Computers and Electrical Engineering 37 (2011) 614–623

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Discrete logarithm based chameleon hashing and signatures without key exposure ☆

Xiaofeng Chen [a,*], Fangguo Zhang [b], Haibo Tian [b], Baodian Wei [b], Kwangjo Kim [c]

[a] Key Laboratory of Computer Networks and Information Security, Ministry of Education, Xidian University, Xi'an 710071, PR China
[b] School of Information Science and Technology, Sun Yat-sen University, Guangzhou 510275, PR China
[c] Computer Science Department, KAIST, Taejon 305-714, South Korea

ARTICLE INFO

Article history:
Received 8 April 2010
Received in revised form 18 March 2011
Accepted 28 March 2011
Available online 6 May 2011

ABSTRACT

Chameleon signatures simultaneously provide the properties of non-repudiation and non-transferability for the signed message. However, the initial constructions of chameleon signatures suffer from the key exposure problem of chameleon hashing. This creates a strong disincentive for the recipient to compute hash collisions, partially undermining the concept of non-transferability. Recently, some constructions of discrete logarithm based chameleon hashing and signatures without key exposure are presented, while in the setting of gap Diffie–Hellman groups with pairings.

In this paper, we propose the first key-exposure free chameleon hash and signature scheme based on discrete logarithm systems, without using the gap Diffie–Hellman groups. This provides more flexible constructions of efficient key-exposure free chameleon hash and signature schemes. Moreover, one distinguishing advantage of the resulting chameleon signature scheme is that the property of "message hiding" or "message recovery" can be achieved freely by the signer, i.e., the signer can efficiently prove which message was the original one if he desires.

1. Introduction

Chameleon signatures, introduced by Krawczyk and Rabin [1], are based on well established hash-and-sign paradigm, where a chameleon hash function is used to compute the cryptographic message digest. A chameleon hash function is a trapdoor one-way hash function, which prevents everyone except the holder of the trapdoor information from computing the collisions for a randomly given input. Chameleon signatures simultaneously provide non-repudiation and non-transferability for the signed message as undeniable signatures [2–6] do, but the former allows for simpler and more efficient realization than the latter. In particular, chameleon signatures are non-interactive and less complicated. More precisely, the signer can generate the chameleon signature without interacting with the designated recipient, and the recipient will be able to verify the signature without the collaboration of the signer. On the other hand, if presented with a forged signature, the signer can deny its validity by only revealing some certain values. That is, the forged-signature denial protocol is also non-interactive. Besides, since the chameleon signatures are based on well established hash-and-sign paradigm, it provides more generic and flexible constructions.

One limitation of the original chameleon signature scheme is that signature forgery (i.e., collision computation) results in the signer recovering the recipient's trapdoor information, i.e., the private key [7]. The signer then can use this information to deny other signatures given to the recipient. In the worst case, the signer could collaborate with other individuals to

☆ Reviews processed and approved for publication by Editor-in-Chief Dr. Manu Malek.

* Corresponding author.
E-mail address: xfchen@xidian.edu.cn (X. Chen).

0045-7906/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compeleceng.2011.03.011

invalidate any signatures which were designated to be verified by the same public key. This will create a strong disincentive for the recipient to compute the hash collisions and thus weakens the property of non-transferability.

Ateniese and de Medeiros [7] firstly addressed the key exposure problem of chameleon hashing and introduced the idea of identity-based chameleon hashing to solve this problem.[1] Due to the distinguishing property of identity-based system, the signer can sign a message to an intended recipient, without having to first retrieve the recipient's certificate. Moreover, the signer uses a different public key (corresponding a different private key) for each transaction with a recipient, so that signature forgery only results in the signer recovering the trapdoor information associated to a single transaction. Therefore, the signer will not be capable of denying signatures on any message in other transactions. We argue that this idea only provides a partial solution for the problem of key exposure since the recipient's public key is changed for each transaction.[2]

Chen et al. [11] proposed the first full construction of a key-exposure free chameleon hash function in the gap Diffie–Hellman (GDH) groups with bilinear pairings. Ateniese and de Medeiros [12] then presented three key-exposure free chameleon hash schemes, two based on the RSA assumption (the first constructions without using pairings), as well as a new construction based on pairings. Recently, Gao et al. [13] claimed to present a key-exposure free chameleon hash scheme based on the Schnorr signature. However, this scheme requires an interactive protocol between the signer and the recipient and thus does not meet the basic definition of chameleon hashing and signatures.

All of the existing discrete logarithm based chameleon hash schemes without key exposure [11,12] can only be constructed in the setting of GDH groups with pairings. Are there efficient (discrete-logarithm-based) constructions for key-exposure free chameleon hashing without using the GDH groups? To the best of our knowledge, there is no research work on this problem in the open literature.

Our contribution. In this paper, we propose two efficient constructions for discrete logarithm based chameleon hash schemes without key exposure. Our contribution is two folds:

(1) We proposed a new key-exposure free chameleon hash scheme in the GDH groups. Compared with the existing schemes in the GDH groups [11,12], the proposed chameleon hash scheme is not only based on the weaker assumption, but also more efficient in both hashing and collision computations.
(2) We propose the first discrete logarithm based key-exposure free chameleon hash scheme without using the GDH groups. One distinguishing advantage of the resulting chameleon signature scheme is that the property of "message hiding" or "message recovery" can be achieved freely by the signer.

Organization. The rest of the paper is organized as follows: Some preliminaries are given in Section 2. The definitions associated with chameleon hashing and signatures are introduced in Section 3. The proposed key exposure freeness chameleon hash and signature schemes in the GDH groups and non-GDH groups are given in Sections 4 and 5, respectively. Finally, conclusions will be made in Section 6.

2. Preliminaries

In this section, we first introduce some well-known number-theoretic problems in the discrete logarithm systems. We then present two proof systems for knowledge of discrete logarithms.

