One-Time Signatures and Chameleon Hash Functions

Payman Mohassel

Computer Science Department, University of Calgary
pmohasse@cpsc.ucalgary.ca

Abstract. In this work we show a general construction for transforming any *chameleon hash function* to a *strongly unforgeable one-time signature* scheme. Combined with the result of [Bellare and Ristov, PKC 2007], this also implies a general construction of strongly unforgeable one-time signatures from Σ-protocols in the standard model.

Our results explain and unify several works in the literature which either use chameleon hash functions or one-time signatures, by showing that several of the constructions in the former category can be interpreted as efficient instantiations of those in the latter. They also imply that any "noticeable" improvement to the efficiency of constructions for chameleon hash functions leads to similar improvements for one-time signatures. This makes such improvements challenging since efficiency of one-time signatures has been studied extensively.

We further demonstrate the usefulness of our general construction by studying and optimizing specific instantiations based on the hardness of factoring, the discrete-log problem, and the worst-case lattice-based assumptions. Some of these signature schemes match or improve the efficiency of the best previous constructions or relax the underlying hardness assumptions. Two of the schemes have very fast signing (no exponentiations) which makes them attractive in scenarios where the signer has limited computational resources.

Keywords: One-time Signatures, Chameleon Hash Functions, Strong Unforgeability, Identification Schemes.

1 Introduction

One-time signature (OTS) schemes are digital signatures that can be used to sign a single message for each pair of verifiction/signing key. Despite their limited functionality, OTS schemes have found numerous applications. In fact, earlier constructions of standard signature schemes, use one-time signatures as their main component [21,22,28,30].

OTS schemes are used as building blocks in many cryptographic constructions such as the (i) design of online/offline signature schemes [18], (ii) design of CCA-secure public key encryption from identity-based encryption [8], (iii) transformation of standard signature schemes to those with *strong* unforgeability properties [26,5], and (iv) secure composition of multiple encryption schemes [17].

A. Biryukov, G. Gong, and D.R. Stinson (Eds.): SAC 2010, LNCS 6544, pp. 302–319, 2011.

Finally, OTS schemes are directly employed in networks protocols, for example to authenticate messages in sensor networks [16] or to provide source authentication in multicast and broadcast networks [35].

The earlier constructions of one-time signatures built such schemes from general assumptions such as the existence of one-way functions [28,6,18]. The main drawback of this family of constructions is that they produce very large signatures. In many cases, this has led researchers to search for alternative methods that avoid the use of OTS schemes. The following are just a few instances: IBE to IND-CCA PKE transformations [9,1], online/offline signatures [39], and transformations for strongly unforgeable signatures [40].

Chameleon hash functions (trapdoor hash functions), in particular, have proven to be an extremely useful tool in such scenarios. Several of the examples we gave above take advantage of chameleon hash functions in their constructions. Roughly speaking, chameleon hash functions [27] are randomized collision-resistant hash functions with the additional property that given a *trapdoor*, one can efficiently generate collisions. More specifically, each function in the family is associated with a pair of public and private (trapdoor) keys with the following properties (i) anyone who knows the public key can compute the associated hash function, (ii) for those who do not know the trapdoor the function is collision resistant in the usual sense, and (iii) the holder of the trapdoor information can easily find collisions for every input. As described in more detail in Section 4 several constructions of chameleon hash functions based on standard number theoretic assumptions are known [27,39,3]. Chameleon hash functions are also closely related to the notion of chameleon commitment schemes originally introduced by Brassard *et al.* [13].

1.1 Our Contribution

We design a black-box construction for transforming any chameleon hash function to a strongly unforgeable one-time signature scheme. Such a connection between one-time signatures and chameleon hash functions is not surprising. On the contrary, as alluded to earlier, the knowledge of a close relation between the two primitives seems to have been "in the air". For example, in [23] Groth shows how to design a DL-based one-time signature from the pedersen trapdoor commitment. However, it is not clear how to generalize his construction to work for arbitrary trapdoor commitments.

Our work appears to be the first to formally study this relation by designing a black-box transformation from one primitive to another, and exploring new and useful implications of this transformation. Next, we elaborate on these implications:

Unifying and explaining the previous work. Our general construction confirms that the usefulness of chameleon hash functions in replacing OTS schemes is not a coincidence. Particularly, in several instances, constructions in one work can be interpreted as efficient instantiations of those in another. Here is one example:

Starting with the work of Boneh *et al.* [10], several works in the literature studied constructions for transforming any standard unforgeable signature scheme to a *strongly* unforgeable scheme. While Boneh *et al.*'s construction only works for signature schemes with a special partitioning property, the later works of [26,40,5] develop techniques that work for any UF-CMA secure signature scheme. The work of [26] and [5] use strongly unforgeable one-time signatures while the work of [40] takes advantage of a chameleon hash function. Our results demonstrate that the construction of [40] can be interpreted as an efficient instantiation of the constructions of [26,5].

OTS schemes from identification protocols. It is well-known how to design standard signature schemes from identification schemes in the *random oracle model* via the fiat-shamir transform [19]. The only analogous result we know of in the standard model is the work of Bellare and Shoup [5] who show a construction of one-time signatures from ID schemes.

When we combine our general construction for OTS with the work of Bellare and Ristov [3] who design chameleon hash functions from three move ID schemes (Σ-protocols), we get a general construction of strongly unforgeable OTS schemes from any Σ-protocol that satisfies natural security requirements. Compared to the work of Bellare and Shoup [5], our security requirements appear to be more modest. Particularly, while there exist efficient Σ-protocols that satisfy our security requirements under assumptions such as the hardness of factoring or RSA inversion, the same is not known to be true about their work which requires the ID scheme to be secure against concurrent attacks.

When instantiated based on specific identification schemes, our transformation also leads to several constructions of one-time signatures.

