# Identity-based Chameleon Hash and Applications

Giuseppe Ateniese      Breno de Medeiros
{ateniese, breno}@cs.jhu.edu

Department of Computer Science
The Johns Hopkins University

**Abstract.** Chameleon signatures are non-interactive signatures based on a hash-and-sign paradigm, and similar in efficiency to regular signatures. The distinguishing characteristic of chameleon signatures is that their are non-transferable, with only the designated recipient capable of asserting its validity. In this paper, we introduce the first identity-based chameleon hash function. The general advantages of identity-based cryptography over conventional schemes relative to key distribution are even more pronounced in a chameleon hashing scheme, because the owner of a public key does not necessarily need to retrieve the associated secret key. We use the identity-based chameleon hashing scheme to build the id-based chameleon signature and a novel sealed-bid auction scheme that is robust, communication efficient (bidders send a single message), and secure under a particular trust model.

**Keywords:** Digital signatures, secure hash functions, chameleon hashing, sealed-bid auctions

## 1 Introduction

Chameleon signature schemes were introduced in [1]. A distinguishing feature of chameleon signature schemes is that they are *non-transferable,* i.e. a signature issued to a designated recipient cannot be validated by another party. While not universally verifiable, chameleon signatures provide *non-repudiation*: If presented with a false signature claim, the signer can prove that the signature is forged, while incapable of doing so for legitimate claims. Accordingly, the signer's refusal to invalidate a signature is considered equivalent to her affirmation that the signature is valid.

Unlike *undeniable signatures* [2–6], which also provide non-repudiation and non-transferability, chameleon signatures are non-interactive protocols. More precisely, the signer can generate the chameleon signature without interacting with the designated recipient, and the latter will be able to verify the signature without interacting with the former. Similarly, if presented with a forged signature, the signer can deny its validity by revealing certain values. These values will revoke the original signature and the forged one simultaneously, and the revocation can be universally verified. In other words, the forged-signature denial protocol is also non-interactive. There also exist non-interactive versions of undeniable signatures [7]. Chameleon signatures are considerably less complex, at the sacrifice of not conferring the signer the ability to engage in non-transferable secondary proofs of signature (non-)validity.

Chameleon signatures are based on the 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: Without knowledge of the associated trapdoor, the chameleon hash function is resistant to the computation of pre-images and of collisions. However, with knowledge of the trapdoor, collisions are efficiently computable.

When a chameleon hash function is used within a hash-and-sign signature scheme, it permits the party with knowledge of the trapdoor to re-use the signature value to authenticate other messages of choice. In particular, if the hash function is part of the recipient's public key, then the signature is publicly verifiable by no one other than the intended recipient. On the other hand, if the recipient re-uses the hash value to obtain a signature on a second message, the signer can prove knowledge of a hash collision, since the original signed message and the claimed signed message have the same hash value. Because computing hash collisions is infeasible for the signer, possession of such a collision is seen as proof of forgery by the signature recipient.

## 1.1 ID-based Chameleon Hashing

In this paper, we propose the first ID-based chameleon hashing scheme. ID-based cryptography is an alternate form of public-key cryptography that does not use certification authorities or certificates. Instead, an ID-based scheme defines "identity strings", which are nothing more than a special string format to describe real entities (persons or machines). For instance, an identity string could be an e-mail address, a URL, a person's address, or any other unambiguous reference. The public keys are derived from these identity strings by means of a public algorithm, which is part of the scheme description. Any entity that can be described with an identity string (as specified in the particular scheme) has automatically a public key. Since the identity string is a 'natural' way to refer to the entity, anyone who knows the entity will also be able to compute the entity's public key, without having to look it up in a key distribution center. Instead, the owner of the key is responsible for contacting an escrow server to obtain the secret key associated to his public key. Identity-based cryptography was originally introduced in the classical paper [8], which described id-based identification and signature schemes. Despite several efforts, id-based encryption eluded researchers until recently [9, 10].

As with other cryptographic primitives, such as encryption and regular digital signatures, there are considerable advantages to be gained from employing an ID-based chameleon hash scheme over a regular scheme. For instance, a signer can sign a message to an intended recipient, without having to first retrieve the recipient's certificate; in fact, the signer can sign a message to the recipient before the recipient has registered with the system and obtained the secret key. This reduces the negative network effects that make it hard to deploy public key infrastructures.

One limitation of the original chameleon signature scheme ([1]) is that signature forgery results in the signer recovering the recipient's trapdoor information. Such feature has some advantages: For instance, since with the knowledge of the trapdoor the signer can compute other collisions, she can prove knowledge of a collision *without revealing the original message signed.* This *message hiding* permits the signer to deny forged messages without the onus of confirming the original signature. However, the signer can use this trapdoor information to deny *other* signatures given to the recipient, whether or not the recipient tried to re-use the corresponding hash values. In the worst case, the signer could collaborate with other individuals to invalidate any signatures which were designated to be verified by the same public key. This creates a strong disincentive for the recipient to forge signatures, partially undermining the concept of non-transferability. If a third party is aware of the potential damage to the recipient that would result from the forgery of a signature, she will likely believe the authenticity of a recipient's claims.