2.1. Number-theoretic problems

Let $\mathbb{G}$ be a cyclic multiplicative group generated by g with the prime order q. We introduce the following problems in $\mathbb{G}$.

- Discrete logarithm problem (DLP): Given two elements g and h, to find an integer $a \in \mathbb{Z}_q^*$, such that $h = g^a$ whenever such an integer exists.
- Computation Diffie–Hellman problem (CDHP): Given (g, g^a, g^b) for $a, b \in \mathbb{Z}_q^*$, to compute g^{ab}.
- Decision Diffie–Hellman problem (DDHP): Given (g, g^a, g^b, g^c) for $a, b, c \in \mathbb{Z}_q^*$, to decide whether $c \equiv ab \bmod q$.

It is proved that the CDHP and DDHP are not equivalent in the GDH groups. More precisely, we call $\mathbb{G}$ a GDH group if the DDHP can be solved in polynomial time but there is no polynomial time algorithm to solve the CDHP with non-negligible probability. Such groups can be found in supersingular elliptic curves or hyperelliptic curves over finite fields. For more details, see [14–16]. Moreover, we call $\langle g, g^a, g^b, g^c \rangle$ a valid Diffie–Hellman tuple if $c \equiv ab \bmod q$.

[1] Shamir and Tauman [8] firstly used the chameleon hash functions to design efficient generic on-line/off-line signature schemes. It also suffers from the key exposure problem of chameleon hashing. Chen et al. [9,10] firstly introduced a special double-trapdoor hash family to solve the problem in the on-line/off-line signatures.

[2] A trivial solution for the key exposure problem is that the signer changes his key pair frequently in the chameleon signature scheme. However, it is only meaningful in theoretical sense because the key distribution problem arises simultaneously.

2.2. Proofs of knowledge

A prover with possession a secret number $x \in \mathbb{Z}_q$ wants to show a verifier that $x = \log_g y$ without exposing x, this is named the proof of knowledge of a discrete logarithm.

This proof of knowledge is basically a Schnorr signature [17] on message (g, y): The prover chooses a random number $r \in_R \mathbb{Z}_q$, and then computes $c = H(g, y, g^r)$, and $s = r - cx \bmod q$, where $H:\{0,1\}^* \to \{0,1\}^k$ is a collision-resistant hash function. The verifier accepts the proof if and only if $c = H(g, y, g^s y^c)$.

Definition 1. A pair $(c, s) \in \{0,1\}^k \times \mathbb{Z}_q$ satisfying $c = H(g, h, g^s y^c)$ is a proof of knowledge of a discrete logarithm of the element y to the base g.

Similarly, we can define the proof of knowledge for the equality of two discrete logarithms: A prover with possession a secret number $x \in \mathbb{Z}_q$ wants to show that $x = \log_g u = \log_h v$ without exposing x.

Chaum and Pedersen [18] firstly proposed the proof as follows: The prover chooses a random number $r \in_R \mathbb{Z}_q$, and then computes $c = H(g, h, u, v, g^r, h^r)$, and $s = r - cx \bmod q$, where $H:\{0,1\}^* \to \{0,1\}^k$ is a collision-resistant hash function. The verifier accepts the proof if and only if $c = H(g, h, u, v, g^s u^c, h^s v^c)$.

Definition 2. A pair $(c, s) \in \{0,1\}^k \times \mathbb{Z}_q$ satisfying $c = H(g, h, u, v, g^s u^c, h^s v^c)$ is a proof of knowledge for the equality of two discrete logarithms of elements u, v with respect to the base g, h.

Trivially, the verifier can efficiently decide whether $\langle g, u, h, v \rangle$ is a valid Diffie–Hellman tuple with the pair (c, s).

3. Definitions

In this section, we introduce the definitions and properties of chameleon hashing and signatures [1,7,8].

3.1. Chameleon hashing

A chameleon hash function is a trapdoor collision-resistant hash function, which is associated with a trapdoor/hash key pair (TK, HK). Anyone who knows the public key HK can efficiently compute the hash value for each input. However, there exists no efficient algorithm for anyone except the holder of the secret key TK, to find collisions for every given input. In the following, we present a formal definition of a chameleon hash scheme.

Definition 3. A chameleon hash scheme consists of four efficient algorithms $(\mathcal{PG}, \mathcal{KG}, \mathcal{H}, \mathcal{F})$:

- **System parameters generation** $\mathcal{PG}$: A probabilistic polynomial-time algorithm that, on input a security parameter k, outputs the system parameters SP.
- **Key generation** $\mathcal{KG}$: A probabilistic polynomial-time algorithm that, on input the system parameters SP, outputs a trapdoor/hash key pair (TK, HK).
- **Hashing computation** $\mathcal{H}$: A probabilistic polynomial-time algorithm that, on input the hash key HK, a customized identity I,[3] a message m, and a random string r, outputs the hashed value $h = \mathrm{Hash}(I, m, r)$. Note that h does not depend on TK.
- **Collision computation** $\mathcal{F}$: A deterministic polynomial-time algorithm that, on input the trapdoor key TK, a message m, a random string r, and another message $m' \neq m$, outputs a string $r' = \mathcal{F}(h, x, I, m, r, m')$ such that

$$h = \mathrm{Hash}(I, m', r') = \mathrm{Hash}(I, m, r).$$

Moreover, if r is uniformly distributed in a finite space $\mathcal{R}$, then the distribution of r' is computationally indistinguishable from uniform in $\mathcal{R}$.

