Hierarchical Identity-Based Chameleon Hash and Its Applications

Feng Bao, Robert H. Deng, Xuhua Ding, Junzuo Lai, Yunlei Zhao

Institute for Infocomm Research, Singapore
Singapore Management University, Singapore
Fudan University

Outline

- Introduction
- Concept of Hierarchical Trapdoor Sanitizable Signature (HTSS)
- Hierarchical Identity-Based Chameleon Hash (HIBCH)
- Construction of HTSS using HIBCH
- Conclusion

Background: Chameleon Hash

- Informally, a chameleon hash function is a trapdoor collision-resistant hash function:
  - Without knowledge of the trapdoor, the chameleon hash function is collision-resistant.
  - Once the trapdoor is known, collision can be easily computed.
- Introduced by Krawczyk and Rabin (NDSS'00).
- Applications: chameleon signature, sanitizable signature.

Background: Sanitizable Signature (SS)

- At ESORICS 2005, Ateniese et al. introduced the notion of sanitizable signature and presented its generic construction based on chameleon hash.
- Informally:
  - 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.
- Applications: authenticated multicast, authenticated database outsourcing and secure routing.

Background: Trapdoor Sanitizable Signature (TSS)

- At ACNS 2008, Canard et al. introduced the notion of TSS and presented its generic construction based on identity-based chameleon hash.
- Informally, TSS allows the signer to delegate the power of sanitization for a specific signed message to possibly several entities.
- TSS vs. SS
  - The sanitizer is predetermined at the time of signature generation by the signer in SS.
  - The signer in TSS can choose to whom and when it will provide the trapdoor information and therefore, any entity can potentially acts as a sanitizer.

Motivation Scenario (1)

- Automated web-service-enabled business processes:
  - Consider a simple quotation response process, involving an electronic distributor (ED), a transportation company (TC), an electronic manufacturer and an electronic retailer (ER).
  - The quotation response process begins when ED receives a request for quotation from ER.
  - ED generates the quotation by providing quotes for each item and the taxes associated and forwards the quotation to TC.
  - 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.

Motivation Scenario (2)

- Tiered multimedia distribution systems:
  - A multinational company 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.

Hierarchical Trapdoor Sanitizable Signature

- We introduce the notion of hierarchical trapdoor sanitizable signature (HTSS).
- 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.
  - 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.

Participants

- Participants: Signer, a set of Sanitizers, a public Verifier.
- A sanitizer's identity is represented as $(\mathsf{ID}_0, \ldots, \mathsf{ID}_i, \ldots)$

Five Algorithms

- Signer: $(pk, sk) \leftarrow \mathsf{KeyGen}(\lambda, \ell)$
- Signer: $\sigma \leftarrow \mathsf{Sign}(m, \mathtt{ADM}, sk)$, where $\mathtt{ADM}$ is a hierarchical sanitizable description on m
- Sanitizer $\mathsf{ID}_{|k-1} = (\mathsf{ID}_0, \ldots, \mathsf{ID}_{k-1})$ authorizes another sanitizer $\mathsf{ID} = (\mathsf{ID}_0, \ldots, \mathsf{ID}_{k-1}, ID_k)$:

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

  where $sk_{\mathsf{ID}_{|k-1}}$ is the trapdoor of the identity $\mathsf{ID}_{|k-1}$. If $k = 1$, $sk_{\mathsf{ID}_{|k-1}} = sk$
- Sanitizer ID: $\sigma' \leftarrow \mathsf{Sanitize}(pk, m, \mathtt{ADM}, \sigma, m', \mathsf{ID}, sk_{\mathsf{ID}})$
- Verifier: $0/1 \leftarrow \mathsf{Verify}(pk, m, \mathtt{ADM}, \sigma)$.

Security Requirements

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

Review of Chameleon Hash and IBCH

Formally, a chameleon hash scheme consists of three algorithms:

- $(pk, sk) \leftarrow \mathsf{KeyGen}(\lambda)$
- Given m, everyone can compute $h \leftarrow \mathsf{Hash}(pk, m, r)$
- Given (m, h), the owner of sk can compute $r' \leftarrow \mathsf{Forge}(sk, m, r, h, m')$, such that

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

For identity-based chameleon hash (IBCH):

- PKG issues $sk_{\mathsf{ID}} \leftarrow \mathsf{Extract}(sk, \mathsf{ID})$ for user ID.
- ID is an input to Hash and Forge algorithms.