Non-transferability is more convincing if the scheme is such that forgery of a hash value does not compromise the long-term key of the recipient. Clearly, this can be accomplished in any chameleon hashing scheme if the public keys are changed often. However, one now has a key distribution problem. The signer must be able to retrieve the most recent public key for the recipient, most likely by interacting with the recipient itself. The protocol then ceases to be non-interactive, at least in a practical sense.

The scheme we present permits the signer to use a different public key for each transaction with a recipient, without having to retrieve a new certificate: The signer concatenates the recipient's identification information with a string which uniquely identifies the transaction. Notice that the recipient *does not need to retrieve the associated secret key to verify the signature's validity. Only if the recipient wishes to compute a collision and a forgery does he need to recover the secret key.* Clearly, the key escrowing function is provided by the scheme manager for the bona fide reason that non-transferability, i.e., the ability of the recipient to forge signatures, is a property of the scheme that protects the legitimate signer. We remark that collision-forgery still results in the trapdoor information – now associated to a single transaction key – being recovered. Therefore, if the recipient produces a hash collision, the signer may deny the original message by providing a different collision, still enjoying the message hiding property. She will not be capable, however, of denying signatures on any other messages.

## 1.2 Our Results

To summarize, chameleon signature schemes introduced in [1] have the following properties.

– Non-repudiation: The signer cannot deny legitimate signature claims.
– Non-transferability or non-universal verifiability.

– Non-interactive.
– Practical and efficient: The algorithms have costs comparable with those of standard signature schemes.
– Semantic security: The hash value does not reveal information about the message signed.
– Message hiding: The signer does not have to reveal the original message to deny the validity of a forgery.
– Convertibility: A variant of the chameleon signature can be transformed into a regular signature by the signer.

The scheme we present in this paper enjoys all the attributes above, with the added characteristics of being ID-based:

– ID-based: Parties can sign messages for recipients, even if these recipients do not yet have the corresponding secret keys. Also, signers do not need to retrieve public key certificates of the intended recipient.
– Lightweight key distribution/refreshment: Public keys do not need to be distributed after a refreshment. Secret key retrieval is optional for recipients.
– Stronger non-transferability: The penalty for the recipient for engaging in forgery can be restricted to the loss of the original signature.

We describe in later sections some applications which make use of these extra properties of the ID-based scheme. These applications are not meant to be the only motivation for the primitive we are introducing, but only illustrative examples of its potential uses.

## 2 ID-based Chameleon Hashing

We assume that all system users are identifiable by a bit-string easily derivable from public knowledge about the individual. For instance, it could be the user's e-mail address, augmented by some information such as an expiration-date. We call such a string an *identity* string. Formally, an ID-based chameleon hashing scheme is defined by a family of efficiently computable algorithms:

Setup: A trusted party, the key escrow, runs this efficient, probabilistic algorithm to generate a pair of keys $\mathcal{SK}$ and $\mathcal{PK}$ defining the scheme. It publishes $\mathcal{PK}$ and keeps $\mathcal{SK}$ secret. The input to this algorithm is a security parameter(s).

Extract: An efficient, deterministic algorithm that, on inputs $\mathcal{SK}$ and an identity string $S$, outputs the trapdoor information $B$ associated to the identity.

Hash: An efficient, probabilistic algorithm that, on inputs $\mathcal{PK}$, an identity string $S$, and a message $m$, outputs a hash value $h$.

Forge: An efficient algorithm that, on inputs $\mathcal{PK}$, an identity string $S$, the trapdoor information $B$ associated with $S$ (i.e., the output of $\mathsf{Extract}(\mathcal{SK}, S)$), a message $m'$, and a hash value $h$ of a message $m$, outputs a sequence of random bits that correspond to a valid computation of $\mathsf{Hash}(\mathcal{PK}, S, m')$ yielding the target value $h$.

Repeating a computation of the hash function involves knowledge of the random choices made by the algorithm. We use also the notation $\mathsf{Hash}(S, m, r)$ to denote the *deterministic* algorithm that takes as inputs an identity string, a message and a string of random bits. (This notation leaves implicit the public key of the trusted party.) In practice, the Forge algorithm needs as input the random string $r$ leading to a valid computation of $h = \mathsf{Hash}(S, m, r)$ and then can output a second $r' \neq r$ such that also $h = \mathsf{Hash}(S, m', r')$. We denote the deterministic algorithm by $\mathsf{Forge}(S, B, m, r, h, m')$, where $B = \mathsf{Extract}(\mathcal{SK}, S)$.

### 2.1 Security Requirements

In establishing the security of a chameleon hashing scheme, one may consider vulnerabilities against different attacks. A total break would be an attack resulting in the recovery of the secret key $\mathcal{SK}$ of the trusted party. This would permit the attacker to obtain the trapdoor information associated to any given identity string.

A lesser powerful attack model involves extracting the trapdoor information of some identity string, not necessarily a meaningful string, chosen by the attacker, i.e., an existential attack on the extract protocol. In this case, the string very probably does not correspond to the identity of any real-world party. Such an attack is dangerous inasmuch as it can lead to an advantage in forging hash collisions for hash values published by real-world entities.

Any attack is said to be successful if it accomplishes *collision forgery*.