New OTS schemes based on standard assumptions. We further study and optimize several instantiations of our general construction based on standard assumptions such as the hardness of factoring integers, the discrete-log problem and the worst-case lattice-based assumptions.

Our new factoring-based construction has a very fast signing algorithm that only involves a modular addition and multiplication. This is a useful property in scenarios where the signing entity is a low-powered device with limited computational resources, while the verification entity has more computational power. This is a common occurrence in sensor networks, MANETS and VANETs. As discussed in detail in Section 4, our construction appears to be the first scheme with such properties, based solely on the hardness of factoring RSA integers.

Our new discrete-log based construction (described in Appendix B also has a very fast signing. It improves the key size compared to the DL-based construction of Bellare and Shoup [5], and matches the efficiency of the DL-based fail-stop signature of Van Heyst and Pedersen [42], in almost all respects.

Our new lattice-based construction, has a signature size of $O(k \log k)$ where k is the security parameter. Compared to the signature scheme of Lyubashevsky and Micciancio [29] who design OTS schemes based on hard problems on *ideal lattices*, our signature scheme has the advantage of being based on a potentially

weaker assumption, since we do not pose such restrictions on the structure of the lattice. On the other hand, their scheme provides smaller key sizes and a faster signing algorithm. Compared to the recent construction of [15] who design standard signature schemes based on worst-case lattice-based assumptions, our scheme has significantly shorter signatures. The work of Boyen [11], shows how to improve that by designing a standard signature scheme with signature size comparable to ours, but the verification key size is still significantly longer than that of ours.

Chameleon hash functions from CRHFs? An interesting open question is whether we can build chameleon hash functions from standard collision-resistant hash functions (CRHFs), and other symmetric-key primitives (e.g. can one use a hash function from the SHA family as the CRHF?). Such a construction has the potential of being significantly more efficient than the existing ones. Our results relate this question to the design of efficient and short OTS schemes based on the same primitives. More specifically, a new construction for chameleon hash functions based on CRHFs, would automatically lead to short (note that chameleon hash functions are compressing) OTS based on the same primitives. Unfortunately, the latter is a long standing open question.

2 Preliminaries

2.1 Collision and Target Collision Resistance

A hash function is a pair $\mathcal{H} = (K, H)$. The key generation algorithm K returns a key k, and the deterministic hash algorithm H takes k and an input x to return a hash value y. We say that $\mathcal{H}$ is collision-resistant (CR) if for every polynomial-time adversary A, the CR-advantage

$$\mathbf{Adv}^{cr}_{\mathcal{H}}(\mathsf{A}) = \Pr[H(k, x_1) = H(k, x_2) : k \xleftarrow{\$} K; (x_1, x_2) \xleftarrow{\$} \mathsf{A}(k)]$$

of A against $\mathcal{H}$ is negligible. Similarly, we say that $\mathcal{H}$ is target-collision resistant (TCR) if for every polynomial-time adversary A the TCR-advantage

$$\mathbf{Adv}^{tcr}_{\mathcal{H}}(\mathsf{A}) = \Pr[H(k, x_1) = H(k, x_2) : (x_1, st) \xleftarrow{\$} \mathsf{A}; k \xleftarrow{\$} K; x_2 \xleftarrow{\$} \mathsf{A}(k, st)]$$

of A against $\mathcal{H}$ is negilgible. This means that the TCR adversary has to commit to an element in the collision before seeing the hash key. As discussed in [4], TCR has some potential benefits over CR, such as being easier to achieve and allowing for shorter output lengths. In this paper, for the most part we need our hash functions to be target collision resistant. While formally such functions are keyed, for simplicity (and since in practice one often uses non-keyed constructions) we abuse the notation and drop the key for such functions in our constructions.

2.2 Chameleon Hash Functions

A family of chameleon hash functions [27] consists of a tuple of three algorithms $H = (Gen, h, h^{-1})$. The key generation algorithm $(ek, td) \xleftarrow{\$} Gen(1^k)$ outputs a pair of keys named the evaluation key and the trapdoor key, respectively. The evaluation algorithm h takes the evaluation key ek, a message $m \in \mathcal{M}_{ek}$ and randomness $r \in \mathcal{R}_{ek}$ and outputs $h(ek, m, r) \in \mathcal{Y}_{ek}$.

Chameleon property. The chameleon property requires that h^{-1} on inputs the trapdoor key td, messages $m, m' \in \mathcal{M}_{ek}$ and randomness $r \in \mathcal{R}_{ek}$, returns $r' \leftarrow h^{-1}(td, m, r, m')$ such that $h(ek, m, r) = h(ek, m', r')$.

Uniformity property. The uniformity property guarantees that there exists a distribution over $\mathcal{R}_{ek}$, which we denote by $D_{\mathcal{R}_{ek}}$, such that for all $m \in \mathcal{M}_{ek}$, the distributions $(ek; h(ek, m, r))$ and $(ek; y)$ are computationally indistinguishable, where $(ek, td) \xleftarrow{\$} Gen(1^k)$, $r \xleftarrow{\$} D_{\mathcal{R}_{ek}}$ and y is chosen uniformly from $\mathcal{Y}_{ek}$. We note that for all existing constructions of chameleon hash functions, the uniformity property actually holds *statistically.*

Collision resistance. Finally, we require that given H and the evaluation key ek, it is hard to compute $(m, r) \neq (m', r')$ such that $h(ek, m, r) = h(ek, m', r')$. More formally we have:

$$\Pr\left[\ (m,r) \neq (m',r') \wedge h(ek,m,r) = h(ek,m',r') \ :\ (ek,td) \xleftarrow{\$} Gen(1^k)\ ;\ (m,r,m',r') \xleftarrow{\$} \mathsf{A}(ek,H)\ \right]$$

is negligible for any probabilistic polynomial time adversary A.

