On the Key Exposure Problem in Chameleon Hashes

Giuseppe Ateniese
Department of Computer Science and
Information Security Institute
The Johns Hopkins University

Breno de Medeiros
Department of Computer Science
The Florida State University

ateniese@cs.jhu.edu, breno@cs.fsu.edu

Abstract

Chameleon signatures were introduced by Krawczyk and Rabin, being non-interactive signature schemes that provide non-transferability. However, that first construction employs a chameleon hash that suffers from a key exposure problem: The non-transferability property requires willingness of the recipient in consequentially exposing a secret key, and therefore invalidating all signatures issued to the same recipient's public key. To address this key-revocation issue, and its attending problems of key redistribution, storage of state information, and greater need for interaction, an identity-based scheme was proposed in [1], while a fully key-exposure free construction, based on the elliptic curves with pairings, appeared later in [7].

Herein we provide several constructions of exposure-free chameleon hash functions based on different cryptographic assumptions, such as the RSA and the discrete logarithm assumptions. One of the schemes is a novel construction that relies on a single trapdoor and therefore may potentially be realized over a large set of cryptographic groups (where the discrete logarithm is hard).

Keywords: Digital signatures, undeniable signatures, collision-resistant hashing, trapdoor commitments, chameleon signatures, chameleon hashing.

1 Introduction

A chameleon hash function is a trapdoor collision-resistant hash function: Without knowledge of the trapdoor information, a chameleon hash function has the same characteristics of any cryptographic hash function, such as pre-image and collision-resistance; however, collisions and second pre-images can be easily computed once the trapdoor is known.

An interesting application of chameleon hashing is to obtain non-transferable signature algorithms known as chameleon signatures.

1.1 Chameleon Signatures

Chameleon signatures, introduced in [12], are signature schemes based on the hash-and-sign paradigm. To authenticate a message m, a signer computes its digest value h using a chameleon hash function, and then signs h using an arbitrary signing algorithm of the signer's choice.

In applications, users include the description of a particular chameleon hash as part of their public keys, attesting their knowledge of the corresponding trapdoors. In this scenario, a signer who wished to provide a recipient with a non-transferable signature could hash the message to be signed with the chameleon hash function of the recipient, signing the resulting digest value. While the recipient is able to verify the signature as correct, a third party would only be able to ascertain that *some* message was signed (by the signer to the recipient). The third party would be aware that the signing value could have been re-used by the recipient to authenticate any message of choice, since the signature is a function of the hash value h alone and not of the original message, and because the recipient can easily find collisions for the hash value h. Therefore, a third party would not be willing to accept a proposed message content from the recipient in the absence of further evidence.

To determine the original message content one depends on a secondary signer affirmation or, if the signer is uncooperative, on the dispute settlement method described next.

Settlement of disputes. In case of disputes, it is easy to ascertain whether or not a proposed message content is indeed the original one committed by the signer. A judge would summon both signer and recipient. If the signer can produce a different message that is authenticated by the same signing value of the proposed message, the contested signature is considered invalid. Remember that such a collision proves that the recipient has put forth a forged signature at some point in time, as nobody apart from the recipient has more than a negligible probability of successfully finding a second message that produces the same signing value.

The dispute-settling procedure described above shows that chameleon signatures provide non-repudiation, as well as non-transferability. Unlike undeniable signatures (introduced in [6]), which similarly provide both of these features, chameleon signatures require no interaction between the signer and recipient for the purpose of issuing a signature. This great improvement in communication complexity comes at a mild disadvantage that, as mentioned above, chameleon signatures leak the information that a user signed something for a specific recipient, a characteristic not shared with undeniable signatures.

Applications of chameleon signatures. The non-transferability property is convenient in many scenarios in which the signer has a legitimate interest in controlling subsequent disclosures of the signed information. One application suggested in [1] is private auctions. In this context, the signer is an auction bidder, and does not wish to have its signed bid value published before the auction closing time and/or unless it wins the bidding. If the bid is leaked by the auctioneer, a second bidder (who does not trust the auctioneer) cannot ascertain the validity of the claimed bid without first contacting the original bidder. Another application that has been widely proposed for both undeniable and chameleon signatures is secure software distribution, resistant to bootlegging. Here a software vendor would distribute multiple binaries implementing the same functionality, making it difficult to recognize between bona-fide versions and potentially virotic ones. To protect legitimate users, the software vendor issues recipient-specific signatures on a particular binary. If later the user were to post the signed binary in a file-sharing network, others downloading the software would do so at their own risk – the presence of the non-transferable signature would not confer authenticity onto the posted binary.

1.2 The Key Exposure Problem

The first construction of a chameleon signature [12] employed for hash function the Chaum-Pedersen trapdoor commitment. More precisely, a potential recipient chooses and publishes a regular discrete logarithm-based public key $y = g^x$, where g is the generator of a cyclic group G and x is the secret key. Later, a user who wishes to sign message m can compute the chameleon hash value $h = y^m g^r$, where r is an auxiliary integer chosen uniformly at random by the signer. Here it is understood that m is a short binary message that has value smaller than the order of the group G when interpreted as the binary expansion of a non-negative integer. However, to extend the scheme to arbitrary length messages, it is sufficient to first hash the long message using a regular, cryptographic hash function.

Notice that if the recipient forges the signature, and two pairs (m, r) and (m', r') become known to the signer (during a dispute), the signer can recover the secret key x of the recipient from $h = g^m y^r = g^{m'} y^{r'}$, giving $x = \frac{m'-m}{r-r'}$.