**Definition 1.** (Collision forgery): *A collision-forgery strategy is an efficient, probabilistic algorithm that, given identity string $S$, message $m$, and random bits $r$, outputs another message $m'$ and random bits $r'$, such that* $\mathsf{Hash}(S, m, r) = \mathsf{Hash}(S, m', r')$*, with non-negligible probability.*

We say that the hashing scheme is *secure against existential collision forgery by passive attacks* if no collision-forgery strategy against it exists. However, to model an active attacker that could compromise various users and obtain their secrets, we must also allow oracle queries to the $\mathsf{Extract}(\cdot)$ algorithm as part of the collision-forgery strategy. (Clearly, querying $\mathsf{Extract}(\cdot)$ on the identity string of the target is not permitted.)

**Definition 2.** (Resistance to collision forgery by active attacks): *Let $S$ be a target identity string. Let $m$ be a target message. The chameleon hashing scheme is secure against (existential) collision forgery by active attacks if, for all four-tuples of non-constant polynomials $f_1(\cdot)$, $f_2(\cdot)$,$f_3(\cdot)$, $f_4(\cdot)$, and for large enough $k$, there is no probabilistic algorithm $A$ which runs in time less than $f_1(k)$, makes at most $f_2(k)$ queries to an* $\mathsf{Extract}(\cdot)$ *oracle (on identity strings other than $S$), and succeeds with probability larger than $\frac{1}{f_3(k)}$ in computing integers $r$ and $r'$, and binary string $m'$, where $m' \neq m$, such that* $\mathsf{Hash}(S, m, r) = \mathsf{Hash}(S, m', r')$*, where* $\mathsf{Hash}(\cdot)$ *is an instance of the scheme with security parameter $f_4(k)$.*

The chameleon hashing scheme must also be semantically secure. ¿From the hash value it is infeasible to determine which message is likely to have resulted in such value by an application of the hash algorithm.

**Definition 3.** (Semantic security): *The chameleon hashing scheme is said to be semantically secure if, for all identity strings $S$, and all pairs of messages $m$ and $m'$, the probability distributions of the random variables* $\mathsf{Hash}(S, m, r)$ *and* $\mathsf{Hash}(S, m', r)$ *are computationally indistinguishable.*

Note that the semantic security of a chameleon hashing scheme implies the non-transferability of the corresponding chameleon signature scheme.

## 3 The Scheme

Let $\tau$ and $\kappa$ be security parameters. The scheme fixes a secure hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^{2\tau}$, mapping strings of arbitrary length to strings of fixed length $2\tau$. Reasonable choices for an implementation of the scheme are $\tau = 80$ and SHA-1 for $\mathcal{H}(\cdot)$.

The setup algorithm is similar to an RSA key generation step. The trusted party $\mathsf{T}$ generates two safe prime numbers $p$ and $q$ in the set $\{2^{\kappa-1}, \ldots, 2^{\kappa} - 1\}$. (A prime $p$ is said to be *safe* if $p = 2p' + 1$ with prime $p'$.) Let $n = pq$. The bit-length of $n$, $\ell(n)$, is no less than $2\kappa$. Let $\mathcal{C} : \{0,1\}^* \to \{0, \cdots, 2^{2\kappa-1}\}$ be a secure deterministic hash-and-encode scheme mapping arbitrary bit-strings to integers less than $n$. For instance, it is possible to use the deterministic version of EMSA-PSS encoding defined in [11,12][1].

$\mathsf{T}$ then generates a random integer $v$ s.t. $v > 2^{2\tau}$, and such that neither $p'$ nor $q'$ divide $v$. To simplify the protocol description and analysis we assume that $v$ is a prime. In particular, $v$ is odd, so it has no common factors with $\phi(n) = (p-1)(q-1)$. Applying the extended Euclidean algorithm for the GCD, $\mathsf{T}$ computes $w$ and $z$ such that $wv + z(p-1)(q-1) = 1$.

$\mathsf{T}$'s public key is $(n, v)$. Its secret key is $(p, q, w)$.

We can now describe the extraction algorithm. Let $S$ be the identity string associated to some party. First we apply the deterministic hash-and-encode scheme to obtain the element $J = \mathcal{C}(S)$ in $\mathbf{Z}_n$. The secret key is extracted as $B = J^w \bmod n$. Notice that being able to compute $B$ from $S$ should be infeasible. In particular, if $\mathcal{C}$ is chosen as the EMSA-PSS encoding, then $B$ is a *secure* RSA signature on the string $S$, under the public key $(n, v)$.

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

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

where, again, $\mathcal{H}(\cdot)$ is the secure hash function, and $J = \mathcal{C}(S)$.

[1] Clearly, any deterministic encoding method would suffice. However, EMSA-PSS is recommended because the security of its deterministic version (when `seedLen=0` [11]) is shown to be *loosely* reducible to the security of the RSA function in the random oracle model.

