Post-quantum signature with preimage chameleon hashing

Thiago Astrizi[1] **, Ricardo Custódio**[1] **, Lucia Moura**[2]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC), Brazil
thiago.leucz.astrizi@posgrad.ufsc.br, ricardo.custodio@ufsc.br

[2]School of Electrical Engineering and Computer Science
University of Ottawa, Canada
lmoura@uottawa.ca

Abstract. *In this work, we propose a generalization of the concept of chameleon hash first described in [Krawczyk and Rabin 1998], which we call preimage chameleon hash. While in the conventional chameleon hash, the trapdoor allows a user to compute second preimages, in this generalization, it is possible to compute first preimages. We show how to adapt the post-quantum chameleon hash from [Cash et al. 2010] to a preimage chameleon hash and use this modified construction to build a new signature scheme based on [Mohassel 2011]. A preimage chameleon hash allows the signer to encode in its signature chosen information to be checked during verification. We prove our signature scheme to be strongly unforgeable under a chosen message attack (SUF-CMA).*

1. Introduction

A traditional cryptographic hash function H is an algorithm that, given a message m of any size, produces output d of fixed size, usually small compared to the size of m. Output d is known as a digest. Finding two different messages with the same digest d should be unlikely. In addition, it must be easy to obtain d from m, and difficult to obtain m, given d.

Unlike the traditional hash function, a chameleon hash is a cryptographic primitive associated with a pair of keys: the hash key and the trapdoor key. The hash key is required to determine the hash of a message, while using the trapdoor key, it is possible to find second-preimages. This type of hash was first proposed by [Krawczyk and Rabin 1998].

Relationships between chameleon hash and other cryptographic primitives were shown by [Bellare and Ristov 2008], where the authors show how to construct sigma protocols from chameleon hashes and also how some suitable sigma protocols could be transformed in chameleon hash schemes. Mohassel [Mohassel 2011] shows how to use chameleon hashes to build one-time signatures. In [Blazy et al. 2015] it was shown how to construct d-times signatures from chameleon hash functions in a way that the key size grows logarithmically as a function of d. Other applications of chameleon hashes are sanitizable signatures [Ateniese et al. 2005], redactable blockchains [Ateniese et al. 2017], privacy-preserving communication protocols [Guo et al. 2013] and dynamic public key certificates [Chien 2018].

The biggest challenge to design and use chameleon hash schemes is guaranteeing security in a scenario where collisions computed with the trapdoor are revealed. Some known constructions suffer from the key exposure problem, where an

adversary could extract the trapdoor if given access to sample collisions, as discussed in [Ateniese and de Medeiros 2005]. Even if an adversary cannot disclose the trapdoor, this does not guarantee that collisions do not leak information allowing an attacker to discover new collisions.

Different chameleon hash schemes can be found in the literature adding to the classic scheme new properties that improve collision resistance such as the secret-coin chameleon hash [Ateniese et al. 2017], labelled chameleon hash, and identity-based chameleon hash. The secret-coin chameleon hash is a scheme where the hash returns a pair (d, p) with the digest and a proof that the digest is correct and uses a VERIFY algorithm to check that the digest was correctly computed, without revealing random decisions taken while computing the hash. A labelled chameleon hash assumes that each hash uses a unique label that should be used only once. Moreover the identity-based chameleon hash requires interaction with a trusted third-party to derive disposable trapdoors to compute collisions in the hash function.

1.1. Our Contributions

We propose to use a variation of the chameleon hash scheme in a new digital signature scheme. While traditional chameleon hash schemes are designed to allow a second preimage computation, this variation allows a first preimage computation.

With this powerful variant, we build a new signature scheme called a digital preimage signature. The main motivation for this is the possibility of using a graphic element, within the message, that visually identifies and indicates the signatory and her consent to the signed message. In more detail, the digital signature is information produced using the secret key such that when concatenated with the message and used as input to the chameleon hash produces as output the graphic element. In addition, our construction aims to be secure even against adversaries that could run quantum algorithms. For this, we use a variation of the construction presented in [Cash et al. 2010].

Finally, we show a proof of security for our signature scheme in the random oracle model, showing that it can be safely used even against quantum attackers.

