f(x_1, \ldots, x_\ell)$ can still be kept secret.

In fact, if we choose a random multivariate polynomial $f(x_1, \ldots, x_\ell)$ as the private key, where the degree of x_i is t_i, our HIBCH scheme is resilient against the adversary who colludes with at most t_i entities at depth i of the hierarchy.

The scheme consists of the following algorithms:

Setup Given a security parameter λ and the maximum hierarchy depth ℓ, it first generates a bilinear map group system $\langle p, \mathbb{G}, \mathbb{G}_T, e\rangle$. Then it chooses a random polynomial (over $\mathbb{Z}_p$) $f(x_1, \ldots, x_\ell) = a_{t,t,\ldots,t}x_1^t x_2^t \cdots x_\ell^t + a_{t-1,t,\ldots,t}x_1^{t-1}x_2^t \cdots x_\ell^t + \cdots + a_{0,0,\ldots,0}$, where the degree of x_i is the threshold parameter t. Next, it chooses a generator g of $\mathbb{G}$ and an idle identity $\overline{\mathsf{ID}} \in \mathbb{Z}_p$. Finally, it chooses a cryptographic hash function $H : \{0,1\}^* \to \mathbb{G}$. The published public key is

$$pk = (p, \mathbb{G}, \mathbb{G}_T, e,\ g,\ H,\ \overline{\mathsf{ID}},\ g^{a_{t,\ldots,t}}, \ldots, g^{a_{0,\ldots,0}}),$$

and the private key is $sk = f(x_1, \ldots, x_\ell)$.

Extract Given an identity $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_k)$ of depth $k \le \ell$, and the trapdoor information $sk_{\mathsf{ID}_{|k-1}} = f(\mathsf{ID}_1, \ldots, \mathsf{ID}_{k-1}, x_k, \ldots, x_\ell)$ of the parent identity $\mathsf{ID}_{|k-1} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_{k-1})$ at depth $k-1$, it first checks whether $\mathsf{ID}_i \neq \overline{\mathsf{ID}}$ for $1 \le i \le k$. If not, it outputs $\perp$, denoted an error. Otherwise, it computes

$$sk_{\mathsf{ID}} = f(\mathsf{ID}_1, \ldots, \mathsf{ID}_{k-1}, \mathsf{ID}_k, x_{k+1}, \ldots, x_\ell),$$

and outputs the trapdoor information sk_{ID} for identity ID.

Note that, if $k = 1$, the trapdoor information $sk_{\mathsf{ID}_{|k-1}}$ of the identity $\mathsf{ID}_{|k-1}$ is sk.

Hash Given pk, an identity $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_k)$ of depth $k \le \ell$, and a message $m \in \{0,1\}^*$, it first chooses a randomness $R \in \mathbb{G}$ uniformly. Then it computes

$$h = e(R, g) \cdot e(H(m), g^{f(\mathsf{ID}_1,\ldots,\mathsf{ID}_k,\overline{\mathsf{ID}},\ldots,\overline{\mathsf{ID}})}),$$

and outputs the hash value h.

Note that, given $g^{a_{t,\ldots,t}}, \ldots, g^{a_{0,\ldots,0}}$, one can compute $g^{f(\mathsf{ID}_1,\ldots,\mathsf{ID}_k,\overline{\mathsf{ID}},\ldots,\overline{\mathsf{ID}})}$.

Forge Given an identity $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_k)$, the trapdoor information $sk_{\mathsf{ID}} = f(\mathsf{ID}_1, \ldots, \mathsf{ID}_k, x_{k+1}, \ldots, x_\ell)$ associated with ID, the hash value h on a message $m \in \{0,1\}^*$ with randomness R, and a new message $m' \in \{0,1\}^*$, it first computes $f(\mathsf{ID}_1, \ldots, \mathsf{ID}_k, \overline{\mathsf{ID}}, \ldots, \overline{\mathsf{ID}})$ using sk_{ID}. Then it computes

$$R' = R \cdot (H(m) \cdot H(m')^{-1})^{f(\mathsf{ID}_1,\ldots,\mathsf{ID}_k,\overline{\mathsf{ID}},\ldots,\overline{\mathsf{ID}})},$$

and outputs the randomness R'.

Note that

$$\begin{aligned}
\mathsf{Hash}(pk, \mathsf{ID}, m', R') &= e(R', g) \cdot e(H(m'), g^{f(\mathsf{ID}_1,\ldots,\mathsf{ID}_k,\overline{\mathsf{ID}},\ldots,\overline{\mathsf{ID}})}) \\
&= e(R \cdot (H(m) \cdot H(m')^{-1})^{f(\mathsf{ID}_1,\ldots,\mathsf{ID}_k,\overline{\mathsf{ID}},\ldots,\overline{\mathsf{ID}})}, g) \\
&\quad \cdot e(H(m'), g^{f(\mathsf{ID}_1,\ldots,\mathsf{ID}_k,\overline{\mathsf{ID}},\ldots,\overline{\mathsf{ID}})}) \\
&= e(R, g) \cdot e(H(m), g^{f(\mathsf{ID}_1,\ldots,\mathsf{ID}_k,\overline{\mathsf{ID}},\ldots,\overline{\mathsf{ID}})}) \\
&= \mathsf{Hash}(pk, \mathsf{ID}, m, R).
\end{aligned}$$

So, the above construction of HIBCH satisfies *correctness*. We now state the security theorems of the above HIBCH scheme. The proofs of the security theorems will be given in the full version of the paper.