The $\mathsf{Forge}$ algorithm is:

$$\mathsf{Forge}(S, B, m, r, h, m') = r' = rB^{\mathcal{H}(m)-\mathcal{H}(m')} \bmod n.$$

Note indeed that

$$\begin{aligned}\mathsf{Hash}(S, m', r') &= J^{\mathcal{H}(m')}r'^{v} \\ &= J^{\mathcal{H}(m')}r^{v}B^{v(\mathcal{H}(m)-\mathcal{H}(m'))} \\ &= J^{\mathcal{H}(m')}J^{vw(\mathcal{H}(m)-\mathcal{H}(m'))}r^{v} \\ &= J^{\mathcal{H}(m')}J^{\mathcal{H}(m)-\mathcal{H}(m')}r^{v} \\ &= J^{\mathcal{H}(m)}r^{v} \\ &= \mathsf{Hash}(S, m, r).\end{aligned}$$

### 3.1 Security Analysis

As remarked above, the extraction algorithm requires knowledge of the trapdoor; obtaining the secret key from the private key without knowledge of the trapdoor requires forgery of a *secure* RSA signature. We now extend this argument to the other security requirements.

**Theorem 1.** *The chameleon hashing scheme is resistant to forgery under active attacks, provided that the* secure *RSA signature scheme is similarly resistant.*

The proof is simple, so we sketch it here. Given a collision, $\mathsf{Hash}(S, m, r) = \mathsf{Hash}(S, m', r')$, it is possible to extract the secret key $B$ associated to the public key $J = \mathcal{C}(S)$.

$$J^{\mathcal{H}(m)}r^{v} = J^{\mathcal{H}(m')}r'^{v} \Longrightarrow (r'/r)^{v} = J^{\mathcal{H}(m)-\mathcal{H}(m')} \Longrightarrow r'/r = B^{\mathcal{H}(m)-\mathcal{H}(m')}. \tag{1}$$

Now, each hash value is the binary representation of a positive integer of value smaller than $2^{2\tau}$, and so the absolute value of $\Delta = \mathcal{H}(m) - \mathcal{H}(m')$ is also smaller than $2^{2\tau}$. Since $v$ is a prime integer larger than $2^{2\tau}$, it follows that these values are relatively prime, i.e. $\gcd(\Delta, v) = 1$. Using the Extending Euclidean algorithm for the GCD, one computes $\alpha$ and $\beta$ such that $\alpha\Delta + \beta v = 1$. $B$ can now be extracted:

$$(r'/r)^{\alpha}J^{\beta} = B^{\alpha\Delta+\beta v} = B. \tag{2}$$

Recall that he secret key $B$ is a *secure* RSA signature on the identity string $S$, as remarked above. Hence, being able to compute collision forgeries for target strings implies in the capability of computing *secure* RSA signatures on messages of choice. $\square$

**Theorem 2.** *The chameleon hashing scheme is semantically secure.*

This is because all elements of the RSA ring are $v$-powers when $v$ is relatively prime to the factors of the modulus $n$. Thus, given a hash value $z$ and any message $m$, there is exactly one random integer $r$, with $0 \le r < (p-1)(q-1)$, such that $\mathsf{Hash}(S, m, r)$ equals $z$, namely $r$ is equal to the $v$-th root of $zJ^{-\mathcal{H}(m)}$.$\square$

## 4 ID-based Chameleon Signatures

Our hash function shares all the benefits of any ID-based construct (such as ID-based encryption or signature schemes), in particular:

- There is no need to retrieve the certificate of the intended recipient. The hash function can be computed from publicly available identity information (e.g., the recipient's email address).
- The hash function can be computed under a recipient's identity even if such recipient does not even exist or will join the system at a later time.
- Exposure of secret keys can be minimized by customizing the identity information under which the hash is computed (e.g., by concatenating time information to the identity of the recipient).

A chameleon signature scheme ([1]) is a signature computed over a chameleon hash function. The recipient can verify that the signature of a certain message $m$ is valid but cannot prove to others that the signer actually signed $m$ and not another message. Indeed, the recipient can find collisions of the chameleon hash function, thus finding a message different from $m$ which would pass the signature verification procedure.

An ID-based chameleon signature on $m$ consists of a traditional signature scheme, such as RSA or DSA, computed over the ID chameleon hash of $m$ under the hashed identity $J$ of the intended recipient. In particular, the signer generates a random $r$ and computes:

$$S_{ID} = \{Sign(S, \mathsf{Hash}(S, m, r)), m, r\},$$

where $\mathsf{Hash}(S, m, r) = J^{\mathcal{H}(m)} r^v \bmod n$ is the id-based chameleon hash function computed for $J = \mathcal{C}(S)$.

As suggested in [1], an authenticated description of the hash function should also be added whenever such a description is not publicly available. As already mentioned, $Sign(\cdot)$ can be any standard signature scheme. In particular, it can also be implemented as any identity-based signature scheme, such as the Fiat-Shamir [8] or Guillou-Quisquatter [13]. This way, the verification process would be also ID-based since the signature can be verified from the identity of the signer.

The above signature could be repudiated by a malicious signer who claimed that the recipient had forged the signature. But a trusted third party, or judge, could intervene to settle the dispute by asking the signer to provide a pair of values, different from $(m, r)$, which would pass the signature verification procedure. If the signer does not provide such a pair then the signature on $m$ is considered valid. If the signer does provide a different pair $(m', r') \neq (m, r)$, which passes the signature verification procedure, then the judge can conclude that the recipient has cheated and the signature on $m$ is marked as invalid.