It is possible to weaken the collision resistance property by only requiring that $m \neq m'$. However, all the instantiations of chameleon hash functions we know of possess this stronger property. Furthermore, we need this version of collision resistance in order to achieve signature schemes that are *strongly* unforgeable.

2.3 Security Definitions for Signature Schemes

We introduce the necessary definitions for signatures schemes in Appendix A.

3 One-Time Signatures from Chameleon Hash Functions

In this section we show general constructions of one-time signatures from chameleon hash functions. First we design an efficient one-time signature scheme that is only secure against apriori message attacks. As we will discuss later, it turns out that this level of security is sufficient for a number of applications of one-time signatures in the literature. Then, we show how to enhance the construction to achieve security against adaptively chosen message attacks.

3.1 A suf-ama Signature Scheme

We start with a simple construction of a one-time SUF-AMA signature scheme from a chameleon hash function.

Construction 31. *Let $H = (Gen, h, h^{-1})$ be a family of chameleon hash functions. We construct a one-time* SUF-AMA *signature scheme OTS in the following way:*

- **Key Generation:**
  1. *Compute $(ek, td) \xleftarrow{\$} Gen(1^k)$.*
  2. *Choose a uniformly random $r_s \in \mathcal{R}_{ek}$.*
  3. *Output (vk, sk) where $vk = (ek, z = h(ek, m_f, r_s))$ and $sk = (td, m_f, r_s)$. Here m_f can be an arbitrary message from $\mathcal{M}_{ek}$. There is no need to keep m_f secret, but it is also not needed for verification.*
- **Signing:** *On input message m, compute and return the signature $\sigma = h^{-1}(td, m_f, r_s, m)$.*
- **Verification:** *On input (m, σ), accept if $h(ek, m, \sigma) = z$ and reject otherwise.*

Claim. If H is a family of chameleon hash functions, the OTS scheme described above is a one-time SUF-AMA signature scheme.

Proof. Let A be the adversary that breaks the OTS scheme in the SUF-AMA game. Then, we build an adversary B that breaks the collision resistance of the chameleon hash function H. B is given ek and H. B runs A. A chooses a message m for his signature query. B generates a random $r \in \mathcal{R}_{ek}$, computes $h(ek, m, r)$, and returns $vk = (ek, h(ek, m, r))$ and the signature $\sigma = r$ of message m to A. Note that the uniformity property of the chameleon hash function implies that for any two messages m, m_f the distribution of $h(ek, m, r)$ and $h(ek, m_f, r_s)$ are computationally indistinguishable (from uniform) when r and r_s are chosen uniformly at random from $\mathcal{R}_{ek}$. Hence B successfully simulates A's view in the SUF-AMA game.

Eventually, A returns $(m', \sigma') \neq (m, \sigma)$ as his forgery. B outputs (m, σ) and (m', σ') as his collision pair for the hash function (see the definition of collision resistance in Section 2.2). It is easy to verify that if A is successful in forging a signature for m', B outputs a valid collision.

Remarks on the proof. Note that since B in the above reduction does not know the trapdoor for the hash function, he can only respond to a signature query for an apriori chosen message (before the verification key is fixed). Also note that the reason we only afford one-time security is that given a collision pair for the hash function, all bets about its collision-resistance in the future are off. In fact, for all existing instantiations of chameleon hash functions, given a collision pair one can easily generate many more collisions. Finally, note that the *chameleon property* of the hash function did not play a role in the security proof. Instead, it was needed for the functionality it provides, as it is used in the signing algorithm.

Efficiency and optimizations. The signature consists of a single element in $\mathcal{R}_{ek}$. The signing involves one invocation of the h^{-1} function. As we will see shortly, for a number of instantiations of chameleon hash functions, this leads to very

efficient signing that only includes modular addition and multiplications (no exponentiations). The verification requires one evaluation of the chameleon hash function.

The verification key for the above signature scheme consists of the tuple $(ek, h(ek, m_f, r_s))$. However, we can shorten the verification key via use of a *target collision resistant* hash function. In other words, given a function T from a family of TCR functions, we can let the verification key be $vk = (ek, T(h(ek, m_f, r_s)))$. It is easy to verify that the above proof of security still goes through without any difficulties. Since $h(ek, m_f, r_s)$ is often a group element this simple optimization can lead to a decent improvement in the key size.

Unifying the previous works. The above construction explains and unifies several previous works that use one-times signatures and/or chameleon hash functions. Here we consider two use cases for chameleon hash functions in the literature. The first application is the design of offline/online signature schemes. The earlier work of Even *et al.* [18] used a one-time signature scheme to design an offline/online signature scheme. It is not hard to show that the notion of security they need for their one-time signature scheme is UF-AMA security (not the UF-CMA security). The later work of Tauman and Shamir [39], proposed the use of chameleon hash functions in order to design offline/online signature schemes. Our construction implies that the latter construction can be interpreted as an efficient instantiation of the former.

Another common application of chameleon hash functions is for transforming UF-AMA secure signature schemes to UF-CMA secure signature schemes (e.g. see [25,12]). Again, it is easy to verify that a UF-AMA secure one-time signature can also be used for such a transformation, and that the transformations based on chameleon hash functions can be seen as efficient instantiations of the construction of [18].

3.2 A Strongly Unforgeable uf-cma Scheme

Security against apriori message attacks is not sufficient in all applications of one-time signatures. In several instances, such as the design of IND-CCA PKE from IBE schemes [14] or general transformation of UF-CMA signature schemes to strongly UF-CMA signature schemes [26], the *strong unforgeability* and security against *chosen message attacks* of the OTS scheme seem crucial. Next we show how to enhance the previous construction to design such a signature scheme from chameleon hash functions as well.