Theorem 4. *In the random oracle model [5], the above construction of HIBCH is t-threshold resistant to collision forgery under active attacks.*

Theorem 5. *The above construction of HIBCH is semantically secure.*

7 Conclusions

In this paper, we introduced the notion of HIBCH, which is a hierarchical extension of IBCH. We showed that HIBCH can be used to construct other cryptographic primitives, including HTSS, which is a hierarchical extension of TSS, and key-exposure free IBCH, even the HIBCH is not key-exposure free. Finally, we proposed a concrete construction of HIBCH, which is t-threshold collusion-resistant. A future direction is to find other constructions of HIBCH, which are fully collusion-resistant and key-exposure free.

Acknowledgement

We are grateful to the anonymous reviewers for their helpful comments. This research is supported in part by A*STAR SERC Grant No. 102 101 0027 in Singapore. Yunlei Zhao is partly supported by a grant from the Major State Basic Research Development (973) Program of China (No. 2007CB807901) and a grant from the National Natural Science Foundation of China NSFC (No. 61070248) and the QiMingXing Program of Shanghai.

References

1. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable Signatures. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)
2. Ateniese, G., de Medeiros, B.: Identity-based chameleon hash and applications. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 164–180. Springer, Heidelberg (2004)
3. Ateniese, G., de Medeiros, B.: On the key exposure problem in chameleon hashes. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 165–179. Springer, Heidelberg (2005)
4. Bellare, M., Goldreich, O., Goldwasser, S.: Incremental cryptography: The case of hashing and signing. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 216–233. Springer, Heidelberg (1994)
5. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73 (1993)
6. Blundo, C., De Santis, A., Herzberg, A., Kutten, S., Vaccaro, U., Yung, M.: Perfectly secure key distribution for dynamic conferences. Inf. Comput. 146(1), 1–23 (1998)
7. Brzuska, C., Fischlin, M., Freudenreich, T., Lehmann, A., Page, M., Schelbert, J., Schröder, D., Volk, F.: Security of sanitizable signatures revisited. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 317–336. Springer, Heidelberg (2009)

8. Canard, S., Laguillaumie, F., Milhau, M.: Trapdoor sanitizable signatures and their application to content protection. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 258–276. Springer, Heidelberg (2008)
9. Chaum, D., van Antwerpen, H.: Undeniable signatures. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 212–216. Springer, Heidelberg (1990)
10. Chen, X., Zhang, F., Kim, K.: Chameleon hashing without key exposure. In: Zhang, K., Zheng, Y. (eds.) ISC 2004. LNCS, vol. 3225, pp. 87–98. Springer, Heidelberg (2004)
11. Chen, X., Zhang, F., Susilo, W., Tian, H., Li, J., Kim, K.: Identity-based chameleon hash scheme without key exposure. In: Steinfeld, R., Hawkes, P. (eds.) ACISP 2010. LNCS, vol. 6168, pp. 200–215. Springer, Heidelberg (2010)
12. Chen, X., Zhang, F., Tian, H., Wei, B., Kim, K.: Key-exposure free chameleon hashing and signatures based on discrete logarithm systems. Cryptology ePrint Archive, Report 2009/035 (2009), http://eprint.iacr.org/
13. Gao, W., Li, F., Wang, X.: Chameleon hash without key exposure based on schnorr signature. Computer Standards & Interfaces 31(2), 282–285 (2009)
14. Gao, W., Wang, X., Xie, D.: Chameleon hashes without key exposure based on factoring. J. Comput. Sci. Technol. 22(1), 109–113 (2007)
15. Gennaro, R., Halevi, S., Krawczyk, H., Rabin, T., Reidt, S., Wolthusen, S.D.: Strongly-resilient and non-interactive hierarchical key-agreement in mANETs. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 49–65. Springer, Heidelberg (2008)
16. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. 17(2), 281–308 (1988)
17. Izu, T., Kanaya, N., Takenaka, M., Yoshioka, T.: PIATS: A partially sanitizable signature scheme. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 72–83. Springer, Heidelberg (2005)
18. Johnson, R., Molnar, D., Song, D.X., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)
19. Klonowski, M., Lauks, A.: Extended sanitizable signatures. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 343–355. Springer, Heidelberg (2006)
20. Krawczyk, H., Rabin, T.: Chameleon signatures. In: NDSS (2000)
21. Micali, S., Rivest, R.L.: Transitive signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 236–243. Springer, Heidelberg (2002)
22. Miyazaki, K., Hanaoka, G., Imai, H.: Invisibly sanitizable digital signature scheme. IEICE Transactions 91-A(1), 392–402 (2008)
23. Miyazaki, K., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H., Tezuka, S., Imai, H.: Digitally signed document sanitizing scheme with disclosure condition control. IEICE Transactions 88-A(1), 239–246 (2005)
24. Steinfeld, R., Bull, L., Zheng, Y.: Content extraction signatures. In: Kim, K.-c. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 285–304. Springer, Heidelberg (2002)
25. Suzuki, T., Ramzan, Z., Fujimoto, H., Gentry, C., Nakayama, T., Jain, R.: A system for end-to-end authentication of adaptive multimedia content. In: Dittmann, J., Katzenbeisser, S., Uhl, A. (eds.) CMS 2005. LNCS, vol. 3677, pp. 237–249. Springer, Heidelberg (2005)
26. Tan, K.W., Deng, R.H.: Applying santitizable signature to web-service-enabled business processes: going beyond integrity protection. In: ICWS, pp. 67–74 (2009)

27. Yum, D.H., Seo, J.W., Lee, P.J.: Trapdoor sanitizable signatures made easy. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 53–68. Springer, Heidelberg (2010)
28. Zhang, F., Safavi-Naini, R., Susilo, W.: Id-based chameleon hashes from bilinear pairings. Cryptology ePrint Archive, Report 2003/208 (2003), http://eprint.iacr.org/