As with all identity-based schemes, only the trusted third party can extract the secret key – the value $B$ such that $J = B^v \bmod n$. One fundamental feature of an identity-based chameleon signature scheme, computed under a hashed identity $J$, is that the recipient does not have to know the secret $B$ unless he wants to forge the signature. Interestingly, this adds a new flavor to the identity-based cryptography in general: *The user does not have to retrieve the secret to terminate a transaction unless he wants to diverge from the basic scheme.* This is somehow different from the identity-based encryption where one has to necessarily retrieve the secret in order to decrypt messages or from identity-based signatures where one has to have the secret in order to start signing messages. In our case, the recipient may never ask for the secret but still successfully complete all transactions. However, the recipient can *potentially* obtain the secret from the trusted third party at any time which makes the original signature non-transferable. (A verifier does not know whether the recipient queried the trusted third party or not.)

## 4.1 Message Hiding

When a dispute takes place, it is often desirable to protect the confidentiality of the original message even against the judge. As suggested in [1], whenever the recipient cheats, the judge can solve any dispute without knowing the message originally signed by the signer. Indeed, it would be enough to reveal any collision of the chameleon hash function to convince the judge that the recipient cheated. This can be easily accomplished since the secret trapdoor information associated to the recipient's public key is revealed whenever a collision of the chameleon hash function is known. For instance, suppose the signer computes the signature on the pair $(m, r)$ and the recipient finds a collision and claims that the signature is on $(m', r')$. During the dispute resolution, the judge reveals $(m', r')$, enabling the signer to compute the trapdoor, as follows: First, the signer calculates $r'/r = B^{\mathcal{H}(m)-\mathcal{H}(m')}$. Then, $B$ is extracted given that $J = B^v$ is known and that $GCD(\mathcal{H}(m) - \mathcal{H}(m'), v) = 1$ always holds. For details, see section §3.1 and the proof of Fact 1.

Once the signer knows $B$, she can use the recipient's $\mathsf{Forge}(\cdot)$ algorithm to compute a new collision $(m'', r'')$ and provide it to the judge, proving that the recipient cheated while keeping the original pair $(m, r)$ private.

## 4.2 Customized Identities

Notice that anyone knowing a collision can find the recipient's secret $B$. Similarly, this happens with the original scheme in [1] where the secret key corresponding to the public key used for the hash computation is revealed. This works as a deterrent to forging. However, if the recipient is unwilling to forge a signature, in order to avoid exposing his secret key, the non-transferability property of chameleon signature schemes is somewhat weakened. Third-party verifiers may be inclined to consider valid any signatures proposed by the recipient, upon knowing his hesitation in forging them. Would a recipient forge a signature when this might jeopardize all other transactions involving his public key? The identity-based chameleon hash function offers a natural way to solve this 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 $B$ will be provided only to the recipient, identified as the person whose identity prefixes the string used to form the public key $J$.

Whenever public keys are employed, one should in principle check whether a public key has been revoked or not. The use of properly customized identities eliminates this need. If the customized identity is unique to each message signed, the signer is certain that the corresponding secret has never been exposed since it has not even been computed by the trusted authority. In this regard, we stress again that the recipient does not have to retrieve the secret key to verify the signature. 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.

## 4.3 Message Recovery

In case of forgery, the recipient's key compromise results in the signer being capable of claiming any message as the one originally signed. Moreover, it becomes impossible for the signer to prove which message was the original one. In some applications, this may not be an acceptable outcome. Circumstances may be such that the signer has an interest in being capable to affirm the original signature if she so desires. This situation has been addressed in [1], and the solution can be applied to our scheme. The *convertible* variant of the basic scheme provides the signer with a non-interactive algorithm to transform any instance of the chameleon signature into a universally verifiable instance.

A different possibility is that the application requires that the original message be recoverable even without the signer's cooperation. In these cases, it may be necessary to add to the signature some additional information about the message itself. One possibility is to include in the signature an encryption of the pair $(m, r)$ computed under the public key of the judge. That is, the signature becomes:

$$S_{ID} = \{Sign(S, \mathsf{Hash}(S, m, r), PK[(m, r)]), m, r\},$$

where the public-key encryption is assumed to be semantically secure. [2] In practice, one would sign a hash of the encryption, so as to compute the signature on parameters of fixed size.

The above construction eliminates the need for the signer to store signatures in order to repudiate forgeries, shifting the storage burden onto the recipient, as in regular (digital) signatures. With this scheme, the signer is protected against a malicious recipient that produces a forgery and then prevents the signer from participating in the adjudication of the claim – a denial-of-service attack on the signer. This scheme 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. It only protects the recipient against a later "change of heart" by the signer.

## 5 Sealed-Bid Auction System

In this section, we describe a very simple sealed-bid auction scheme based on ID-based chameleon signatures. We stress that this application is not meant to be the sole motivation for the primitive we are introducing, but only an illustration of some of its potential uses and benefits.

In public seller auctions, also called English or forward auctions, bidders compete by announcing successive bids, with the highest offer acquiring the good being auctioned. Alternatively, public buyer auctions (also called reverse auctions) are used by the Government and by businesses to procure supplies and services. In reverse auctions the lowest bid wins.