Construction 32. Let $H = (Gen, h, h^{-1})$ be a family of chameleon hash functions as defined in section 2 and $T_{ek,ek'} : \mathcal{Y}_{ek} \rightarrow \mathcal{M}' \subseteq \mathcal{M}_{ek'}$ be a target collision resistant hash function, where ek and ek' are two evaluation keys for the chameleon hash family. We construct a one-time SUF-CMA signature OTS in the following way:

- **Key Generation:**
  1. Compute $(ek_i, td_i) \xleftarrow{\$} Gen(1^k)$ for $i \in \{0,1\}$.
  2. Choose uniformly random strings $r_s^0 \in \mathcal{R}_{ek_0}, r_s^1 \in \mathcal{R}_{ek_1}$.
  3. Compute $z_0 = h(ek_0, m_f, r_s^0)$ and $z_1 = T_{ek_1,ek_0}(h(ek_1, m_f, r_s^1))$. Here m_f is an arbitrary message in $\mathcal{M}_{ek_1}$. m_f can potentially be different for different signers. As before, there is no need to keep m_f secret, but it is also not needed for verification.
  4. Output (vk, sk) where $vk = (ek_0, ek_1, z_0)$ and $sk = (td_0, td_1, r_s^0, r_s^1, z_1)$.
- **Signing:** On input message m, return the signature $\sigma = (h^{-1}(td_1, m_f, r_s^1, m), h^{-1}(td_0, m_f, r_s^0, z_1))$.
- **Verification:** On input $(m, \sigma = (r, r'))$, accept if $h(ek_0, T_{ek_1,ek_0}(h(ek_1, m, r)), r') = z_0$. Else, reject.

Claim. If H is a family of chameleon hash functions, and T_{ek_1,ek_0} is a TCR hash function, then the OTS scheme described above is a one-time SUF-CMA secure signature scheme.

Proof. Let A be the adversary that breaks the OTS scheme in the SUF-CMA game. Then we build an adversary B that breaks the collision resistance of the chameleon hash function H. Let A's signature query and answer be $(m, \sigma = (\sigma_0, \sigma_1))$, and denote the forgery made by A with $(m^*, \sigma^* = (\sigma_0^*, \sigma_1^*))$. We divide the proof into two separate parts for two different types of forgers. Note that the two types of forger we consider are complements of each other and therefore partition the space of all possible forgers. Hence, adversary A is a member of exactly one of these two sets. Our adversary B has to randomly guess which type of forger A is going to be. He will be correct with probability 1/2 which leads to a factor of two reduction in tightness of security.

– **Type I Forger.** In this type of forgery we have that either $(m, \sigma_0) = (m^*, \sigma_0^*)$ or $h(ek_1, m, \sigma_0) \neq h(ek_1, m^*, \sigma_0^*)$. In this case B finds a collision for H under the public key ek_0. B is given ek_0.

He generates $(ek_1, td_1) \xleftarrow{\$} Gen(1^k)$ on his own, computes $z_1 = T(h(ek_1, m_f, r))$ and $z_0 = h(ek_0, z_1, r')$ for random $r \in \mathcal{R}_{ek_0}, r' \in \mathcal{R}_{ek_1}$ and sends $vk = (ek_0, ek_1, z_0)$ to A. Note that even though B computes z_1 before seeing the signature query, the *uniformity property* of the chameleon hash function guarantees that A's view is indistinguishable from the real scheme.

A makes a signature query for a message m. B computes $\sigma_0 = h^{-1}(td_1, m_f, r, m)$ and $\sigma_1 = r'$ and returns the signature $\sigma = (\sigma_0, \sigma_1)$ to A. A eventually sends a forgery $m^*, \sigma_0^*, \sigma_1^*$ where either $(m, \sigma_0) = (m^*, \sigma_0^*)$ or $h(ek_1, m, \sigma_0) \neq h(ek_1, m^*, \sigma_0^*)$. If the latter is the case, due to the target collision resistance of T_{ek_1,ek_0}, with all but negligible probability we have that $z_1 \neq z_1^*$ where $z_1 = T_{ek_1,ek_0}(h(ek_1, m, \sigma_0))$ and $z_1^* = T_{ek_1,ek_0}(h(ek_1, m^*, \sigma_0^*))$.

Hence, it is easy to see that for this type of forger $(z_1, \sigma_1) \neq (z_1^*, \sigma_1^*)$ which is exactly what B outputs as his collision for H under the public key ek_0. Given the way the verification algorithm is defined, it is easy to see that B outputs a collision iff A successfully forges a signature for m^*.

– **Type II Forger.** In this type of forgery we have that $(m, \sigma_0) \neq (m^*, \sigma_0^*)$ and $h(ek_1, m, \sigma_0) = h(ek_1, m^*, \sigma_0^*)$. In this case B finds a collision for H under the public key ek_1. B is given ek_1.

He generates $(ek_0, td_0) \xleftarrow{\$} Gen(1^k)$ on his own, computes $z_0 = h(ek_0, m_f, r)$ for a random $r \in \mathcal{R}_{ek_0}$ and sends $vk = (ek_0, ek_1, z_0)$ to A. A makes a signature query for a message m. B computes $z_1 = h(ek_1, m, r')$, lets $\sigma_0 = r'$ and $\sigma_1 = h^{-1}(td_0, m_f, r, z_1)$ and returns the signature (σ_0, σ_1) to A. Eventually, A will output a forgery $(m^*, \sigma_0^*, \sigma_1^*)$. B outputs the pair $(m, \sigma_0) \neq (m^*, \sigma_0^*)$ as his collision for H under the public key ek_1. It is easy to see that based on the way this type of forgery is defined B can successfully find a collision as long as he simulates A's view (which we showed he can).