A secure chameleon hashing scheme satisfies the following properties:

- **Collision resistance**: Without the knowledge of trapdoor key TK, there exists no efficient algorithm that, on input a message m, a random string r, and another message m', outputs a string r' that satisfy $\mathrm{Hash}(I, m', r') = \mathrm{Hash}(I, m, r)$, with non-negligible probability.
- **Semantic security**: Let $H[X]$ denote the entropy of a random variable X, and $H[X|Y]$ the entropy of the variable X given the value of a random function Y of X. Semantic security is the statement that the conditional entropy $H[m|h]$ of the message given its chameleon hash value h equals the total entropy $H[m]$ of the message space.
- **Key exposure freeness**: If a recipient has never computed a collision under I, then there is no efficient algorithm for an adversary to find a collision for a given chameleon hash value $\mathrm{Hash}(I, m, r)$. This must remain true even if the adversary has oracle access to $\mathcal{F}$ and is allowed to make polynomially many queries on triples (I_j, m_j, r_j) of his choice, except that I_j is not allowed to equal the challenge I.

[3] A customized identity is actually a label for each transaction [7,12].

3.2. Chameleon signatures

A chameleon signature is generated by digitally signing a chameleon hash value of the message. More precisely, we have the following definition:

Definition 4. A chameleon signature scheme consists of the following efficient algorithms and a specific denial protocol:

- **System parameters generation** $\mathcal{PG}$: A probabilistic polynomial-time algorithm that, on input a security parameter k, outputs the system parameters SP.
- **Key generation** $\mathcal{KG}$: A probabilistic polynomial-time algorithm that, on input the system parameters SP, outputs a trapdoor/hash key pair (TK, HK) and a signing/verification key pair (sk, vk).
- **Signature generation** $\mathcal{SG}$: A probabilistic polynomial-time algorithm that, on input the hash key HK, the signing key sk, a customized identity I, a message m, and a random string r, outputs a signature σ on the chameleon hash value $h = \text{Hash}(I, m, r)$.
- **Signature verification** $\mathcal{SV}$: A deterministic polynomial-time algorithm that, on input the hash key HK, the verification key vk, a customized identity I, a message m, a random string r, and a signature σ, outputs a verification decision $b \in \{0,1\}$.
- **Denial protocol** $\mathcal{DP}$: A non-interactive protocol between the signer and the judge. Given a chameleon signature (σ, r) on the message m, the signer provides the judge a valid collision (m', r') and some auxiliary information Σ. If and only if $m \neq m'$ and Σ is valid, the judge claims that the signature σ on the message m is a forgery.

A secure chameleon signature scheme should satisfy the properties [1,7,11]:

- **Unforgeability**: No party can produce a valid chameleon signature not previously generated by the signer. Also, the recipient can only produce a forgery of a chameleon signature previously generated by the signer.
- **Non-transferability**: The recipient cannot convince a third party that the signer indeed generated a signature on a certain message, thus the signature is not universal verifiable.
- **Non-repudiation**: The signer cannot deny legitimate signature claims.
- **Deniability**: The signer can deny a forgery of the signature.
- **Message hiding**: In case of a dispute, the signer can compute a new collision to deny the forgery and thus the original message is never revealed.
- **Message recovery** (or **Convertibility**): A variant of the chameleon signature can be transformed into a regular signature by the signer. That is, the signer is also able to prove which message is the original one in case of forgery.

4. Constructions in the GDH groups

In this section, we present an efficient construction of chameleon hashing without key exposure in the GDH groups. As pointed out by Ateniese and de Medeiros [12], the double-trapdoor mechanism is a necessary condition for the construction of key-exposure free chameleon hashing. There are two consecutive trapdoors in our proposed chameleon hash scheme: One is the master trapdoor key x of the user, the other is the ephemeral trapdoor h^x for each transaction with the customized identity I. Given a collision of the chameleon hash function, only the ephemeral trapdoor h^x is revealed, but the master trapdoor x still remains secret.

4.1. The proposed chameleon hash scheme

- **System parameters generation** $\mathcal{PG}$: Let $\mathbb{G}$ be a GDH group generated by g, whose order is a prime q. Let $H : \{0,1\}^* \to \mathbb{G}^*$ be a full-domain collision-resistant hash function. The system parameters are $SP = \{\mathbb{G}, q, g, H\}$.
- **Key generation** $\mathcal{KG}$: Any user randomly chooses an integer $x \in_R \mathbb{Z}_q^*$ as his trapdoor key, and publishes his hash key $y = g^x$. The validity of y can be ensured by a certificate issued by a trusted certification authority.
- **Hashing computation** $\mathcal{H}$: On input the hash key y, a customized identity I, let $h = H(y, I)$. Chooses a random integer $a \in_R \mathbb{Z}_q^*$, and computes $r = (g^a, y^a)$. Our proposed chameleon hash function is defined as

$$\mathcal{H} = \text{Hash}(I, m, r) = g^a h^m.$$

- **Collision computation** $\mathcal{F}$: For any valid hash value $\mathcal{H}$, the algorithm $\mathcal{F}$ can be used to compute a hash collision with the trapdoor key x as follows:

$$\mathcal{F}(\mathcal{H}, x, I, m, r, m') = r' = (g^{a'}, y^{a'}),$$

where $g^{a'} = g^a h^{m-m'}$ and $y^{a'} = y^a h^{x(m-m')}$.

Note that

$$\mathrm{Hash}(I, m', r') = g^{a'} h^{m'} = g^{a} h^{m-m'} h^{m'} = g^{a} h^{m} = \mathrm{Hash}(I, m, r)$$

and $\langle g, y, g^{a'}, y^{a'} \rangle$ is a valid Diffie–Hellman tuple. Therefore, the forgery is successful. Moreover, if r is uniformly distributed then the distribution of r' is computationally indistinguishable from uniform.