By contrast, in sealed-bid auctions (whether forward or reverse), bidders are allowed to submit a single, sealed bid. The seller stipulates a period for accepting bids, at the end of which the bids are unsealed and compared. The winning bid is revealed so that all competitors can ascertain its winning merits.

The Internet has transformed the auctioning business. Auction systems were some of the earliest success stories in the commercial Internet, and continue to grow profitably. While the site eBay© is a household name, multiple other sites cater to the vast segment of business-to-business procurement. The auction type (forward/reverse) varies but proxy bidding is commonly found. Proxy bidding-based auctions are intermediate between open and sealed-bid auctions. As in public auctions, the current winning price is openly determined, constantly changing while the auction is ongoing. However, in similarity with closed auctions the maximum bid (or minimum bid in a reverse auction) is not known to competitors.

Proxy bidding has a few characteristics that makes it less desirable than sealed-bid auctions, at least in some circumstances. For instance, late bidders

[2] This is required to maintain the non-transferability of the signature once augmented by the encrypted message. Informally, a semantically secure encryption is a randomized encryption that protects all the bits of the plaintext. In particular, if one of two publicly known messages is encrypted, no-one can tell which of the two was encrypted by just looking at the ciphertext.

(those who submit their bids closest to the auction closing time) are at an advantage, raising both questions of fairness to all bidders and of possible underestimation of the market value of the product being auctioned, which is unfair to the seller. As the value, variety, and complexity of items auctioned online increase, it is possible that some markets will favor auction systems that optimize for factors such as efficiency and fairness, for instance by featuring sealed-bid auctions [14].

## 5.1 Closed auction system models

Standard models for sealed-bid auction systems require that the following properties hold:

**Correctness:** The correct winner and clearing price are determined in each auction.

**Confidentiality:** Bids remain hidden before the action ends.

**(Optionally):** Only the winning bid and the clearing price are revealed. Losing bids remain private.

**Fairness:** Bidders do not learn information about competing bids before the auction closes, neither can they change their bid after the closing of the auction.

**Non-repudiation:** The winner cannot repudiate his bid.

**Robustness:** No party, whether or not a legitimate party in the auction, may maliciously or accidentally compromise the correct functioning of the system.

There has been considerable research on secure, closed e-auction systems, such as [15–21]. Most of these systems are fairly complex, employing techniques such as secure multi-party evaluation of predicates on encrypted bids. The general idea is that the bidders submit encrypted bids. The auctioneer, or other bidders, cannot determine from the encrypted values the original bids, but can cooperate to determine the highest bid value and the auction winner. It is necessarily true that only systems based on encrypted bids can achieve the strongest possible security guarantees. In particular, without keeping the bids encrypted before the auction closing time, it is not possible to provide fairness without making some additional trust assumptions, because if a bid is ever available as clear-text its content could be communicated to competitors. Similarly, losing bids must never be decrypted if full privacy is desired.

In practical applications one must consider the optimal trade-off between security properties and other worthwhile goals such as lower communication complexity (reducing the number of messages sent greatly simplifies system design as a whole), computational costs and administrative ones (such as key management issues). Our approach considers the design of a system that is extremely practical. The design goals are as follows:

**Thin clients:** The client software/hardware (assisting the bidder function) should be of minimal complexity. Optimally, it should consist of a non-customized (or minimally customized) web browser or even a text-messaging application.

**Stateless clients:** A stateless protocol is much easier to implement correctly and to analyze for its security properties. Moreover, less state information results in greater robustness.

**Low communication:** The communication complexity of a distributed protocol can be defined either as the number of bits sent or the number of messages sent. From a practical perspective, the number of messages influences the complexity of protocol implementation and analysis, while the number of bits is important whenever network bandwidth is at a premium.

**Asynchronous communication model:** The only timeliness constraint is that auctioneer and bidders can agree on whether a bid is submitted after or before the closing of an auction.

**Auction rule flexibility:** This is an important requirement for a general-purpose auction system. Different auction styles require different methods to determine the clearing price for the item(s). For instance, a $(k+1)$-auction is one in which $k$ items are auctioned. The $k$-th highest bidders claim items, all at the same clearing price, which corresponds to the $k+1$-st highest bid.

## 5.2 The auction scheme

We present a sealed-bid auction protocol satisfying all the above design goals, while providing only partially the security guarantees of the general model. We first describe a straightforward protocol, which provides non-repudiation:

1. *Each bidder sends a single signed message to the auctioneer, containing the bid information.*
2. *The auctioneer publishes a receipt for each bid received.*
3. *The auctioneer publishes the winning bid. (After auction closes.)*

The above protocol also reaches correctness, if the following conditions hold:

– All bids submitted before the auction closes are accepted by the auctioneer. In particular, this implies that all parties can agree on the auction closing time – a soft synchronization requirement.
– *Consistency of view* of the published information must be assured.

We remark that both conditions are required of any *robust* protocol. We assume that the underlying communication channel between bidders and the auctioneer is such that both properties above hold, as the techniques for achieving such guarantees are well understood and can be achieved in practice. Under such assumption, the above protocol is robust.