Efficiency. We will shortly look at specific instantiations of the above construction based on standard assumptions, but it is useful to evaluate the efficiency of the general construction as well. The signature consists of two elements in $\mathcal{R}_{ek}$. In most instantiations of chameleon hash functions this translates to two elements from the underlying group. The cost of signing a message is two invocations of h^{-1}. As mentioned earlier, in a number of constructions for chameleon has functions, this translates to simple arithmetic operations and does not include exponentiations. In these cases, signing a message becomes very fast. The verification requires two invocations of the function h. The verification key for the scheme includes two evaluation keys for the chameleon hash function and one element from the range of the chameleon hash. Similar to the construction of previous section, the last element can be shortened by applying a target collision resistant hash function.

Strong OTS from Σ-protocols. Bellare and Ristov [3] design chameleon hash functions based on Σ-protocols that satisfy certain security properties (in the standard model). Combined with our result, this leads to a general transformation of Σ-protocols to strongly unforgeable one-time signature schemes in the standard model:

Theorem 33. *There exists an efficient transformation for building* SUF-CMA *one-time signature schemes from any Σ-protocol that satisfies* strong special soundness *and* strong HVZK*.*

We refer the reader to [3] for formal definitions of the above security notions where the authors show that these two requirements are already satisfied by several existing constructions such as the Schnorr [38] and GQ [24] protocols. It is also shown that for Σ-protocols such as Fiat-Shamir [19], Micali-Shamir [31], and Okamoto's [32], one can modify the original constructions to satisfy these properties without affecting the underlying hardness assumption or the efficiency of the original scheme. The one-time signature schemes one derives from several of these protocols are new, and warrant further study. In this paper, however, we focus on constructions that are directly based on the existing chameleon hash functions.

We point out that Bellare and Shoup [5] also show a general transformation for designing strong one-time signature schemes from Σ-protocols, but the security properties they require from the starting protocol appear to be stronger than ours (security against *concurrent attacks*). Particularly, while one can efficiently instantiate our construction based on the RSA problem or even the hardness of factoring integers, the same is not known about their constructions.

Unifying the previous works. Starting with the work of Boneh *et al.* [10], several works in the literature studied constructions for transforming any UF-AMA signature scheme to a *strongly* UF-CMA signature scheme. While Boneh *et al.*'s construction only works for signature schemes with a special partitioning property (a property not possessed by most signature schemes), the later works of [26,40,5] develop techniques that work for any UF-AMA secure signature scheme. The works of [5] and [26] use a strongly unforgeable one-time signature in their construction while that of [40] takes advantage of a chameleon hash function. Without going into the details of their constructions, we observe that based on our results the latter construction based on a chameleon hash function can be interpreted as an efficient instantiation of the former.

4 Instantiations Based on Standard Assumptions

Our general construction leads to a wide range of new constructions for strongly unforgeable one-time signature schemes. Several of these constructions match or improve the efficiency of the previous constructions based on similar assumptions or relax the underlying hardness assumption. Here, we study the properties of three specific instantiations of our construction based on assumptions such as the discrete-log problem, the hardness of factoring, and lattice-based assumptions.

4.1 A Construction Based on the Factoring Assumption

Next we describe an instantiation of our general construction based on the hardness of factoring integers. There are multiple constructions of chameleon hash functions based on factoring but we focus on the hash function of Shamir and Tauman [39], since it leads to a signature scheme with very fast signing (though the verification still requires exponentiation).

- **Key Generation:**
  1. Generate two random safe primes $p, q \in \{0,1\}^{k/2}$ and compute $n = pq$.
  2. Choose at random an element $g \in \mathbb{Z}_n^*$ of order $\lambda(n)$ where $\lambda(n) = \phi(n)/2 = (p-1)(q-1)/2$.
  3. The public key is $ek = (n, g)$ and the trapdoor key is $td = (p, q)$. The evaluation and inversion for the hash function $h(ek, \cdot) : \mathbb{Z}_n \times \mathbb{Z}_{\lambda_n} \to \mathbb{Z}_n^*$ is defined as:
- **Evaluation:** On message $m \in \mathbb{Z}_n$ and randomness $r \in \mathbb{Z}_{\lambda(n)}$, compute and return $g^{m||r} \mod n \in Z_n^*$. The concatenation treats m and r as bitstrings

where we assume that each $r \in \mathbb{Z}_{\lambda(n)}$ is represented using exactly k bits, even if some of the leading bits are zero.

- **Inversion:** On input messages $m, m' \in \mathbb{Z}_n$ and randomness $r \in \mathbb{Z}_{\lambda(n)}$, return $r' = 2^k(m - m') + r \mod \lambda(n)$.

See [39] for a proof that the above construction is a chameleon hash function based on the factoring assumption. An important property of the above construction is that the inversion function only requires one modular addition and a multiplication. Given the above hash function, the strongly unforgeable one-time signature scheme is as follows:

Construction 41. Let T_0, T_1 be target collision resistant functions, where T_0 maps elements of $\mathbb{Z}^*_{n_0}$ to bitstrings, and T_1 maps elements of $\mathbb{Z}^*_{n_1}$ to a subset of $\mathbb{Z}_{n_0}$.

- **Key Generation:**
  1. Compute $n_0 = p_0 q_0$ and $n_1 = p_1 q_1$ where p_i, q_i are safe primes, for $i \in \{0, 1\}$.
  2. Choose at random elements $g_0 \in \mathbb{Z}^*_{n_0}$ and $g_1 \in \mathbb{Z}^*_{n_1}$ of orders $\lambda(n_0)$ and $\lambda(n_1)$ respectively.
  3. Compute $z_0 = T_0(g_0^{0||r_0} \mod n_0)$ and $z_1 = T_1(g_1^{0||r_1} \mod n_1)$.
  4. Return $vk = (n_0, n_1, g_0, g_1, z_0)$ and $sk = ((p_0, q_0), (p_1, q_1), r_0, r_1, z_1)$.
- **Signing:** On a message $m \in \mathbb{Z}_n$, return the signature $\sigma = (\sigma_0, \sigma_1)$ where $\sigma_0 = 2^k(0 - z_1) + r_0 \mod \lambda(n_0)$ and $\sigma_1 = 2^k(0 - m) + r_1 \mod \lambda(n_1)$.
- **Verification:** On a message m and the signature $\sigma = (\sigma_0, \sigma_1)$
  1. Compute $y = T_1(g_1^{m||\sigma_1} \mod n_1)$.
  2. Compute $y' = T_0(g_0^{y||\sigma_0} \mod n_0)$.
  3. accept if $y' = z_0$ and reject otherwise.