Theorem 1. *The proposed chameleon hash scheme is collision resistance under the assumption that the CDHP in $\mathbb{G}$ is intractable.*

Proof. We can prove this theorem by contradiction. Assume that there exists a polynomial time algorithm $\mathcal{A}$, with a non-negligible probability, that outputs two pairs (m,r) and (m', r') which satisfy Hash $(I, m', r') = \mathrm{Hash}(I, m, r)$, i.e., $g^{a'} h^{m'} = g^{a} h^{m}$, we can compute $h^{x} = (y^{a'}/y^{a})^{(m-m')^{-1}}$ efficiently. Note that h^x is a GDH signature on message I [15], which is proved secure against existential forgery on adaptively chosen message under the assumption that the CDHP in $\mathbb{G}$ is intractable. Therefore, the proposed chameleon hash scheme is collision resistance under the assumption that the CDHP in $\mathbb{G}$ is intractable. □

Theorem 2. *The proposed chameleon hash scheme is semantically secure.*

Proof. Given a customized identity I, there is a one-to-one correspondence between the hash value $\mathcal{H}$ and the string r for each message m. Therefore, the conditional probability $\mu(m|\mathcal{H}) = \mu(m|r)$. Note that m and r are independent variables, the equation $\mu(m|\mathcal{H}) = \mu(m)$ holds. Then, we can prove that the conditional entropy $H[m|\mathcal{H}]$ equals the entropy $H[m]$ as follows:

$$H[m|\mathcal{H}] = -\sum_{m}\sum_{\mathcal{H}} \mu(m,\mathcal{H}) \log(\mu(m|\mathcal{H})) = -\sum_{m}\sum_{\mathcal{H}} \mu(m,\mathcal{H}) \log(\mu(m)) = -\sum_{m} \mu(m) \log(\mu(m)) = H[m].$$

So, the proposed chameleon hash scheme is semantically secure. □

Theorem 3. *The proposed chameleon hash scheme is key-exposure free.*

Proof. Even if the adversary has oracle access to $\mathcal{F}$ and is allowed to make polynomially many queries on triples $(I_j, m_j, g^{a_j}, y^{a_j})$ of his choice, there is no efficient algorithm for him to find a collision of the hash value $\mathcal{H} = \mathrm{Hash}(I, m, g^{a}, y^{a})$ where $I \neq I_j$. Note that h^x is a GDH signature on message I [15], and computing collisions is equivalent to breaking the signature scheme. However, the GDH signature scheme is proved to be secure against existential forgery on adaptive chosen-message attacks in the random oracle model. In other words, even if the adversary has obtained polynomially many GDH signatures h_j^x on message I_j, he cannot forge a signature h^x on message $I \neq I_j$. □

4.2. The proposed chameleon signature scheme

There are two users, a signer S and a recipient R, in our signature scheme. When a dispute occurs, a judge J can involve in the scheme.

- **System parameters generation** $\mathcal{PG}$: Let $\mathbb{G}$ be a GDH group generated by g, whose order is a prime q. Let $H : \{0,1\}^* \to \mathbb{G}^*$ be a full-domain collision-resistant hash function. The system parameters are $SP = \{\mathbb{G}, q, g, H\}$.
- **Key generation** $\mathcal{KG}$: S randomly chooses an integer $x_S \in_R \mathbb{Z}_q^*$ as his signing key, and publishes his verification key $y_S = g^{x_S}$. Similarly, R randomly chooses an integer $x_R \in_R \mathbb{Z}_q^*$ as his trapdoor key, and publishes his hash key $y_R = g^{x_R}$.
- **Signature generation** $\mathcal{SG}$: Suppose the message to be signed is m. S randomly chooses an integer $a \in_R \mathbb{Z}_q^*$, and computes the chameleon hash value $\mathcal{H} = g^{a} h^{m}$, where $h = H(y_R, I)$ and I is a customized identity. Assume SIGN is any secure signature scheme. The signature σ for message m consists of $(m, g^{a}, y_R^{a}, \mathrm{SIGN}_{x_S}(\mathcal{H}))$.
- **Signature verification** $\mathcal{SV}$: Given a signature σ, R first verifies whether the equation $(g^{a})^{x_R} = y_R^{a}$ holds.[4] If the verification fails, he rejects the signature; else, he computes the chameleon hash value $\mathcal{H} = g^{a} h^{m}$ and verifies the validity of $\mathrm{SIGN}_{x_S}(\mathcal{H})$ with the verification key y_S.
- **Denial protocol** $\mathcal{DP}$: When a dispute occurs, i.e., R provides a signature $\sigma = (m^*, g^{a^*}, y_R^{a^*}, \mathrm{SIGN}_{x_S}(\mathcal{H}))$ to J. If either $\langle g, y_R, g^{a^*}, y_R^{a^*} \rangle$ is not a valid Diffie–Hellman tuple or $\mathrm{SIGN}_{x_S}(\mathcal{H})$ is invalid, J rejects it. Otherwise, J summons S to accept/deny the claim. If S wants to accept the signature, he just confirms to J this fact. Otherwise, he provides a collision for the chameleon hash function as follows:

[4] If the equation $(g^{a})^{x_R} = y_R^{a}$ holds, then R can be convinced that $\langle g, y_R, g^{a}, y_R^{a} \rangle$ is a valid Diffie–Hellman tuple.

- If S wants to achieve the property of "message hiding", he provides J a collision $(m', g^{a'}, y_R^{a'})$. If and only if $m^* \neq m', \langle g, y_R, g^{a'}, y_R^{a'} \rangle$ is a valid Diffie–Hellman tuple, and $\mathcal{H} = g^{a'} h^{m'}$, then J can be convinced that R forged the signature on message m^*.
- If S wants to achieve the property of "message recovery", he provides the tuple (m, g^a, y_R^a, Σ) as the collision, where Σ is a non-interactive proof of knowledge of the discrete logarithm $a = \log_g g^a$. If and only if $m^* \neq m$, $\langle g, y_R, g^a, y_R^a \rangle$ is a valid Diffie–Hellman tuple, $\mathcal{H} = g^a h^m$, and Σ is valid, then J can be convinced that R forged the signature on message m^* and S only generated a valid signature on message m.