On the other hand, this protocol achieves neither *fairness* or *privacy* unless the auctioneer can be trusted not to publish any undue information. As alluded to before, fairness and absolute privacy can only be achieved by performing all computations (such as clearing price and auction winner determination) on encrypted bids. Use of encryption increases the communication complexity (once to collect bids and another to perform decryptions) and makes robustness harder to achieve. (For instance, one must ensure that encryption keys can be recovered for the winning bid or else there is no robustness of the non-repudiation guarantee.) This poses the question of whether partial *fairness* and *privacy* guarantees can be achieved without resorting to computation on encrypted information.

Our approach is to simplify the trust model. Clearly, if the auctioneer is entirely trusted then privacy and fairness can be achieved simply by protecting the bid-submission channel. If the auctioneer may arbitrarily collude with bidders then complex cryptographic techniques are required to ensure fairness or privacy. These are not the only two possible trust models. Alternatively, the auctioneer might not be trusted to keep the bids privately and yet also untrusted by dishonest bidders to provide the correct information about competitors' bids. *This model does not protect against auctioneer-bidder collusions but can accommodate situations in which a hacker or malicious insider breaks into the system and tries to peddle the information to a suspectful bidder.*

Notice that the simple scheme described above is still vulnerable under this restricted trust model. An insider could easily convince a suspicious bidder of the correctness of the peddled information because the bids are signed. However, if the bids were to be signed using a chameleon-hashing scheme that would not be the case. As the signature on bids are non-transferable, the suspicious insider cannot be assured of the correctness of the reported bid value. We now describe this in some detail. The final protocol is simply:

1. *Each bidder sends a single signed message to the auctioneer. The message contains the bid terms, signed using the identity-based chameleon signature scheme.*
2. *The auctioneer publishes a receipt for each bid received.*
3. *The auctioneer publishes the winning bid. (After auction closes.)*

The bids are submitted to the auctioneer in the clear, but since they are signed using chameleon signatures, their validity can only be determined by the auctioneer. The auctioneer publishes the signatures on each bid, which acts as a receipt on the bid, but does not publish the value of the bids themselves. If the auctioneer were to cheat and reveal to a bidder some of the competitors' bids, that bidder would have no means to determine whether or not the auctioneer is telling the truth. Yet, since chameleon signatures are non-repudiatable, the bid winner must make good on her offer: In case of disputes, a judge would determine the validity of the bid.

An important point is that it be believable that the auctioneer might cheat. If regular chameleon signatures are used, it is unlikely that an auctioneer would cheat by forging bids because, by cheating, it would reveal a collision of the chameleon hash function. This in turn discloses its long-term secret key, affording the colluding bidder to repudiate his bid during any later auctions as well as during the current one. However, the non-transferability of a chameleon signature comes from the fact that the recipient of such a signature may be willing to forge it. If such an event is unlikely then the chameleon signature behaves, de facto, much like a traditional signature scheme.

An alternative solution would be for the auctioneer to generate temporary public keys, perhaps one for each item being auctioned. At the highest level of granularity, the auctioneer might issue a different public key for each potential bidder on the item. Clearly, such public keys cannot be pre-computed in large systems with millions of users such as eBay©. That implies that a bidder must first contact the auctioneer about a particular auction and request the respective key. While correct, this solution requires a second round of communication when compared to the previous one. Instead, an identity-based scheme allows the bidder to generate a signature under a customized hashed identity $J$ of the following form:

$$J = \mathcal{C}(auctioneer\_id \mid\mid bidder\_id \mid\mid item\_id).$$

This can be done without interacting with the auctioneer and would offer the highest level of granularity. The auctioneer does not have to retrieve the secret corresponding to $J$ unless he wants to forge the signature. In case of forgery and subsequent dispute, only the secret related to a particular transaction from a specific bidder is revealed, leaving the rest of the transactions valid. This makes any signed bids completely useless for other bidders and non-transferable.

Our identity-based scheme allows users to participate in digital auctions that interface with real world. For instance, items for bidding can be labeled by reference strings. Unlike regular public keys, reference strings can be mnemonic, facilitating entry by potential buyers. For instance, realtors could put a sign in front of houses available on the market with a realtor identity and a number identifying the property. Auctioneers could advertise items for auction in public newspapers, including item reference strings. (Today, in the US, government-seized property auctions and foreclosure-related auctions must be advertised in newspapers to comply with the law.) Potential buyers would view the property and could type that information into an e-mail or text-messaging capable computing device to submit a bid. Not needing to download a public key simplifies the process, adding convenience: System users would be able to submit bids from low-bandwidth communication devices such as cell phones and PDAs. The above scheme is even more convenient if clients, such as cell phones, take advantage of an architecture of strong authentication (permitted by the device smartcard, in the case of cell phones) to eliminate the need for dedicated cryptographic software and key management on the client side. The bids could be forwarded to a proxy signature server (for instance, the site of the mortgage banker) which would chameleon-sign the bid on behalf of the user.