Efficiency Comparison. The verification key contains two integers, two group elements, and one hash output. The signature consists of two group elements. The signing algorithm consists of two modular additions and two modular multiplications. The verification algorithm includes two exponentiations. Hence, the scheme is desirable in situations where the signer has low computational power, but the verifier does not have such limits.

General constructions of OTS schemes lead to secure schemes based on the factoring assumption, however the resulting constructions are impractical. Goldwasser *et al.* [22] also design a signature scheme from claw-free trapdoor permutations with instantiations based on the factoring assumption. Their signature scheme leads to short signatures, but the signing algorithm requires one exponentiation per bit of the message, which makes the resulting scheme inefficient. A number of works in the literature have studied the design of fail-stop signature schemes based on factoring-related assumptions [7,33,36,41,37]. However, almost all such constructions rely on assumptions that are stronger than the standard factoring assumption. The only exception is the construction of [7,33] which is

based on the intractability of factoring integers $n = pq$ for p, q with $p = q = 3 \mod 4$ (i.e. Blum integers) and $p \neq q \mod 8$. However, it is not known if factoring integers of this special form is as hard as factoring arbitrary RSA integers (see [37] for a more detailed discussion).

Hence, to the best of our knowledge, our construction is the first short OTS based solely on the standard factoring assumption, where the signing algorithm only requires simple arithmetic operations.

More constructions based on factoring-related Assumptions. [3] and [2] design multiple constructions of chameleon hash functions based on factoring and the RSA problem, respectively. When plugged into our general construction, each of these leads to a new strongly unforgeable one-time signature scheme. However, the cost of signing for these schemes is higher than the scheme we described as they involve exponentiation.

4.2 A Construction from Lattice-Based Assumptions

We briefly describe the lattice-based chameleon hash function of [20,34] based on preimage samplable (trapdoor) function (PSF). Each function in the family is represented by matrices $A \in \mathbb{Z}_q^{k \times m_1}$ and $B \in \mathbb{Z}_q^{k \times m_2}$ where k is the security parameter, $q = poly(k)$ is an odd prime and $m_1, m_2 = O(k \log q)$. The message space is $\mathcal{M} = \{x \in \mathbb{Z}_q^{m_1} : ||x||_2 \leq \beta_1\}$, and the randomness domain is $\mathcal{R} = \{r \in \mathbb{Z}_q^{m_2} : 0 < ||r||_2 \leq \beta_2\}$ where β_1 and β_2 are chosen appropriately. The randomness distribution is a discrete Gaussian over $\mathbb{Z}_q^{m_2}$, and the range of the function is $\mathcal{Y} = \mathbb{Z}_q^k$.

A function in the family is defined by $h_{A,B}(m, r) = Am + Br$ where $m \in \mathcal{M}$ and $r \in \mathcal{R}$. The hardness of collision follows from the *short integer solution* (SIS) assumption, that is related to worst-case lattice-based assumptions such as approximating the shortest vector problem. The trapdoor is a short basis for the lattice whose parity-check matrix is B. The inversion function $h_{A,B}^{-1}$ involves simple linear algebra operations. We do not explain the details here but refer the reader to [20,34] for more information.

The strongly unforgeable one-time signature scheme is then obtained by plugging in the above chameleon hash function in our general construction. Hence we directly move to the analysis of the efficiency for the resulting scheme.

Efficiency and Comparison. The verification key for the resulting scheme consists of matrices A and B which leads to a key size of $O(k^2 \log k)$. The signature contains two elements in $\mathcal{R}$, and hence the signature is of size $O(k \log k)$. The signing algorithm requires two invocations of the inversion for the preimage samplable function described above.

Lyubashevsky and Micciancio [29] also design an asymptotically efficient one-time signature based on lattice-based assumptions, but they need to work with lattices with special structure (i.e. ideal lattices). As pointed out by the authors, it is desirable to avoid such extra assumptions while achieving a similar level of efficiency. Our construction removes this extra assumption, while still providing

a signature of size $O(k \log k)$. However, the verification key and the signing cost for our scheme are larger than theirs.

Recently, Cash *et al.* [15] designed digital signature schemes based on worst-case lattice-based assumptions in the standard model. However, the signature size in their constructions is in the order of $O(k^2 \log k)$. In [11] Boyen improved their result by building a signature scheme with signature size linear in k, while the verification key size is still significantly longer than ours (i.e. cubic in k).

Acknowledgement. I would like to thank Greg Zaverucha for helpful discussions and the anonymous reviewers for valuable comments.