Hierarchical Identity-Based Chameleon Hash(1)

- We introduce the notion of hierarchical identity-based chameleon hash (HIBCH), which is a hierarchical extension of IBCH.
- HIBCH vs. IBCH
  - The same four algorithms: Setup, Extract, Hash and Forge.
  - In HIBCH, identities are organized into a hierarchy, where an identity at depth k of the hierarchy is represented as a vector of dimension k, e.g. $\mathsf{ID} = (\mathsf{ID}_0, \ldots, \mathsf{ID}_k)$, and the trapdoor information for an identity is generated by its parent $\mathsf{ID}' = (\mathsf{ID}_0, \ldots, ID_{k-1})$.

Hierarchical Identity-Based Chameleon Hash(3)

- The security requirements of HIBCH include:
  - *resistance to collision forgery under active attacks*: unfeasibility to extract the trapdoor information of ID, even if all other users are compromised and collude.
  - *semantic security*: the hash value hides information about the message.
  - *forgery indistinguishability*: unfeasibility to decide whether (m, r, h) is a forgery or not.

Construction(1)

- Overview:
  - The private key of the HIBCH scheme is a random multivariate polynomial $f(x_0, \ldots, x_\ell)$
    - the degree of x_i is a threshold parameter t,
    - ℓ is the maximum depth of the hierarchy.
  - The trapdoor information of an identity $\mathsf{ID} = (\mathsf{ID}_0, \ldots, \mathsf{ID}_k)$ is $f(\mathsf{ID}_0, \ldots, \mathsf{ID}_{k-1}, \mathsf{ID}_k, x_{k+1}, \ldots, x_\ell)$, which can be derived from his parent's trapdoor information $f(\mathsf{ID}_0, \ldots, \mathsf{ID}_{k-1}, x_k, \ldots, x_\ell)$.

Construction(2)

- Setup:
  $\langle p, \mathbb{G}, \mathbb{G}_T, e \rangle$, $g \in \mathbb{G}$, $H : \{0,1\}^* \to \mathbb{G}$, an *idle identity* $\overline{\mathsf{ID}} \in \mathbb{Z}_p$, choose a random polynomial (over $\mathbb{Z}_p$) $f(x_0, \ldots, x_\ell) = a_{t,t,\ldots,t} x_0^t x_2^t \cdots x_\ell^t + a_{t-1,t,\ldots,t} x_1^{t-1} x_2^t \cdots x_\ell^t + \cdots + a_{0,0,\ldots,0}$, where the degree of x_i is a threshold parameter t.

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

  and $sk = f(x_0, \ldots, x_\ell)$.

Construction(3)

Given $\mathsf{ID} = (\mathsf{ID}_0, \ldots, \mathsf{ID}_k)$

- Its public key is $pk_{\mathsf{ID}} = g^{f(\mathsf{ID}_0,\ldots,\mathsf{ID}_k,\overline{\mathsf{ID}},\ldots,\overline{\mathsf{ID}})}$, which can be computed by everyone using $g^{a_{t,\ldots,t}}, \ldots, g^{a_{0,\ldots,0}}$ and $\mathsf{ID}_0, \ldots, \mathsf{ID}_k$.
- Extract:
  and the trapdoor information
  $sk_{\mathsf{ID}_{|k-1}} = f(\mathsf{ID}_0, \ldots, \mathsf{ID}_{k-1}, x_k, \ldots, x_\ell)$ of the parent identity
  $\mathsf{ID}_{|k-1} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_{k-1})$,
  compute
  $$sk_{\mathsf{ID}} = f(\mathsf{ID}_{|k-1}, \mathsf{ID}_k, x_{k+1}, \ldots, x_\ell)$$

Construction(4)

- Hash Given pk, $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_k)$, m, choose a randomness $r \in \mathbb{G}$ uniformly, compute

$$h = e(r, g) \cdot e(H(m), pk_{\mathsf{ID}})$$

- Forge Given $\mathsf{ID} = (\mathsf{ID}_0, \ldots, \mathsf{ID}_k)$, $sk_{\mathsf{ID}} = f(\mathsf{ID}_0, \ldots, \mathsf{ID}_k, x_{k+1}, \ldots, x_\ell)$, the hash value h on (m, r), and a new message m', compute $f(\mathsf{ID}_0, \ldots, \mathsf{ID}_k, \overline{\mathsf{ID}}, \ldots, \overline{\mathsf{ID}})$ using sk_{ID}, compute

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

and output the randomness R' satisfying

$$h = \mathsf{Hash}(pk, \mathsf{ID}, m, r) = \mathsf{Hash}(pk, \mathsf{ID}, m', r')$$

Construction(5)

- It is *semantic secure* and *forgery indistinguishable*.
- **weaker resistance.** It achieves *t-threshold resistance to collision forgery under active attacks*, where the adversary compromises at most t users at each depth of the hierarchy.