To summarize we have designed a very simple e-auction protocol that achieves correctness and non-repudiation in all cases, and which achieves fairness and privacy only under certain trust assumptions: The auctioneer is allowed to malfunction or to be dishonest, and the same is allowed of bidders, but the model does not handle the case when bidders and the auctioneer may collude. However, the simplicity and convenience afforded by the proposed scheme would be of considerable practical significance. In particular, we would like to point out the statelessness of clients and the fact that protocol requires the sending of a *single message* by bidders. Notice that the identity-based properties of the chameleon signature are essential to enable the single-message feature within this trusted model: Alternative methods would require schemes for delivering temporary keys, for instance. The use of an identity-based scheme eliminates the need of a certificate distribution architecture.

Apart from (and partly because of) being very efficient, and achieving the lowest possible message complexity, the protocol is quite robust. We believe that the importance of robustness has been overlooked in some of the existing protocols. Without robustness, the strong fairness and privacy conferred by some of the protocols based on encryption cannot be achieved in practice, because in any realistic implementation one must consider how a protocol performs under attacks or when parties malfunction.

## 6 Conclusions

Krawczyk and Rabin introduced in [1] the concept of chameleon hashing. In this paper, we extended their work by introducing an ID-based chameleon hash function. ID-based cryptography in general enjoys advantages relative to key distribution over conventional schemes. In the case of chameleon hashing these advantages are multiplied by the fact that the owner of a public key does not necessarily need to retrieve the associated secret key. Therefore, ID-based chameleon hashing can support single-use public keys very efficiently. We exploit this feature to design a novel application of chameleon signatures to e-auction schemes that enjoys efficiencies difficult to achieve with other cryptographic techniques.

**Acknowledgments.** Many thanks to the anonymous referees for their insightful comments. This work was partly funded by a NSF grant.

## References

1. Krawczyk, H., Rabin, T.: Chameleon signatures. In: Proceedings of NDSS 2000. (2000) 143–154
2. Chaum, D., Antwerpen, H.: Undeniable signatures. In: Advances in Cryptology – CRYPTO'89. Volume 435 of LNCS., Springer-Verlag (1991) 212–216
3. Chaum, D.: Zero-knowledge undeniable signature. In: Advances in Cryptology – EUROCRYPT'90. Volume 473 of LNCS., Springer-Verlag (1990) 458—464
4. Boyar, J., Chaum, D., Damgård, I.B., Pedersen, T.P.: Convertible undeniable signatures. In: Advances in Cryptology – CRYPTO'90. Volume 537 of LNCS., Springer-Verlag (1990) 189–205

5. Chaum, D., van Heijst, E., Pfitzmann, B.: Cryptographically strong undeniable signatures, unconditionally secure for the signer. In: Proc. of Advances in Cryptology – CRYPTO'91. Volume 576 of LNCS., Springer-Verlag (1991) 470–ff
6. van Heijst, E., Pedersen, T.: How to make efficient fail-stop signatures. In: Proc. of Advances in Cryptology – EUROCRYPT'92. Volume 658 of LNCS., Springer-Verlag (1993) 366–377
7. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: Proc. of Advances in Cryptology – EUROCRYPT'96. Volume 1070 of LNCS., Springer-Verlag (1996) 143–ff
8. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Advances in Cryptology – CRYPTO'84. Volume 196 of LNCS., Springer-Verlag (1984) 47–53
9. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Proc. of CRYPTO'01. Volume 2139 of LNCS. (2001) 213 ff.
10. Cocks, C.: An identity based encryption scheme based on quadratic residues. (http://www.cesg.gov.uk/technology/id-pkc/media/ciren)
11. 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)
12. RSA Labs: RSA Cryptography Standard: EMSAPSS – PKCS#1 v2.1. (2002)
13. Guillou, L.C., Quisquater, J.J.: A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmition and memory. In: Proc. of Advances in Cryptology – EUROCRYPT'88. Volume 330 of LNCS., Springer-Verlag (1988) 123–128
14. Chin, S.: Chip trade group calls for better governing of reverse auctions. In: `http://www.ebnonline.com/showArticle.jhtml?articleID-=12803165`. EBN Online (2003)
15. Franklin, M., Reiter, M.: The design and implementation of a secure auction service. In: Proc. IEEE Symp. on Security and Privacy, Oakland, CA, IEEE Computer Society Press (1995) 2–14
16. Lipmaa, H., Asokan, N., Niemi, V.: Secure Vickrey auctions without threshold trust. In: Proc. of the 6th Annual Conference on Financial Cryptography. (2002)
17. Baudron, O., Stern, J.: Non-interactive private auctions. In: LNCS. Volume 2339., Springer-Verlag (2002) 364 ff
18. Harkavy, M., Tygar, J.D., Kikuchi, H.: Electronic auctions with private bids. In: 3rd USENIX Workshop on Electronic Commerce. (1998) 61–74
19. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: Proc. of the 1st conf. on Electronic Commerce, ACM (1999) 129–139
20. Cachin, C.: Efficient private bidding and auctions with an oblivious third party. In: 6th ACM Conference on Computer and Communications Security (CCS), ACM Press (1999) 120–127
21. Kikuchi, H.: $(m+1)$st-price auction protocol. In Syverson, P., ed.: Financial Cryptography – Fifth International Conference. LNCS, Springer-Verlag (2002)
22. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Advances in Cryptology – CRYPTO'86. Volume 263 of LNCS., Springer-Verlag (1987) 186–194