References

1. Abe, M., Cui, Y., Imai, H., Kiltz, E.: Efficient hybrid encryption from ID-based encryption. Designs, Codes and Cryptography 54(3), 205–240 (2010)
2. Ateniese, G., de Medeiros, B.: Identity-based chameleon hash and applications. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 164–180. Springer, Heidelberg (2004)
3. Bellare, M., Ristov, T.: Hash functions from sigma protocols and improvements to VSH. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 125–142. Springer, Heidelberg (2008)
4. Bellare, M., Rogaway, P.: Collision-resistant hashing: Towards making UOWHFs practical. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 470–484. Springer, Heidelberg (1997)
5. Bellare, M., Shoup, S.: Two-tier signatures, strongly unforgeable signatures, and fiat-shamir without random oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 201–216. Springer, Heidelberg (2007)
6. Bleichenbacher, D., Maurer, U.M.: On the efficiency of one-time digital signatures. In: Kim, K., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 145–158. Springer, Heidelberg (1996)
7. Bleumer, G., Pfitzmann, B., Waidner, M.: A Remark on Signature Scheme Where Forgery Can Be Proved. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 441–445. Springer, Heidelberg (1991)
8. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. SIAM Journal on Computing 36(5), 915–942 (2006)
9. Boneh, D., Katz, J.: Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 87–103. Springer, Heidelberg (2005)
10. Boneh, D., Shen, E., Waters, B.: Strongly unforgeable signatures based on computational diffie-hellman. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 229–240. Springer, Heidelberg (2006)
11. Boyen, X.: Lattice Mixing and Vanishing Trapdoors: A Framework for Fully Secure Short Signatures and More. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 499–517. Springer, Heidelberg (2010)
12. Brakerski, Z., Kalai, Y.T.: A Framework for Efficient Signatures, Ring Signatures and Identity Based Encryption in the Standard Model, `http://eprint.iacr.org/2010/086.pdf`
13. Brassard, G., Chaum, D., Crépeau, C.C.: Minimum disclosure proofs of knowledge. Journal of Computer and System Sciences 37(2), 156–189 (1988)

14. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
15. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai Trees, or How to Delegate a Lattice Basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010)
16. Dahmen, E., Krauß, C.: Short hash-based signatures for wireless sensor networks. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 463–476. Springer, Heidelberg (2009)
17. Dodis, Y., Katz, J.: Chosen-ciphertext security of multiple encryption. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 188–209. Springer, Heidelberg (2005)
18. Even, S., Goldreich, O., Micali, S.: Online/offline signatures. Journal of Cryptology (1996)
19. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
20. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th Annual ACM Symposium on Theory of Computing, pp. 197–206. ACM Press, New York (May 2008)
21. Goldreich, O.: Two remarks concerning the goldwasser-micali-rivest signature scheme. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 104–110. Springer, Heidelberg (1987)
22. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 17(2), 281–308 (1988)
23. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006)
24. Guillou, L.C., Quisquater, J.-J.: A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 123–128. Springer, Heidelberg (1988)
25. Hohenberger, S., Waters, B.: Short and stateless signatures from the RSA assumption. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 654–670. Springer, Heidelberg (2009)
26. Huang, Q., Wong, D.S., Zhao, Y.: Generic transformation to strongly unforgeable signatures. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 1–17. Springer, Heidelberg (2007)
27. Krawczyk, H., Rabin, T.: Chameleon signatures. In: ISOC Network and Distributed System Security Symposium – NDSS 2000. The Internet Society, San Diego (February 2000)
28. Lamport, L.: Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory (October 1979)
29. Lyubashevsky, V., Micciancio, D.: Asymptotically efficient lattice-based digital signatures. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 37–54. Springer, Heidelberg (2008)
30. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)

31. Micali, S., Shamir, A.: An improvement of the fiat-shamir identification and signature scheme. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 244–247. Springer, Heidelberg (1990)
32. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)
33. Pedersen, T.P., Pfitzmann, B.: Fail-stop signatures. SIAM Journal on Computing 26, 291–330 (1997)
34. Peikert, C.: Bonsai trees (or, arboriculture in lattice-based cryptography). Cryptology ePrint Archive, Report 2009/359 (2009), http://eprint.iacr.org/
35. Perrig, A.: The BiBa one-time signature and broadcast authentication protocol. In: ACM CCS 2001: 8th Conference on Computer and Communications Security, pp. 28–37. ACM Press, New York (November 2001)
36. Safavi-Naini, R., Susilo, W.: Threshold fail-stop signature schemes based on discrete logarithm and factorization. In: Pieprzyk, J., Okamoto, E., Seberry, J. (eds.) ISW 2000. LNCS, vol. 1975, pp. 292–307. Springer, Heidelberg (2000)
37. Schmidt-Samoa, K.: Factorization-based fail-stop signatures revisited. In: López, J., Qing, S., Okamoto, E. (eds.) ICICS 2004. LNCS, vol. 3269, pp. 118–131. Springer, Heidelberg (2004)
38. Schnorr, C.-P.: Efficient signature generation by smart cards. Journal of Cryptology 4(3), 161–174 (1991)
39. Shamir, A., Tauman, Y.: Improved online/Offline signature schemes. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 355–367. Springer, Heidelberg (2001)
40. Steinfeld, R., Pieprzyk, J., Wang, H.: How to strengthen any weakly unforgeable signature into a strongly unforgeable signature. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 357–371. Springer, Heidelberg (2006)
41. Susilo, W., Safavi-Naini, R.: An efficient fail-stop signature scheme based on factorization. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 62–74. Springer, Heidelberg (2003)
42. van Heyst, E., Pedersen, T.P.: How to make efficient fail-stop signatures. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 366–377. Springer, Heidelberg (1993)
43. Zaverucha, G.M., Stinson, D.R.: Short one-time signatures. Cryptology ePrint Archive, Report 2010/446 2010, http://eprint.iacr.org/

A Security Definitions for Signature Schemes

A signature scheme is a tuple of the following algorithms:

- $Gen(1^k)$**:** The key generation algorithm outputs a key pair (vk, sk).
- $Sign(sk, m)$**:** The signing algorithm takes in a secret key sk, and a message m, and produces a signature σ.
- $Ver(vk, m, \sigma)$**:** The verification algorithm takes in a verification key vk, a message m, and a purported signature σ, and returns 1 if the signature is valid and 0 otherwise.