This is a highly undesirable outcome from the recipient's viewpoint, as it invalidates all signatures ever issued to the associated public key y. A third-party is therefore more likely to believe claims made by the recipient about presenting an original (non-forged) signature, knowing that such forgery would negatively affect the recipient. In fact, the deterrent effect of key exposure on forgeries threatens the claims of non-transferability provided by the scheme. Therefore, to support non-transferability in any practical sense, we believe chameleon signatures schemes should *necessarily* rely on key-exposure free chameleon hash function, described next.

Previous work on Key Exposure Freeness. The problem of key exposure was partly addressed in [1], where it is shown how to build identity-based chameleon hash functions. The advantage of using the identity-based primitives is that applications could direct the use of transaction-specific chameleon hashes: The public key associated to a transaction-specific hash function is computed by the signer from specially formatted strings that describe the transaction, and which include the signer and recipient information as well as some nonce or time-stamp. In that paper, these strings were called *customized identities*. Later, if the recipient wishes to forge the signature, it suffices for him to communicate with the trusted authority (of the identity-based scheme) to recover the trapdoor information associated with the transaction-specific public key. It is understood that the trusted authority will only provide the trapdoor information to the recipient designated in the formatted string. Notice that the trapdoor recovery is an optional step, the trapdoor information being necessary only when the recipient wishes to deviate from the basic protocol by finding hash collisions and re-using signing tokens. This extra interaction adds less communication complexity than key revocation and key update in a classical public key infrastructure, but may still be too burdensome in certain applications, and therefore offering only a partial answer to the key exposure problem.

In [7], Chen et al. provide a specific construction of a key-exposure free chameleon hash function, working in the setting of Gap groups with bilinear pairings. While that certainly constitutes the first full construction of a key-exposure free chameleon hash, it does not settle the question of whether constructions exist that are either based on other cryptographic assumptions, or of more efficient schemes, for instance of comparable performance to the original chameleon hash function in [12].

Our Contribution. In this paper we show that key-exposure-free solutions exist whose security depends on non-pairing-based assumptions, such as the security of the RSA signature scheme. In fact, we show that the construction of [1] already enjoys the key-exposure-freeness property when used in a PKI setting instead of as the proposed identity-based application.

In all of the constructions, the public key is divided into two components, one permanent and the other ephemeral. Except for the scheme in section §4, all require a double-trapdoor context, and the components of the public key is made to correspond to each of the trapdoors. Non-transferability is supported through eventual compromise of the ephemeral component of the public key only. We also show that this technique can be applied broadly whenever a double-trapdoor is available.

More surprisingly, we have a novel construction of a key-exposure free chameleon hash function that does *not* rely on a double-trapdoor mechanism (section §4). To the best of our knowledge this is a novel result, of independent interest.

1.2.1 Organization of the paper.

In the following section, we provide the precise definition of key-exposure free chameleon hashes, and present several requirements that such hashes should satisfy for efficient application to a chameleon signature scheme. We follow that with a section that shows how chameleon hashes satisfying different subsets of these security requirements correspond to trapdoor commitment schemes satisfying different properties. Sections §4 and §5 present constructions of key-exposure free chameleon hashes based on single, and double trapdoors, respectively, and are followed by a few concluding remarks.

2 Definition and Requirements

A key-exposure free chameleon hash function is defined by a set of efficient algorithms:

Key Generation accepts as input a security parameter κ in unary form, and outputs a pair $(\mathcal{SK}, \mathcal{PK})$. It is a probabilistic algorithm, denoted as:

$$\mathbf{KeyGen}: \ 1^{\kappa} \longrightarrow (\mathcal{SK}, \mathcal{PK})$$

Hash accepts as input a public key $\mathcal{PK}$, a *l*abel $\mathcal{L}$, a message m and an auxiliary random parameter r and outputs a bitstring h of fixed length τ.

$$\mathbf{Hash}: \ (\mathcal{PK}, \mathcal{L}, m, r) \longrightarrow C \in \{0,1\}^{\tau}$$

Universal Forge accepts as input the secret key $\mathcal{SK}$ associate to public key $\mathcal{PK}$, a label $\mathcal{L}$, a message m, and auxiliary parameter r, and computes a second message m' and random parameter r' such that $\mathbf{Hash}(\mathcal{PK}, \mathcal{L}, m, r) = C = \mathbf{Hash}(\mathcal{PK}, \mathcal{L}, m', r')$.

$$\mathbf{UForge}(\mathcal{SK}, \mathcal{L}, m, r) \longrightarrow (m', r'), \text{ such that}$$
$$\mathbf{Hash}(\mathcal{PK}, \mathcal{L}, m, r) = C = \mathbf{Hash}(\mathcal{PK}, \mathcal{L}, m', r')$$

Instance Forge accepts as input a tuple $(\mathcal{PK}, \mathcal{L}, m, r, m', r')$ of a public key, a label, and two pairs of a message and auxiliary random parameter, where $C = \mathbf{Hash}(\mathcal{PK}, \mathcal{L}, m, r) = \mathbf{Hash}(\mathcal{PK}, \mathcal{L}, m', r')$, and computes another collision pair (m'', r'') that also satisfies $C = \mathbf{Hash}(\mathcal{PK}, \mathcal{L}, m'', r'')$.

$$\mathbf{IForge}(\mathcal{PK}, \mathcal{L}, m, r, m', r') \longrightarrow (m'', r''), \text{ such that}$$
$$\mathbf{Hash}(\mathcal{PK}, \mathcal{L}, m, r) = C = \mathbf{Hash}(\mathcal{PK}, \mathcal{L}, m', r') = \mathbf{Hash}(\mathcal{PK}, \mathcal{L}, m'', r'')$$

The security requirements of a chameleon hash include:[1]

Collision-resistance: There is no efficient algorithm that given only $\mathcal{PK}$, $\mathcal{L}$, m and r, (but not the secret key $\mathcal{SK}$) can find a second pair m', r' such that $C = \mathbf{Hash}(\mathcal{PK}, \mathcal{L}, m, r) = \mathbf{Hash}(\mathcal{PK}, \mathcal{L}, m', r')$ with more than negligible probability over the choices of $\mathcal{PK}$, $\mathcal{L}$, m and r.