Different from the basic chameleon signature schemes [1,7], the proposed chameleon signature scheme has the following distinguishing advantages:

- In the previous chameleon signature schemes, the customized identity I and the identity of the recipient ID_R must be explicitly committed to the signature. While in our scheme, this is unnecessary since no one knows the discrete logarithm of the element h to the base g.
- Another distinguishing advantage of our scheme is that the signer can efficiently prove which message was the original one if he desires. This is due to the following observations: Firstly, no one can provide a proof of knowledge of the discrete logarithm $a' = \log_g g^{a'}$ for any collision $g^{a'} = g^a h^{m-m'}$; Secondly, only S can provide a proof of knowledge of the discrete logarithm $a = \log_g g^a$ for the original input g^a.

On the other hand, the enhanced schemes [1,7] can be converted into universally verifiable instances. The trick is that the signer encrypts the message using a semantically secure probabilistic encryption scheme ENC and then includes the ciphertext in the signature. However, as noted in [7], this solution does not provide the recipient with a mechanism for adjudicated convertibility, because the recipient has no guarantee that the signer has encrypted the correct information during the signing step.

4.3. Security analysis

Theorem 4. *The proposed chameleon signature scheme satisfies the properties of unforgeability, non-transferability, non-repudiation, deniability, message hiding, and key exposure freeness.*

Proof. We prove the proposed chameleon signature scheme satisfies the above properties one by one.

- **Unforgeability**: No third party can produce a valid chameleon signature which has not been previously generated by the signer, as this requires either to break the underlying signature scheme SIGN, or find a valid collision of the chameleon hash function $\mathcal{H}$. Also, it is trivial that the recipient can only produce a forgery of a chameleon signature previously generated by the signer. However, it is meaningless since the judge can detect this forgery after the signer provides a different collision.
- **Non-transferability**: Note that the semantic security of a chameleon hashing scheme implies the non-transferability of the corresponding chameleon signature scheme [7]. Therefore, the recipient cannot transfer a signature of the signer to convince any third party.
- **Non-repudiation**: Given a valid signature $\sigma = (m, g^a, y_R^a, \mathrm{SIGN}_{x_S}(\mathcal{H}))$, the signer cannot generate a valid hash collision $(m', g^{a'}, y_R^{a'})$ which satisfies $\mathcal{H} = \mathrm{Hash}(I, m', g^{a'}, y_R^{a'})$ and $m \neq m'$ because it is equivalent to computing the CDHP in $\mathbb{G}$.
- **Deniability**: It is ensured by the denial protocol.
- **Message hiding**: Given a collision (m, g^a, y_R^a) and $(m^*, g^{a^*}, y_R^{a^*})$, though the trapdoor key x is never divulged, the signer can compute the *ephemeral* trapdoor key h^x. Then the signer can provide any other collision $(m', g^{a'}, y_R^{a'})$ to ensure the confidentiality of the original message m, where $g^{a'} = g^a h^{m-m'}$, $y_R^{a'} = y_R^a (h^x)^{m-m'}$.
- **Message recovery**: Note that (only) S can provide a proof of knowledge of the discrete logarithm $a = \log_g g^a$ (only) for the original input g^a. Therefore, any verifier can be convinced that the original message to be signed is m. □

4.4. Comparison

Compared with the existing two key-exposure free chameleon hash schemes in the GDH groups [11,12], the proposed chameleon hash scheme is a little more efficient in both hashing computation and collision computation. Moreover, the security of the scheme [12] is equivalent to the q-Strong Diffie–Hellman problem (q-SDHP), while the security of our proposed scheme is equivalent to the CDHP, which is harder than the q-SDHP for any q.

In the proposed chameleon signature scheme, both the signature verification and the denial protocol are non-interactive, so it is more efficient and simple than undeniable signature schemes. Moreover, compared with two previous chameleon signature schemes in the GDH groups [11,12], our signature scheme provides more efficient and explicit convertibility.

Table 1
Comparison with two previous chameleon hash schemes.

	Scheme [18]	Scheme [17]	Our scheme
Mathematical assumption	q-SDHP	CDHP	CDHP
Hashing computation	$4M + 2m$	$3M + 2m$	$3M + 1m$
Collision computation	$2M + 2m + 1I$	$2M + 2m + 1I$	$2M + 1m$

Table 2
Comparison with two previous chameleon signature schemes.

	Scheme [18]	Scheme [17]	Our scheme
Signature generation	$4M + 2m + 1C(S)$	$3M + 2m + 1C(S)$	$3M + 1m + 1C(S)$
Signature verification	$2M + 1m + 1C(V)$	$2M + 2m + 1C(V)$	$2M + 1m + 1C(V)$
Denial protocol (message hiding)	$2M + 3m + 1I$	$2M + 3m + 1I$	$2M + 3m + 1I$
Denial protocol (message recovery)	$1C(E)$	$1C(E)$	$1M$

Tables 1 and 2 present the comparison between our scheme and two previous schemes. We denote by M the exponentiation in $\mathbb{G}$, by m the multiplication in $\mathbb{G}$, and by I the inversion in $\mathbb{G}$. We also denote by $C(S)$, $C(V)$, and $C(E)$ the computation cost of signing, verifying in scheme SIGN and encrypting in scheme ENC, respectively. We omit other operations such as hashing and the multiplication in $\mathbb{Z}_q$ in all schemes.

5. Constructions in the non-GDH groups

In this section, we propose a construction of key exposure freeness chameleon hashing in the non-GDH groups, e.g., the multiplicative group of finite fields.

5.1. Main idea

The chameleon hash scheme in the non-GDH groups is almost the same as the one in the GDH groups. The only difference is the way to verify the validity of a Diffie–Hellman tuple. Given the original input (g^a, y^a) in the GDH groups, anyone can easily check that $\langle g, y, g^a, y^a\rangle$ is a valid Diffie–Hellman tuple using the decisional Diffie–Hellman (DDH) oracle. While in the non-GDH groups, no one is allowed to access the DDH oracle. Trivially, the holder of the trapdoor key x can verify the validity of the Diffie–Hellman tuple $\langle g, y, g^a, y^a\rangle$ as follows: The holder can check whether the equation $(g^a)^x = y^a$ holds using the trapdoor key x. However, any third party without knowing x cannot verify the validity of (g^a,y^a) with the same method.