Message Structure of HTSS

- Each message is partitioned into L blocks $m = m_1 \| \cdots \| m_L$.
- We define a hierarchical sanitizable description $\texttt{ADM}$ of m using a tree.
  - Each node of the tree represents an identity in the hierarchy.
  - If a leaf node is labeled $i \in [1, L]$, it means that m_i is sanitizable by the corresponding identity.

Example of ADM

A message $m = m_1||m_2||m_3||m_4||m_5||m_6||m_7||m_8$ where m_1, m_4, m_5, m_8 are sanitizable by $(\mathsf{ID}_0, \mathsf{ID}_1^1, \mathsf{ID}_2^1, \mathsf{ID}_3^1)$, $(\mathsf{ID}_0, \mathsf{ID}_1^1, \mathsf{ID}_2^2, \mathsf{ID}_3^2)$, $(\mathsf{ID}_0, \mathsf{ID}_1^1, \mathsf{ID}_2^1, \mathsf{ID}_3^3)$, and $(\mathsf{ID}_0, \mathsf{ID}_1^2, \mathsf{ID}_2^3)$, respectively.

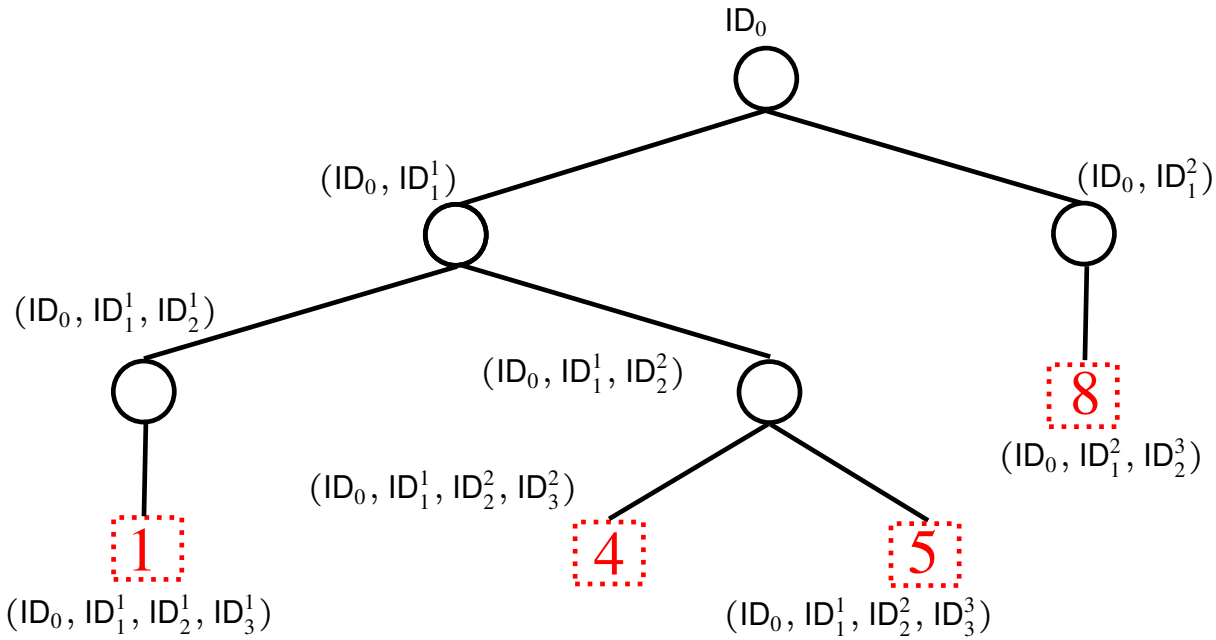

Figure: An example of hierarchical sanitizable description

HTSS from HIBCH(1)

- Overview:
  - to sign a message $m = m_1 \| \cdots \| m_L$, the signer determines the set of indices $I \subseteq [1, L]$ for sanitizable message blocks, and a description $\mathtt{ADM}$ specifying $\mathsf{ID}^{(i)}$ associated with the sanitizable m_i block, $i \in I$.
  - the signer sets $\tilde{m} = \tilde{m}_1 \| \cdots \| \tilde{m}_L$, where $\tilde{m}_i = m_i$ if $i \notin I$ and otherwise, $\tilde{m}_i = h_i = \mathsf{HIBCH.Hash}(pk, \mathsf{ID}^{(i)}, m_i, r_i)$.
  - Then, the signer signs the message $\tilde{m}$ 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.

HTSS from HIBCH(2)

- Given a conventional signature scheme $\Sigma = (\Sigma.\mathsf{KeyGen}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Verify})$ and an HIBCH scheme $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Extract}, \Pi.\mathsf{Hash}, \Pi.\mathsf{Forge})$.
- KeyGen:

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

Then, set the public key $pk = (pk_\Sigma, pk_\Pi)$ and the private key $sk = (sk_\Sigma, sk_\Pi)$.