Semantic Security: The chameleon hash value C does not reveal anything about the possible message m that was hashed. In formal terms, 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 $H[m|C]$ of the message given its chameleon hash value C equals the total entropy $H[m]$ of the message space.

Message hiding: Assume the recipient has computed a collision using the universal forgery algorithm, i.e., a second pair (m', r') s.t. $\mathbf{Hash}(\mathcal{PK}, \mathcal{L}, m, r) = C = \mathbf{Hash}(\mathcal{PK}, \mathcal{L}, m', r')$, where (m, r) was the original value signed. Then the signer, upon seeing the claimed values (m', r'), can successfully contest this invalid claim by releasing a third pair (m'', r''), without having to reveal the original signed message. Moreover, the entropy of the original value (m, r) is unchanged by the revelation of the pairs (m', r'), (m'', r''), and any further collisions: $H[(m, r)|C, (m', r'), (m'', r'')] = H[(m, r)|C]$.

Key Exposure Freeness: If a recipient with public key $\mathcal{PK}$ has never computed a collision under label $\mathcal{L}$, then given $C = \mathbf{Hash}(\mathcal{PK}, \mathcal{L}, m, r)$ there is no efficient algorithm that can find a collision (a second pair m', r' mapping to the same digest C). This must remain true even if the adversary has oracle access to $\mathbf{UForge}(\mathcal{SK}, \cdot, \cdot, \cdot)$ and is allowed polynomially many queries on triples $(\mathcal{L}_i, m_i, r_i)$ of his choice, except that $\mathcal{L}_i$ is not allowed to equal the challenge label $\mathcal{L}$.

[1]We adopt information-theoretic formulations of semantic security and message hiding properties because these lead to simpler proofs. Moreover, information-theoretic security (with respect to semantic security and message hiding) is indeed achieved by all constructions of chameleon hashing schemes currently in the literature as well the ones proposed in this paper.

Remark: Notice that when a chameleon hash with key-exposure freeness is employed within a chameleon signature then any label $\mathcal{L}$ must be explicitly committed to the signature along with the identity of the recipient and a description of the hashes (see [12]).

3 Previous Work on Trapdoor Commitments

Trapdoor commitment schemes were introduced as early as 1988, with the work of Brassard et al. [4]. Trapdoor commitment schemes are closely related to chameleon hashes. Yet the two notions are not truly equivalent. The reason is that chameleon hashes, intended for use in combination with signature schemes, require extra properties that are not enjoyed by every trapdoor commitment scheme. Indeed, in reviewing the literature in trapdoor commitments, we have identified at least four categories of commitments, which have all different degrees of suitability for use as a chameleon hashing scheme. The first category, what we called "stateful" trapdoor commitment schemes, cannot be used at all as chameleon hashes.

Stateful Trapdoor Commitments: These refer to trapdoor commitments which have the property that the knowledge of the trapdoor by itself is not sufficient to enable the computation of alternate de-commitments. In fact, it is necessary that the committing party know the trapdoor, execute a variant commitment algorithm that produces the commitment plus some auxiliary information, and save that auxiliary information (state) for later use in the alternate de-commitment algorithm. One example of such constructions are simulation-sound trapdoor commitments, see [13].

Such trapdoor commitment schemes cannot be used as chameleon hashes. In the chameleon hashing setting it is required that the recipient be able to find collisions (alternate de-commitments) from the digest value and the trapdoor information alone. *All chameleon hashes must be stateless trapdoor commitment schemes.*

Key Exposing Commitments: The chameleon hashing algorithm of Krawczyk and Rabin suffered from the key exposure problem, which limits its application in the chameleon hashing setting. It corresponds to the well-known Pedersen commitment scheme [17]. Another example of a trapdoor commitment that suffer from the key exposure problem is given by several constructions in Marc Fischlin's thesis [10], including one based on factoring, originally introduced in [8] (without reference to trapdoors). Another construction is provided by Shamir and Kalai [20] in relation to *online/offline* signatures (notion introduced by Even, Goldreich, and Micali [9]). In short, a signature on a chameleon hash is computed *offline* and then a signature on a new message is derived by computing a collision of the hash (*online* phase). Although very efficient, their original chameleon hash scheme, based on factoring, also suffers from the key exposure problem. This implies, in particular, that the online phase of their mechanism can actually be performed only once.

Non-Perfect-Hiding Commitments: These schemes are key exposure free, but they do not allow the execution of the instance forge algorithm **IForge**. In other words, they do not permit adjudication without the original valid signature being disclosed at some stage. These schemes might be interesting in a variety of application scenarios for which key exposure is not acceptable due to high cost of key-redistribution, but where the privacy of the message signed is no longer important at the time the matter comes for adjudication. In the following section we describe one original scheme of this type.