As we mentioned before, the proof of knowledge for the equality of two discrete logarithms can substitute the DDH oracle. Therefore, the holder with trapdoor key x could provide such a knowledge proof to convince any third party of the fact. We explain it in more details as below.

Note that if (g, g^u, g^v, g^{uv}) is a valid Diffie–Hellman tuple, then (g, g^v, g^u, g^{uv}) is also a valid Diffie–Hellman tuple, vice versa. That is, there are two different ways (based on the knowledge u or v, respectively) to prove that (g, g^u, g^v, g^{uv}) is a valid Diffie–Hellman tuple when using the proof of knowledge for the equality of two discrete logarithms. This is the main trick to design key exposure freeness chameleon hash scheme in the non-GDH groups. Therefore, for any collision $(g^{a'}, y^{a'})$, the holder with knowledge x can provide a proof of knowledge for the equality of two discrete logarithms, i.e., $x = \log_g y = \log_{g^{a'}} y^{a'}$. In particular, it is also holds for the original input (g^a, y^a).[5] Moreover, we argue that it is **NOT** required to know the value a' or a in this knowledge proof.

On the other hand, only the signer (with knowledge a) can provide a proof of knowledge that $a = \log_g g^a = \log_y y^a$ for the original input (g^a, y^a), and no one can provide a proof of knowledge that $a' = \log_g g^{a'} = \log_y y^{a'}$ for any collision $(g^{a'}, y^{a'})$. This is the main trick to achieve the property of "message recovery" in the denial protocol of the proposed chameleon signature scheme. For more details, please refer to Section 5.3.

5.2. The proposed chameleon hash scheme

- **System parameters generation** $\mathcal{PG}$: Let $\mathbb{G}$ be a multiplicative group generated by g, whose order is a prime q. Let $H : \{0,1\}^* \to \mathbb{G}^*$ be a full-domain collision-resistant hash function. The system parameters are $SP = \{\mathbb{G}, q, g, H\}$.
- **Key generation** $\mathcal{KG}$: Any user randomly chooses an integer $x \in_R \mathbb{Z}_q^*$ as his trapdoor key, and publishes his hash key $y = g^x$. The validity of y can be ensured by a certificate issued by a trusted certification authority.

[5] This ensures the property of non-transferability in the resulting chameleon signature scheme.

- **Hashing computation** $\mathcal{H}$: On input the hash key y, a customized identity I, let $h = H(y, I)$. Chooses a random integer $a \in_R \mathbb{Z}_q^*$, and computes $r = (g^a, y^a)$. Our proposed chameleon hash function is defined as

$$\mathcal{H} = \mathrm{Hash}(I, m, r) = g^a h^m.$$

- **Collision computation** $\mathcal{F}$: For any valid hash value $\mathcal{H}$, the algorithm $\mathcal{F}$ can be used to compute a hash collision with the trapdoor key x as follows:

$$\mathcal{F}(\mathcal{H}, x, I, m, r, m') = r' = (g^{a'}, y^{a'}),$$

where $g^{a'} = g^a h^{m-m'}$ and $y^{a'} = y^a h^{x(m-m')}$.

Note that $\mathrm{Hash}(I, m', r') = \mathrm{Hash}(I, m, r)$. Also, for any collision r', the holder of the trapdoor key x can convince any third party that $\langle g, y, g^{a'}, y^{a'} \rangle$ is a valid Diffie–Hellman tuple, using a proof of knowledge for the equality of two discrete logarithms, i.e., $\log_g y = \log_{g^{a'}} y^{a'}$. In particular, it also holds for the original input (g^a, y^a). Therefore, the forgery is successful. Besides, if r is uniformly distributed then the distribution of r' is computationally indistinguishable from uniform.

Theorem 5. *The construction above is a secure chameleon hash scheme under the assumption that the CDHP in $\mathbb{G}$ is intractable.*

Proof. The proof for the properties of collision resistance and semantic security is the same as that of Theorem 1. In the following, we only focus on the key exposure freeness.

Note that even if the adversary has obtained polynomially many signatures h_j^x on message I_j, he can not forge a signature h^x on message $I \neq I_j$, otherwise the full domain hash (FDH) [19,20] variant of Chaum's undeniable signature scheme can be broken. However, Ogata et al. [21] showed that the unforgeability of the FDH variant of Chaum's scheme with non-interactive zero-knowledge proof confirmation and disavowal protocols is equivalent to the CDHP. Therefore, even if the adversary has oracle access to $\mathcal{F}$ and is allowed to make polynomially many queries on triples $(I_j, m_j, g^{a_j}, y^{a_j})$ of his choice, there is no efficient algorithm for him to find a collision of the hash value $\mathcal{H} = \mathrm{Hash}(I, m, g^a, y^a)$ where $I \neq I_j$. □

5.3. The proposed chameleon signature scheme

There are two users, a signer S and a recipient R, in our signature scheme. When dispute occurs, a judge J can involve in the scheme.