HTSS from HIBCH(3)

- Sign: Given a message $m = m_1 \| \cdots \| m_L$, a hierarchial sanitizable description ADM and $sk = (sk_\Sigma, sk_\Pi)$, extract a set of indices $I \subseteq [1, L]$ that are sanitizable from ADM, proceed 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 ADM labeled by the block index i, it chooses a randomness r_i uniformly, and computes $\tilde{m}_i = \Pi.\mathsf{Hash}(pk_\Pi, \mathsf{ID}^{(i)}, m_i, r_i)$. 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.

HTSS from HIBCH(6)

- 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 ADM, it proceeds as follows.
  1. It extracts a set of indices $I \subseteq [1, L]$ that are sanitizable from 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 ADM labeled by the block index i, it computes $\tilde{m}_i = \Pi.\mathsf{Hash}(pk_\Pi, \mathsf{ID}^{(i)}, m_i, r_i)$.
  4. It sets $\tilde{m} = \tilde{m}_1 \| \cdots \| \tilde{m}_L$ and outputs $\Sigma.\mathsf{Verify}(pk_\Sigma, \tilde{m}, \tilde{\sigma})$.

HTSS from HIBCH(4)

- Trapdoor: same as Π.Extract.

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

and output the trapdoor sk_{ID} associated with ID.
Note that, if $k = 1$, the trapdoor $sk_{\mathsf{ID}_{|k-1}}$ is sk_{Π}.

HTSS from HIBCH(5)

- Sanitize: Given a message $m = m_1 \| \cdots \| m_L$, σ, $\mathtt{ADM}$, ID, sk_{ID}, a new message $m' = m'_1 \| \cdots \| m'_L$, it proceeds as follows.
  1. Let $I' = \{i \in [1, L] | m_i \neq m'_i\}$. Extract a set of indices $I \subseteq [1, L]$ that are sanitizable from $\mathtt{ADM}$.
  2. Retrieve $\{r_i | i \in I\}$ from the signature $\sigma = \tilde{\sigma} \| r$.
  3. For all $i \in I'$, it computes $h_i \leftarrow \Pi.\mathsf{Hash}(pk_\Pi, \mathsf{ID}^{(i)}, m_i, r_i)$ and
  $$r'_i \leftarrow \Pi.\mathsf{Forge}(sk_{\mathsf{ID}^{(i)}}, \mathsf{ID}^{(i)}, m_i, r_i, h_i, m'_i).$$
  4. For all $i \in I \backslash I'$, it sets $r'_i = r_i$. Let r' be the concatenation of all random values r'_i, $i \in I$.
  5. It sets $\sigma' = \tilde{\sigma} \| r'$ and outputs the new signature σ' on m'.

Conclusions

- Introduced the notion of HIBCH and HTSS.
- Showed HTSS from HIBCH and key-exposure free IBCH from HIBCH.
- Proposed the construction of HIBCH.
- Future direction: constructing fully collusion-resistant HIBCH.

Thanks!

Hierarchical Identity-Based Chameleon Hash(2)

- Formally, four algorithms:
  - $(pk, sk) \leftarrow \mathsf{Setup}(\lambda, \ell)$, where ℓ is the max. hierarchy depth.
  - $sk_{\mathsf{ID}} \leftarrow \mathsf{Extract}(sk_{\mathsf{ID}_{|k-1}}, \mathsf{ID})$, where $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_k)$, $\mathsf{ID}_{|k-1} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_{k-1})$ and $sk_{\mathsf{ID}_{|k-1}}$ is the trapdoor information of the identity $\mathsf{ID}_{|k-1}$. If $k = 1$, $sk_{\mathsf{ID}_{|k-1}} = sk$. We also denote this algorithm as

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

    where ID' is a descendant of ID.
  - $h \leftarrow \mathsf{Hash}(pk, \mathsf{ID}, m, r)$.
  - $r' \leftarrow \mathsf{Forge}(sk_{\mathsf{ID}}, \mathsf{ID}, m, r, h, m')$.

For correctness, it requires that

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

Key-Exposure Free IBCH from HIBCH (1)

- Informally, key-exposure free HIBCH requires that given many collisions, an adversary be not able to output a new collusion:
- Given an HIBCH scheme $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Extract}, \Pi.\mathsf{Hash}, \Pi.\mathsf{Forge})$.
- Setup: 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.

Key-Exposure Free IBCH from HIBCH (2)

- 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.

Key-Exposure Free IBCH from HIBCH (3)

- **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

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

Then it runs

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

and outputs r'.