Message-Hiding and Key-Exposure-Free Commitments: This lead to the most flexible and suitable application as chameleon hashes. The first claim of construction of such a scheme is included in [7], where a pairings-based algorithm is given. However, the property is actually present (though not recognized as such) in the scheme in [1], based on a well-known RSA commitment scheme, first described in [15]. Therefore the use of pairings is not needed to obtain key-exposure-free schemes. In the following we explain how the RSA commitment scheme can be used to provide both guarantees, and we also present two new constructions. The first is based on a trapdoor commitment scheme [5] based on the Paillier's cryptosystem [16]. The second requires pairings, and is based on the trapdoor scheme described in [11]. Its security is dependent on the so-called Strong Diffie-Hellman assumption.

The crucial feature enjoyed by the commitment schemes in this category is that they are all double-trapdoor. One of the trapdoors is the secret key (used in the algorithm **UForge** to find collisions) and the other is some trapdoor function of the label used in the **IForge**.

4 Key Exposure Freeness Without Message Hiding

In this section we describe a trapdoor commitment scheme that can be seen as a chameleon hash providing key exposure freeness but no perfect message hiding. Unlike other schemes, it has the unique and appealing feature of relying on a single trapdoor.

The scheme is related to a twin Nyberg-Rueppel signature, introduced in [14]. The key generation is similar to that of other discrete logarithm-based schemes. Notice that while we describe the scheme in the finite field setting, the scheme may potentially be instantiated in other groups where the DL is hard.

Key Generation: The scheme specifies a *safe* prime p of bitlength κ. This means that $p = 2q + 1$, where q is also prime, and a generator g of the subgroup of quadratic residues $\mathbf{Q}_p$ of $\mathbf{Z}_p^*$, i.e, g has order q. The recipient chooses as secret key x at random in $[1, q-1]$, and his public key is computed as $(g, y = g^x)$. Let $\mathcal{H}$ be a collision-resistant hash function, mapping arbitrary-length bitstrings to strings of fixed length τ: $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\tau$.

The Hash Scheme: To commit to a message m, it is sufficient to choose random values $(r, s) \in \mathbf{Z}_q \times \mathbf{Z}_q$, and compute:

$$e = \mathcal{H}(m, r); \text{ and } \mathbf{Hash}(m, r, s) = r - (y^e g^s \bmod p) \bmod q.$$

Collision Finding: Let C denote the output of the chameleon hash on input the triple (m, r, s). A collision (m', r', s') can be found by computing (m', r', s') such that:

$$e' = \mathcal{H}(m', r'); \text{ and } C = r' - (y^{e'} g^{s'} \bmod p) \bmod q.$$

First, the recipient chooses a random message m', a random value $k' \in [1, q-1]$, and computes $r' = C + (g^{k'} \bmod p) \bmod q$, $e' = \mathcal{H}(m', r')$, and $s' = k' - e'x \bmod q$. Notice that indeed:

$$r' - (y^{e'} g^{s'} \bmod p) \bmod q = C + (g^{k'} \bmod p) - (g^{xe'} g^{s'} \bmod p) \bmod q) = C.$$

Key Exposure Freeness and Collision-Resistance: The security of the scheme depends on whether twice signing a message (without redundancy), using the above variant of Nyberg-Rueppel, is secure. This was proven in appendix A to [14], where the concept of twinning signature schemes is considered. The only difference from the scheme above is that we have substituted $e = \mathcal{H}(m, r)$ for r in the exponent of y. The only modification to the proof, which is formally the same, is that the probability of collisions is changed from finding collisions in the whole ring $\mathbf{Z}_q$ to finding them over the image of $\mathcal{H}(\cdot)$. Therefore, provided that this hash is collision-resistant, the conclusion of security is unchanged. Notice that we do not need to model the function $\mathcal{H}(\cdot)$ as a random oracle. Instead, the proof of security for the twin Nyberg-Rueppel works in the generic model of computation.

Semantic Security: Notice that when the committing party computes the value C, it can choose s completely independently of m and r. Since g is an element of order q, the term g^s uniformly covers the whole subgroup of quadratic residues, independently of the value $ry^{\mathcal{H}(m,r)}$. The result follows. More formally, from the commitment equation, for each random r and message m, there is a one-to-one correspondence between the commitment C and the value s. This implies that the conditional probability $\mu(m, r|C)$ equals $\mu(m, r|s)$. But the latter value is simply $\mu(m, r)$ since s is chosen independently of m and r.

Now, consider the definition of conditional entropy:

$$H[m, r|C] = - \sum_{m,r \in \{0,1\}^\tau \times \mathbf{Z}_q} \sum_{C \in \mathbf{Z}_q} \mu(m, r, C) \log(\mu(m, r|C)).$$

The internal summation becomes $\sum_{C \in \mathbf{Z}_q} \mu(m, r, C) \log(\mu(m, r))$, which equals $\mu(m, r) \log(\mu(m, r))$. Therefore,

$$H[m, r|C] = - \sum_{m,r \in \{0,1\}^\tau \times \mathbf{Z}_q} \mu(m, r) \log(\mu(m, r)) = H[m, r].$$

Notice that the proof is automatic once shown that the probability of the message conditioned on the commitment value is really equal to the unconditioned probability. In the remaining schemes we describe here, our proofs of semantic security will be shortened to only show the equality between conditioned and unconditioned probabilities.

Remark: We have defined the above scheme using the short version of the Nyberg-Rueppel signature, for convenience of reference to the twin signatures work [14]. It is also possible to define a trapdoor commitment scheme using the long form, as:

$$e = \mathcal{H}(m, r); \text{ and } \mathbf{Hash}(m, r, s) = ry^e g^s \bmod p.$$

5 Key Exposure Freeness with Message Hiding

In this section we provide some examples of chameleon hash with key exposure freeness. Any stateless trapdoor commitment with two trapdoors may be adequate, but the schemes below are based on common assumptions and well-known constructions.