- **System parameters generation** $\mathcal{PG}$: Let $\mathbb{G}$ be a multiplicative group generated by g, whose order is a prime q. Let $H : \{0,1\}^* \to \mathbb{G}^*$ be a full-domain collision-resistant hash function. The system parameters are $SP = \{\mathbb{G}, q, g, H\}$.
- **Key generation** $\mathcal{KG}$: S randomly chooses an integer $x_S \in_R \mathbb{Z}_q^*$ as his signing key, and publishes his verification key $y_S = g^{x_S}$. Similarly, R randomly chooses an integer $x_R \in_R \mathbb{Z}_q^*$ as his trapdoor key, and publishes his hash key $y_R = g^{x_R}$.
- **Signature generation** $\mathcal{SG}$: Suppose the message to be signed is m. S randomly chooses an integer $a \in_R \mathbb{Z}_q^*$, and computes the chameleon hash value $\mathcal{H} = g^a h^m$, where $h = H(y_R, I)$ and I is a customized identity. Assume SIGN is any secure signature scheme. The signature σ for message m consists of $(m, g^a, y_R^a, \mathrm{SIGN}_{x_S}(\mathcal{H}))$.
- **Signature verification** $\mathcal{SV}$: Given a signature σ, R first verifies whether the equation $(g^a)^{x_R} = y_R^a$ holds. If the verification fails, he rejects the signature; else, he computes the chameleon hash value $\mathcal{H} = g^a h^m$ and verifies the validity of $\mathrm{SIGN}_{x_S}(\mathcal{H})$ with the verification key y_S.
- **Denial protocol** $\mathcal{DP}$: When a dispute occurs, i.e., R provides J a signature $\sigma = (m^*, g^{a^*}, y_R^{a^*}, \mathrm{SIGN}_{x_S}(\mathcal{H}))$ and a non-interactive proof of knowledge Π^* for the equality of two discrete logarithms that $x_R = \log_g y_R = \log_{g^{a^*}} y_R^{a^*}$. If either $\mathrm{SIGN}_{x_S}(\mathcal{H})$ or Π^* is invalid, J rejects it. Otherwise, J summons S to accept/deny the claim. If S wants to accept the signature, he just confirms to J this fact. Otherwise, he provides a collision for the chameleon hash function as follows:
  - If S wants to achieve the property of "message recovery", he provides J the tuple (m, g^a, y_R^a, Π) as a collision, where Π is a non-interactive proof of knowledge for the equality of two discrete logarithms that $\log_g g^a = \log_{y_R} y_R^a$. If and only if $m^* \neq m$, $\mathcal{H} = g^a h^m$, and Π is valid, then J can be convinced that R forged the signature on message m^* and S only generated a valid signature on message m.
  - If S wants to achieve the property of "message hiding", he provides J the tuple $(g^a, y_R^a, \Sigma, \Pi)$ as a collision, where Σ is a non-interactive proof of knowledge of a discrete logarithm that $m = \log_h \mathcal{H}/g^a$, and Π is a non-interactive proof of knowledge for the equality of two discrete logarithms that $\log_g g^a = \log_{y_R} y_R^a$. If and only if $g^{a^*} \neq g^a$, and Σ and Π are both valid, then J can be convinced that R forged the signature on message m^* and the original message m is still confidential.

Remark 1. For any collision $(g^{a^*}, y_R^{a^*})$, R can provide a proof of knowledge that $\log_g y_R = \log_{g^{a^*}} y_R^{a^*}$, which is also holds even when $a = a^*$. That is, the original input (g^a, y_R^a) is totally indistinguishable with any collision $(g^{a^*}, y_R^{a^*})$. Besides, only S can provide a proof of knowledge that $\log_g g^a = \log_{y_R} y_R^a$, and no one can provide a proof of knowledge that $\log_g g^{a^*} = \log_{y_R} y_R^{a^*}$ when $a \neq a^*$. Therefore, S can efficiently prove which message was the original one if he desires.

Remark 2. In the proposed chameleon signature scheme, both the signature verification and the denial protocol are non-interactive, so it is more efficient and simple than undeniable signature schemes.

Compared with our key-exposure free chameleon signature scheme based on GDH groups in Section 5.3, the proposed scheme is as efficient as in the signature generation and verification algorithms. While in the denial protocol, the proposed scheme requires a (very) little more computation and communication cost for the non-interactive proofs of knowledge. We argue that these proofs of knowledge requires at most 2 modular exponentiation operations and about $2q$ bits storage. Therefore, the proposed chameleon signature scheme is much efficient for the real applications.

5.4. Security analysis

Theorem 6. *The proposed chameleon signature scheme satisfies the properties of unforgeability, non-transferability, non-repudiation, deniability, message hiding, and key exposure freeness.*

Proof. We prove the proposed chameleon signature scheme satisfies the above properties one by one.

- **Unforgeability**: No third party can produce a valid chameleon signature which has not been previously generated by the signer, as this requires either to break the underlying signature scheme SIGN, or find a valid collision of the chameleon hash function $\mathcal{H}$. Also, it is trivial that the recipient can only produce a forgery of a chameleon signature previously generated by the signer. However, it is meaningless since the judge can detect this forgery after the signer provides a different collision.
- **Non-transferability**: The semantic security of the proposed chameleon hash scheme implies the non-transferability of the resulting chameleon signature scheme.
- **Non-repudiation**: Given a valid signature $\sigma = (m, g^a, y_R^a, \mathrm{SIGN}_{x_S}(\mathcal{H}))$, the signer cannot generate a valid hash collision $(m', g^{a'}, y_R^{a'})$ which satisfies $\mathcal{H} = \mathrm{Hash}(I, m', g^{a'}, y_R^{a'})$ and $m \neq m'$ because it is equivalent to computing the CDHP in $\mathbb{G}$.
- **Deniability**: It is ensured by the denial protocol.
- **Message hiding**: Since Σ is a proof of knowledge of a discrete logarithm that $m = \log_h \mathcal{H}/g^a$, the information for original signed message m is never revealed.
- **Message recovery**: Note that only S can provide a proof of knowledge that $\log_g g^a = \log_{y_R} y_R^a$, and no one can provide a proof of knowledge that $\log_g g^{a^*} = \log_{y_R} y_R^{a^*}$ when $a^* \neq a$. Therefore, any verifier can be convinced that the original message to be signed is m. □

6. Conclusions

We first propose a key-exposure free chameleon hash scheme based on discrete logarithm systems, without using the GDH groups. Moreover, one distinguishing advantage of the resulting chameleon signature scheme is that the property of "message hiding" or "message recovery" can be achieved freely by the signer. In addition, we prove that the proposed chameleon hash and signature schemes satisfy the desired security requirements.