Unforgeability against Chosen Message Attacks

The standard security notion for signatures is existential unforgeability with respect to adaptive chosen-message attacks (uf-cma) as formalized by Goldwasser,

Micali and Rivest [22]. It is defined using the following game between a challenger and an adversary A over a message space $\mathcal{M}$:

- **Setup:** The challenger runs the algorithm $Gen(1^k)$ to obtain the verification key vk and the secret key sk, and gives vk to the adversary.
- **Queries:** Proceeding adaptively, the adversary may request a signature on any message $m \in \mathcal{M}$ and the challenger will respond with $\sigma = Sign(sk, m)$. Let Q be the set of messages queried by the adversary.
- **Output:** Eventually, the adversary will output a pair (m, σ) and is said to win the game if $m \notin Q$ and $Ver(vk, m, \sigma) = 1$.

We define $\mathbf{Adv}_{\text{uf-cma,A}}$ to be the probability that adversary A wins in the above game.

Definition 1. *(UF-CMA [22]) A signature scheme $(Gen, Sign, Ver)$ is existentially unforgeable with respect to adaptive chosen message attacks if for all probabilistic polynomial time adversaries A, $\mathbf{Adv}_{\mathsf{A}}$ is negligible in the security parameter.*

Unforgeability against Apriori Message Attacks

Several works (e.g. [25]) have considered a weaker definition called existential unforgeability with respect to apriori chosen-message attacks (uf-ama). It is defined using the following game between a challenger and an adversary A over message space $\mathcal{M}$:

- **Queries:** The adversary sends the challenger a list Q of messages $m_1, \ldots, m_n \in \mathcal{M}$.
- **Response:** The challenger runs the algorithm $Gen(1^k)$ to obtain the verification key vk and the secret key sk. Next, the challenger signs each queried message as $\sigma_i = Sign(sk, m_i)$ for $i = 1$ to n. The challenger then sends $vk, \sigma_1, \ldots, \sigma_n$ to the adversary.
- **Output:** Eventually, the adversary will output a pair (m, σ) and is said to win the game if $m \notin Q$ and $Ver(vk, m, \sigma) = 1$.

We define $\mathbf{Adv}_{\text{uf-ama,A}}$ to be the probability that adversary A wins in the above game.

Definition 2. *(UF-AMA) A signature scheme $(Gen, Sign, Ver)$ is existentially unforgeable with respect to apriori chosen message attacks if for all probabilistic polynomial time adversaries A, $\mathbf{Adv}_{\text{uf-ama,A}}$ is negligible in the security parameter.*

Strongly Unforgeable One-time Signatures

A signature is called *one-time* if the adversary A in the above games is only allowed to make a single message query before creating a forgery.

A signature scheme is called *strongly* unforgeable if in the above security games, the adversary A wins even if in the output stage he forges a signature for a message $m \in Q$, by creating a different signature for it. We denote the corresponding notions of security with SUF-CMA and SUF-AMA .

B A Construction Based on the Discrete-Log Assumption

We use a well-known chameleon hash function based on the DL problem for our construction. The signature scheme we obtain is new and is as efficient as the best existing constructions based on the DLP (see [5] and [42]). Recently, Zaverucha and Stinson [43] designed a new OTS scheme based on the discrete-log assumption that is shorter than all previous schemes including ours. Further improvements in efficiency of the underlying chameleon hash function would lead to even more efficient constructions. The DL-based chameleon hash function we use is described next:

- **Key Generation:**
  1. Fix a generator g_1 for a prime-order group G, of size p.
  2. Generate a random x in $\mathbb{Z}_p$, and compute $g_2 = g_1^x$.
  3. The public key $ek = (p, g_1, g_2)$ and the trapdoor key is $td = x$.
- **Evaluation:** On message m and randomness r in $\mathbb{Z}_p$, return $g_1^m g_2^r$.
- **Inversion:** On inputs messages m, m' and randomness r, compute $r' = x^{-1}(m - m') + r \mod p$.

Given the above hash functions, the strongly unforgeable one-time signature scheme is as follows:

Construction B1. A SUF-CMA one-time signature based on the DL problem

- **Key Generation:**
  1. Fix a generator g_1 for a prime order group G of size p. Let T be target collision hash function that maps elements of G to a subset of $\mathbb{Z}_p$.
  2. Generate random $x, x' \in \mathbb{Z}_p$ and compute $g_2 = g_1^x$ and $g_3 = g_1^{x'}$.
  3. Compute $z_0 = T(g_1 g_2^r)$ and $z_1 = T(g_1 g_3^{r'})$ for random $r, r' \in \mathbb{Z}_p$.
  4. The verification key is $vk = (g_1, g_2, g_3, z_0)$ and the signing key is $sk = (y = x^{-1}, y' = x'^{-1}, r, r', z_1)$[1].
- **Signing:** To sign a message m, compute and return signature $\sigma = (\sigma_0, \sigma_1)$ where $\sigma_0 = y'(1 - m) + r' \mod p$ and $\sigma_1 = y(1 - z_1) + r \mod p$.
- **Verification:** On message m and the signature $\sigma = (\sigma_0, \sigma_1)$, accept if $T(g_1^{T(g_1^m g_3^{\sigma_0})} g_2^{\sigma_1}) = z_0$ and reject otherwise.

Efficiency Comparison. The verification key contains three group elements and one TCR hash output. The signature consists of two integers in $\mathbb{Z}_p$. Signing is very fast and only includes simple arithmetic operations. Verification requires exponentiations.

[1] Note that in this instantiation we let the fixed message $m_f = 1$.

Compared to the DL-based construction of [5] which is based on Okamoto's identification scheme, our construction has a shorter verification key (one less group element). The signature sizes are the same and the signing algorithm for both schemes only requires arithmetic operations[2].

Our construction matches the efficiency of the DL-based construction of Van Heyst and Pedersen [42] in many respects such as the key and signature sizes and the cost of signing a message.

[2] We note that [5] designs a more efficient OTS scheme based on a stronger assumption called *one-more DLP*.