We stress that these candidate functions are not new but are rather well-known and have been proposed as trapdoor commitments by others. We are showing that, unlike other trapdoor commitment schemes, they can be easily adapted and transformed in chameleon hashes that provide simultaneously key exposure freeness and message hiding.

5.1 Scheme based on RSA and Factoring

The scheme below is based on a well-known trapdoor commitment and has two trapdoors, the factors of an RSA modulus and roots of elements in the RSA ring. This fact has been exploited in [1] to build an identity-based chameleon hash and was independently noticed by Gennaro in [11], where the basic scheme is also extended to allow for several trapdoors and applied to the construction of concurrently non-malleable proofs of knowledge.

Key Generation: Let τ and κ be security parameters. As before, let $\mathcal{H}$ be a collision-resistant hash function mapping strings of arbitrary length to strings of fixed length τ. Let $n = pq$ with the two prime numbers p and q in the set $\{2^{\kappa-1}, \ldots, 2^{\kappa} - 1\}$. A random prime integer e is computed s.t. $e > 2^{\tau}$, and such that it is relatively prime to the order $\phi(n) = (p-1)(q-1)$ of the multiplicative residues modulo n. The secret key d is computed such that $ed \equiv 1 \bmod \phi(n)$.

The recipient's public key is (n, e) and his secret key is (p, q, d).

The Hash Scheme: Let S be the string uniquely identifying the recipient and let $\mathcal{L}$ be a label. Let $\mathcal{C} : \{0,1\}^* \rightarrow \{0, \cdots, 2^{2\kappa-1}\}$ be a secure hash-and-encode scheme, mapping arbitrary bit-strings to integers less than n. In general, such schemes are probabilistic, requiring an auxiliary random string. For instance, the EMSA-PKCS encoding, defined in [19], requires a random (or pseudo-random) salt at least 64 bits long, while the EMSA-PSS encoding, defined in [2, 18], can take an auxiliary random string of bit-length equal to τ, the output length of the cryptographic hash function $\mathcal{H}$. Our notation will not make the random parameter explicit, as the nature of the encoding (i.e., deterministic or non-deterministic) is immaterial to the following discussion as long as the encoding scheme is uniquely invertible, i.e., the output of the encode function can be decoded to recover the hash value.

Given $J = \mathcal{C}(\mathcal{L})$ in $\mathbf{Z}_n$, the secret trapdoor is extracted as $B = J^d \bmod n$, i.e., a secure RSA signature on $\mathcal{L}$.

The **Hash**$(\cdot)$ algorithm is:

$$\textbf{Hash}(\mathcal{L}, m, r) = J^{\mathcal{H}(m)} r^e \bmod n, \text{ where } J = \mathcal{C}(\mathcal{L})$$

Collision Finding: To compute a collision (m', r'), the recipient would chose a random message m' and consider the following equation:

$$J^{\mathcal{H}(m)}r^e = J^{\mathcal{H}(m')}r'^e,$$

and solve it for r' modulo n, that is:

$$r' = rB^{\mathcal{H}(m)-\mathcal{H}(m')} \bmod n.$$

Collision-Resistance and Key Exposure Freeness: Exposing a collision allows anybody to extract the secret key B associated to the value $J = \mathcal{C}(L)$. Indeed,

$$J^{\mathcal{H}(m)}r^e = J^{\mathcal{H}(m')}r'^e \Longrightarrow r'/r = B^{\mathcal{H}(m)-\mathcal{H}(m')}.$$

Clearly, the absolute value of $\Delta = \mathcal{H}(m) - \mathcal{H}(m')$ is smaller than 2^τ and given that e is a prime integer larger than 2^τ, it follows that $\gcd(\Delta, v) = 1$. Using the extended Euclidean algorithm for the GCD, one computes α and β such that $\alpha\Delta + \beta v = 1$. B can now be extracted:

$$B = (r'/r)^\alpha J^\beta.$$

As B is a secure RSA signature on $\mathcal{L}$, and computing collisions is equivalent to breaking this signature scheme, we conclude that finding collisions is hard without knowledge of the trapdoor. Finally, notice that since revealing collisions is equivalent to computing signatures, the scheme is safe from key exposure as the EMSA-PSS RSA signature scheme is resistant against active attacks.

Semantic Security: For each message m, the value $C =$**Hash**$(\mathcal{L}, m, r)$ is uniquely determined by the value r, and vice-versa. Therefore, the conditional probability $\mu(m|C)$ equals that of $\mu(m|r)$, which equals $\mu(m)$, as m and r are independent variables. The semantic security follows – see the example in the previous section for details.

Message Hiding: Let C be the commitment value. It is sufficient to show that, once a collision is revealed, a person who does not know the trapdoor can compute a de-commitment to C under any message m'' of her choice. From the above proof of the collision-resistance property we see that the revelation of a collision $(m, r), (m', r')$ discloses the trapdoor information $B = \mathcal{C}(\mathcal{L})^d$. In order to find another collision, it is sufficient to choose m'' and set $r'' = rB^{\mathcal{H}(m)-\mathcal{H}(m')}$.

Remark: This RSA-based scheme is a multi-trapdoor scheme in the sense of Gennaro [11], as the second trapdoor is multiply instantiated – in other words, there is one trapdoor per label. Instead of relying on the Strong RSA assumption as the scheme described in [11], the version described above relies on the security of the EMSA-PSS RSA signature.

5.2 Scheme based on RSA[n,n] and Factoring

In [16], Paillier proved that, for each $h \in \mathbf{Z}^*_{n^2}$ of order a non-zero multiple of n, the function $\mathcal{F}_h$ that maps elements in $(\mathbf{Z}_n, \mathbf{Z}^*_n)$ to elements in $\mathbf{Z}^*_{n^2}$, defined as:

$$\mathcal{F}_h : (a, b) \longrightarrow h^a b^n \bmod n^2,$$

is a trapdoor permutation.