Acknowledgement

The authors are grateful to the anonymous referees for their invaluable suggestions for improving this paper. This work is supported by National Natural Science Foundation of China (Nos. 60970144, 61003244, 61070168, and 60803135), and the Fundamental Research Funds for the Central Universities (No. K50510010003, 10lgpy31 and JY10000901034).

References

[1] Krawczyk H, Rabin T. Chameleon hashing and signatures. Proc NDSS 2000. Internet Society; 2000. p. 143–54.
[2] Chaum D, van Antwerpen H. Undeniable signatures. Advances in Cryptology-Crypto 1989, LNCS 435. Springer-Verlag; 1989. p. 212–6.
[3] Chaum D. Designated confirmer signatures. Advances in Cryptology-Eurocrypt 1994. LNCS 950. Springer-Verlag; 1994. p. 86–91.
[4] Galbraith S, Mao W. Invisibility and anonymity of undeniable and confirmer signatures. CT-RSA 2003. LNCS 2612. Springer-Verlag; 2003. p. 80–97.
[5] Jakobsson M, Sako K, Impagliazzo R. Designated verifier proofs and their applications. Advances in Cryptology-Eurocrypt 1996, LNCS 1070. Springer-Verlag; 1996. p. 143–54.
[6] Kang B, Boyd C, Dawson E. Identity-based strong designated verifier signature schemes: attacks and new construction. Comput Elect Eng 35(1). Elsevier; 2009. p. 49–53.
[7] Ateniese G, de Medeiros B. Identity-based chameleon hash and applications. FC 2004, LNCS 3110. Springer-Verlag; 2004. p. 164–80.
[8] Shamir A, Tauman Y. Improved online/offline signature schemes. Advances in Cryptology-Crypto 2001, LNCS 2139. Springer-Verlag; 2001. p. 355–67.
[9] Chen X, Zhang F, Susilo W, Mu Y. Efficient generic on-line/off-line signatures without key exposure. ACNS 2007, LNCS 4521. Springer-Verlag; 2007. p. 18–30.
[10] Chen X, Zhang F, Tian H, Wei B, Susilo W, Mu Y, et al. Efficient generic on-line/off-line (threshold) signatures without key exposure. Inform Sci 178(21). Elsevier; 2008. p. 4192–4203.
[11] Chen X, Zhang F, Kim K. Chameleon hashing without key exposure. ISC 2004, LNCS 3225. Springer-Verlag; 2004. p. 87–98.
[12] Ateniese G, de Medeiros B. On the key exposure problem in chameleon hashes. SCN 2004, LNCS 3352. Springer-Verlag; 2005. p. 165–79.
[13] Gao W, Li F, Wang X. Chameleon hash without key exposure based on Schnorr signature. Comput Stand Interf 31(2). Elsevier; 2009. p. 282–5.
[14] Barreto P, Kim H, Lynn B, Scott M. Efficient algorithms for pairing-based cryptosystems. Advances in Cryptology-Crypto 2002, LNCS 2442. Springer-Verlag; 2002. p. 354–68.
[15] Boneh D, Lynn B, Shacham H. Short signatures from the Weil pairings. Advances in Cryptology-Asiacrypt 2001, LNCS 2248. Springer-verlag; 2001. p. 514–32.

[16] Okamoto T, Pointcheval D. The gap-problems: a new class of problems for the security of cryptographic Schemes. PKC 2001, LNCS 1992. Springer-Verlag; 2001. p. 104–18.
[17] Schnorr CP. Efficient signature generation for smart cards. J Cryptol 4(3), Springer-Verlag; 1991. p. 239–52.
[18] Chaum D, Pedersen T. Wallet databases with observers. Advances in Cryptology Crypto 1992, LNCS 740. Springer-Verlag; 1993. p. 89–105.
[19] Bellare M, Rogaway P. The exact security of digital signatures – how to sign with RSA and Rabin. Advances in Cryptology-Eurocrypt 1996* LNCS 1070. Springer-Verlag; 1996. p. 399–416.
[20] Coron J. On the exact security of full domain hash. Advances in Cryptology-Crypto 2000, LNCS 1880. Springer-Verlag; 2000. p. 229–35.
[21] Ogata W, Kurosawa K, Heng S. The security of the FDH variant of Chaum's undeniable signature scheme. PKC 2005, LNCS 3386. Springer-Verlag; 2005. p. 328–45.

Xiaofeng Chen has received his Ph.D. in Cryptography from School of Telecommunications Engineering, Xidian University in 2003. He is an associate professor at School of Telecommunications Engineering in Xidian University, China. His research interest includes public key cryptography and applications.

Fangguo Zhang is a professor in the School of Information Science and Technology, Sun Yat-sen University, China. He obtained his Ph.D. degree in Cryptography from School of Telecommunications Engineering, Xidian University in 2001. His research interest includes cryptography and its applications.

Haibo Tian is a lecturer in the School of Information Science and Technology at Sun Yan-sen University, China. He obtained his Ph.D. degree from School of Telecommunications Engineering, Xidian University in 2006. His research interest includes design and analysis of cryptographic protocols.

Baodian Wei is an associate professor in the School of Information Science and Technology at Sun Yan-sen University, China. He obtained his Ph.D. degree from School of Telecommunications Engineering, Xidian University in 2004. His research interests include cryptology and its applications.

Kwangjo Kim has received his Ph.D. in Div. of Electrical and Computer Engineering in Yokohama National University, Japan in 1991. He is currently a Professor at Computer Science Department in KAIST, Korea. His research interest includes the theory and practice of cryptology and information security.