In [5], a trapdoor commitment scheme was introduced that is based on the Paillier trapdoor permutation. The authors of [5] suggest to use their trapdoor commitment to build a chameleon signature. However, if used directly as they described (i.e., a standard signature over a trapdoor commitment), the problem of key exposure would arise. We simply observe that their trapdoor commitment has actually two trapdoors and can be easily extended to support labels as described below.

Key Generation: Let $n = pq$ with p and q large primes. Let $\mathcal{H}(\cdot)$ be a cryptographic hash function that maps elements of arbitrary length to element of a subset of $\mathbf{Z}^*_{n^2}$. The public key of the scheme is n, while the private key is (p, q).

The Hash Scheme: Given a message $m \in \mathbf{Z}_n$, and a label $\mathcal{L}$, compute $h = \mathcal{H}(\mathcal{L})$. Next, generate randomly the pair $(r_1, r_2) \in (\mathbf{Z}_n, \mathbf{Z}^*_n)$, and compute ([5]):

$$C = \mathbf{Hash}(\mathcal{L}, m, r_1, r_2) = (1 + mn)h^{r_1} r_2^n \bmod n^2.$$

To extend the scheme to commit arbitrary length messages, it is sufficient to employ a cryptographic hash function with codomain in $\mathbf{Z}_n$.

Collision-Finding and Collision-Resistance: Let C be a commitment with respect to label $\mathcal{L}$, where $h = \mathcal{H}(\mathcal{L})$. From here to compute a collision under a second message m', the recipient finds $C' = C(1 - m'n) \bmod n^2$ and computes its inverse under the trapdoor permutation $\mathcal{F}_h$:

$$\mathcal{F}_h^{-1}(C') = (a, b), \text{ with } a \in \mathbf{Z}_n \text{ and } b \in \mathbf{Z}^*_n.$$

The new de-commitment is $(m', r'_1 = a, r'_2 = b)$. Without knowledge of the trapdoor, computing this collision is equivalent to breaking the RSA[n, n] assumption, as shown in [5].

Key Exposure Freeness: Suppose a party can compute a collision under label $\mathcal{L}$, i.e., values $(m, r_1, r_2, m', r'_1, r'_2)$ such that

$$(1 + mn)h^{r_1} r_2^n = (1 + m'n)h^{r'_1}(r'_2)^n,$$

where $h = \mathcal{H}(\mathcal{L})$. It follows that (see full argument in [5]) the one can recover values a and b such that

$$\mathcal{H}(\mathcal{L}) = h = (1 + an)b^n, \tag{1}$$

i.e., the party can compute the Paillier signature (a, b) on the "message" $\mathcal{L}$. This is not feasible, since the Paillier signature is resistant against existential forgeries under active attacks in the random oracle model, by reduction to the RSA[n,n] assumption.

Semantic Security: The semantic security of this commitment scheme has been shown in [5].

Message Hiding: Assuming a collision (m', r_1', r_2') has been revealed to the committing party, she has learned a Paillier signature (a, b) on the value $\mathcal{L}$. To obtain a collision, she computes the value $\delta = a^{-1} \bmod n$, chooses an arbitrary message m'', and computes $r_1'' = r_1' + \delta(m' - m'')$ (as an integer) and $r_2'' = r_2' + \delta(m'' - m') \bmod n$. One may readily verify that (m'', r_1'', r_2'') commits to the same value as (m', r_1', r_2').

Remark: Note that when computing the collision, the new value r_1' may fall outside the interval $[1, n-1]$. This is not a problem, as there is no requirement that collisions look like plausible initial commitments. In a chameleon hashing scheme the goal is just to prove that the trapdoor-owner has revealed a collision. If it is required that derived collisions look like plausible commitments, the scheme can be "fixed" by redefining the interval where r_1 is chosen to be much larger than $[1, n-1]$.

5.3 Scheme based on SDH and DL

Gennaro in [11] proposes a new trapdoor commitment scheme based on the Strong Diffie-Hellman (SDH) assumption introduced by Boneh and Boyen [3]. Informally, the ℓ-SDH assumption says that if G is a cyclic gap-DDH group of prime order q and g is a generator of such a group, then an attacker that sees G, g, g^x, ..., g^{x^ℓ}, for an $x \in \mathbf{Z}_q$, should not be able to compute h and e such that $h^{x+e} = g$.

We show here that such a trapdoor commitment scheme can support labels and that collisions cannot be used to compute the master trapdoor.

Key Generation: Let $G = \langle g \rangle$ be a gap-DDH group of order q and let $x \in \mathbf{Z}_q$. The public key of the recipient is $h = g^x$. Let $\mathcal{H}(\cdot)$ be a cryptographic hash function that maps elements of arbitrary length to elements of $\mathbf{Z}_q$.

The Hash Scheme: Let $e = \mathcal{H}(\mathcal{L})$, where $\mathcal{L}$ is the label. Given a message m, select a random $r \in \mathbf{Z}_q$ and compute ([11]):

$$\mathbf{Hash}(\mathcal{L}, m, r) = g^{\mathcal{H}(m)}(g^e h)^r.$$

Let F denote the output of the chameleon hash divided by $g^{\mathcal{H}(m)}$. To verify that the hash was computed correctly, one can check whether (g^r, hg^e, F) is a Diffie-Hellman triple. Remember that Gap groups have efficient algorithms to decide the Diffie-Hellman problem.

Collision Finding, Collision-Resistance, and Key Exposure Freeness: Following [11], given a pair (m, g^r), it is efficient to find a collision $(m', g^{r'})$ if x is known by setting:

$$g^{r'} = g^r g^{[(\mathcal{H}(m) - \mathcal{H}(m'))/(x+e)]}. \tag{2}$$

Conversely, exposing a collision (m, g^r) and $(m', g^{r'})$ allows anybody to efficiently compute $g^{1/(x+e)}$ (which, in general, is a hard computational problem under the SDH assumption). To obtain key exposure freeness, one needs the result that even if several values $f_i = g^{1/(x+e_i)}$ are known, it is difficult computing other values f_j under different e_j – i.e., under different labels $\mathcal{L}_j$. As remarked in [11], this result has been proved in [3], where the values f_i are shown to be "weak signatures." Moreover, since the knowledge of x permits computing such values efficiently, it is clear that deriving x from collisions is infeasible.

Semantic Security and Message Hiding: From the discussion above, it is clear that with the trapdoor information, exactly one collision can be computed under each message m', proving the semantic security of the scheme. It remains to be shown that collisions, and the consequent exposure of the value $f_e = g^{1/(x+e)}$, permit finding other collisions under the same label. It is sufficient to observe that the collision-finding equation (2) does not require knowledge of x, but only of the value f_e.

6 ID-based Chameleon Hashing

The double-trapdoor mechanism, introduced in the previous sections to build key-exposure-free chameleon hashes, can be used to build identity-based ones, as shown in [1]. The resulting ID-based mechanisms are not key-exposure free, though key exposure may be overcome through "identity customization", described below.

It is a standard assumption with ID-based schemes that system users are universally identifiable by IDs, i.e., unique bit-strings easily derivable from the public information associate with the the individual within the system. For example, an ID could be a user's e-mail address, augmented by some information such as an expiration-date. If identity customization is desired, this basic ID is augmented with information specific to a particular transaction (such as a message sequence number and IDs of the other parties to the transaction).

An ID-based chameleon hashes share all the benefits typical of ID-based constructs (such as ID-based encryption or signature schemes); in particular they dispense with the need for explicit mechanisms for distribution of keys and certificates. In the context of chameleon hashing these benefits are compounded by the fact that entities/users do not need to contact the trusted authority to recover the secret key associated to their IDs, as is the case with identity-based schemes in general. Indeed, the only use of the trapdoor information within a chameleon hashing scheme is the production of forgeries. As long as the parties do not wish to deviate from the basic scheme they do not need to contact the third party, while still enjoying the non-transferability conferred simply by the *potentiality* of such forgeries – a third-party cannot ascertain whether the recipient of a signature has not recovered the trapdoor information it is entitled to.

Moreover, through identity customization – having an augmented ID appear as the recipient of a single signature – the ID-based scheme may reduce the penalty for the recipient for engaging in forgery to the loss of the original signature only. Therefore, in practice, such schemes achieve similar benefits of key-exposure free chameleon hashes as well as all the advantages of being identity-based.

6.1 Description

An ID-based chameleon signature on a message consists of a traditional signature scheme, such as RSA or DSA, computed over the ID chameleon hash of a message under the identity of the intended recipient.

As with all identity-based schemes, only the trusted third party can extract the secret key corresponding to an identity – though as mentioned above, such a secret does not need to be retrieved by the recipient unless he wants to forge the signature. This last observation is crucial to the performance of the scheme, particular when used in combination with identity customization. If the secret key needed to be recovered for every customized identity, the scheme would revert to an online protocol.

6.1.1 Identity Customization

The identity-based chameleon hash function offers a natural way to mitigate the key exposure problem without requiring the recipient to interact with the signer. Indeed, the hash can be computed under a *customized* identity J; for instance, J could be the result of applying a hash-and-encode function $\mathcal{C}(\cdot)$ to the identity of the recipient, concatenated with the identity of the signer, plus some transaction identifier:

$$J = \mathcal{C}(identity\_recipient \mid\mid identity\_signer \mid\mid transaction\_ID).$$

In this case, the scheme should stipulate that the secret key corresponding to J will be provided only to the recipient, identified as the person whose identity prefixes the string used to form the public key J. The use of properly customized identities eliminates also the need for checking whether secret keys have been compromised given that customized identities are unique to each message signed and the corresponding secrets have not even been computed by the trusted authority. As remarked in [1], in a practical system, recipients would be eventual signers, and thus would have an interest in recovering secret keys occasionally, to ensure that the trusted party works correctly and that the non-transferability property holds. Clearly such frequency could be considerably lower than the frequency of transactions the recipient participates in.

6.1.2 Constructions

The idea is to have a trusted third party replace the role of the recipient in the key generation process and produce several instances of the *ephemeral* public key, one per each recipient. The label $\mathcal{L}$ in the key-exposure-free schemes is then substituted with a customized identity when computing the chameleon hash.

For instance, the scheme described in Section §5.1 can be easily turned into an ID-based one as shown in [1]. In practice, the *main* trapdoor (the factors of the RSA modulus n) is owned by the trusted party T, which then generates all the parameters required by the scheme. The hash-and-encode mapping, $\mathcal{C}$, must now be deterministic (for instance, it is possible to use the deterministic version of the EMSA-PSS encoding) and J is computed as $J = \mathcal{C}(S)$ in $\mathbf{Z}_n$ where S is the identity string associated to the recipient. The secret key B, a secure RSA signature on J, is given by T to the recipient (optionally, under request from such recipient).

The $\mathsf{Hash}(\cdot)$ algorithm is:

$$\mathsf{Hash}(S, m, r) = J^{\mathcal{H}(m)} r^v \bmod n,$$

where $\mathcal{H}(\cdot)$ is the secure hash function.

Given the knowledge of B, the recipient can easily find a collision (m', r') by computing:

$$r' = rB^{\mathcal{H}(m)-\mathcal{H}(m')} \bmod n.$$

A similar technique can be used to build ID-based chameleon hashes from the schemes described in Sections §5.2 (based on RSA[n,n] and Factoring) and §5.3 (based on SDH and DL).

7 Conclusions and Open Problems

In this paper we outline a formal security model for chameleon hash functions, including precise specifications of the message-hiding and key-exposure freeness properties. We conclude that single-trapdoor commitment schemes are not sufficient for the construction of chameleon hashes – instead a double-trapdoor mechanism is required. Here an interesting question poses itself: The double-trapdoor mechanism can either be used to construct an identity-based chameleon hash scheme (in the sense of [1]) or a key-exposure free one, but not both. Are there *efficient* schemes that are simultaneously identity-based and key-exposure free, perhaps based on a construction with multiple (more than two) consecutive trapdoors?

Our results include three constructions of schemes satisfying the full security model, two based on RSA, as well as a construction based on pairings. This significantly augments the family of chameleon hashes satisfying both key-exposure freeness and message hiding, of which only one example was previously known ([7]), based on pairings. We have also provided an example of trapdoor commitment that provides key-exposure freeness, but not message hiding – and that relies on a single trapdoor, the first such construction to our knowledge.

Acknowledgments. We are grateful to Rosario Gennaro and David Molnar for their insightful comments. This work was partly supported by a NSF grant.

References

[1] Ateniese, G., de Medeiros, B.: Identity-based chameleon hash and applications. In Financial Cryptography 2004. LNCS 3110, Springer-Verlag (2004) 164–180. Available online at `http://eprint.iacr.org/2003/167/`.

[2] Bellare, M., Rogaway, P.: PSS: Provably secure encoding method for digital signature. IEEE P1363a: Provably secure signatures. `http://grouper.ieee.org/groups/1363/-p1363a/pssigs.html` (1998)

[3] Boneh, D. and Boyen, X.: Short Signatures Without Random Oracles, Proc. of Advances in Cryptology – EUROCRYPT '04, Volume 3027 of LNCS Springer-Verlag (2004) 56–73

[4] Brassard, G., Chaum, D., and Crepeau, C.: Minimum disclosure proofs of knowledge. Journal of Computer an Systems Sciences, vol. 37 no. 2, (1988) 156-189.

[5] Catalano, D., Gennaro, R., Howgrave-Graham, N., Nguyen, P.Q.: Paillier's Cryptosystem Revisited. ACM CCS 2001.

[6] Chaum, D., Antwerpen, H.: Undeniable signatures. In: Advances in Cryptology – CRYPTO'89. Volume 435 of LNCS., Springer-Verlag (1991) 212–216

[7] Chen, X., Zhang, F., and Kim, K.: Chameleon hashing without key exposure. To appear in the proceedings of the 7th Information Security Conference (ISC '04), Palo Alto, California. Available online at `http://eprint.iacr.org/2004/038/`.

[8] Damgård, I.: Practical and provable secure release of a secret and exchange of signature. In: Journal of Cryptology, Volume 8. Springer-Verlag (1995) 201–222

[9] Even S., Goldreich O., Micali S.: On-line/off-line Digital Signatures. In: Advances in Cryptology – CRYPTO '89. LNCS of Springer-Verlag, 263–277. August 1990.

[10] Fischlin, M.: Trapdoor commitment schemes and their applications. Ph.D. thesis, 2001.

[11] Gennaro, R.: Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks. In: Advances in Cryptology – CRYPTO '04. LNCS of Springer-Verlag. August 2004.

[12] Krawczyk, H., Rabin, T.: Chameleon signatures. In: Proceedings of NDSS 2000. (2000) 143–154

[13] MacKenzie, P. and Yang, K.: On Simulation-Sound Trapdoor Commitments. In: Proc. of Advances in Cryptology – EUROCRYPT '04. Volume 3027 of LNCS, Springer-Verlag (2004) 382–400

[14] Naccache, D., Pointcheval, D., and Stern, J.: Twin signatures: an alternative to the hash-and-sign paradigm. In: Proc. of the 8th ACM Conference on Computer and Communication Security (ACM CCS), ACM Press (2001) 20–27

[15] Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Proc. of Advances in Cryptology – CRYPTO '92. Volume 740 of LNCS, Springer-Verlag (1992) 31–53

[16] Paillier, P.: Public key cryptosystems based on composite degree residuosity classes. In: Proc. of Advances in Cryptology – EUROCRYPT '99. Volume 1592 of LNCS, Springer-Verlag (1999) 223–238

[17] Pedersen, T. Non-interactive and information-theoretic secure verifiable secret sharing. In: Proc. of Advances in Cryptology – CRYPTO '91. Volume 576 of LNCS, Springer-Verlag (1991) 129–149

[18] RSA Labs: RSA Crypt. Std: EMSAPSS – PKCS#1 v2.1., pp. 26–28, 32–37 (2002)

[19] RSA Labs: RSA Crypt. Std: EMSAPKCS1-v1_5 - PKCS#1 v2.1. pp. 29–33, 37–38 (2002)

[20] Shamir A, Kalai, Y.: Improved Online/Offline Signature Schemes. In: Proc. of Advances in Cryptology – CRYPTO'01, Volume 2139 of LNCS. Springer-Verlag (2001) 355–367.