2. Preliminaries

2.1. Notation

We use small caps to represent algorithms like in "KEYGEN". Generic elements of sets are represented by multi-letter variables in italic, like in "msg". Natural numbers are represented as single-letter variables in italic. Matrices are represented by an uppercase italic A, with some numeric subscript if we need to define more than one. Vectors of integers are represented with boldface letters like in $\mathbf{b}$. Polynomials are represented by accented letters like in "$\hat{p}$" and vectors of polynomials are accented boldface letters like "$\hat{\mathbf{b}}$". All our polynomials have integer coefficients. The set of all vectors of integers modulo q with n dimensions is represented by $\mathbb{Z}^n$. The set of all vectors with n dimensions modulo q is $\mathbb{Z}_q^n$. Furthermore, the set of all $n \times m$ matrices of integers modulo q is $\mathbb{Z}_q^{n \times m}$.

When we talk about the norm $||\mathbf{b}||$ we assume the euclidean norm. For vector of polynomials $\hat{\mathbf{b}}$ we assume that its norm is $||\hat{\mathbf{b}}|| = \sqrt{\sum_{i=1}^{m} ||\mathbf{z}_i^2||}$ where each $\mathbf{z}_i$ is a vector of integers formed by the coefficients from each polynomial from $\hat{\mathbf{b}}$. Notwithstanding,

different definitions of norm can also achieve the desired results, with different trade offs. We represent as $\mathbb{Z}[x]/(f(x))$ a polynomial ring with integer coefficients and $\mathbb{Z}_q[x]/(f(x))$ is a polynomial ring with the coefficients being integers modulo q. Here $f(x)$ is a monic and irreducible polynomial.

The expression $A \cdot \mathbf{b}$ is a multiplication of a matrix and a vector. The expression $\mathbf{a} \cdot \mathbf{b}$ and $\hat{\mathbf{a}} \cdot \hat{\mathbf{b}}$ are the dot product of two vectors. In the second case each vector component is a polynomial and its sum and multiplication follows the usual multiplication rules for polynomials.

2.2. Signature Schemes

A **signature scheme** is a tuple of 3 efficient algorithms: $(\text{GEN}, \text{SIGN}, \text{VERIFY})$ defined over $(\mathcal{M}, \mathcal{S})$, where $\mathcal{M}$ is the set of all possible messages and $\mathcal{S}$ is the set of signatures:

- GEN is an efficient probabilistic algorithm that on input 1^n with n as the security parameter, outputs a cryptographic key pair (pk, sk);
- SIGN is an efficient probabilistic algorithm that given a secret key sk, and a message $msg \in \mathcal{M}$, produces a signature $sig \in \mathcal{S}$;
- VERIFY is an efficient determistic algorithm that given a public key pk, a message $msg \in \mathcal{M}$ and a signature $sig \in \mathcal{S}$, it returns `accept` if the signature is valid and `reject` otherwise.

We base the signature scheme security on the Strongly Unforgeable under Chosen Message Attacks model (SUF-CMA). This security model was formalized by [Goldwasser et al. 1988] as a game between the adversary and a challenger. The game is formed by the following parts:

- **Setup:** The challenger runs $(pk, sk) \leftarrow \text{GEN}(1^n)$ and send pk to the adversary;
- **Queries:** The adversary request sample signatures for the messages msg_i with $i \in \{1, \ldots, p\}$. For each message msg_i requested. The challenger responds sending $sig_i = \text{SIGN}(sk, msg_i)$, the challenger can choose the next requested message after examining all the previous responses;
- **Output:** The adversary outputs a pair (msg, sig) and wins the game if $\text{VERIFY}(\mathcal{PK}, msg, sig) = \texttt{accept}$ and $(msg, sig) \neq (msg_i, sig_i)$ for all $i \in \{1, \ldots, p\}$.

A function $f : \mathbb{Z} \to \mathbb{R}$ is *negligible* if and only if $\lim_{n\to\infty} f(n)n^c = 0$ for all integer positive values of c. A signature scheme is considered secure under the SUF-CMA model if, for all efficient adversary $\mathcal{A}$, the probability of winning the above game as a function of n is a negligible function.

2.3. Chameleon Hash Schemes

A **chameleon hash scheme** CH is a tuple of 3 efficient algorithms (KEYGEN, HASH, COLLISION) defined over a message space $\mathcal{M}$, a randomness space $\mathcal{R}$ and a digest space $\mathcal{D}$, with the following properties:

- KEYGEN is an efficient probabilistic algorithm that on input 1^n, outputs a key pair (hk, tk). We call hk the hash key and tk the trapdoor key;

- HASH is an efficient deterministic algorithm that on input hk, a message $msg \in \mathcal{M}$ and a parameter $rnd \in \mathcal{R}$, outputs a digest $dgt \in \mathcal{D}$;
- COLLISION is an efficient probabilistic algorithm that on input tk, msg_1, rnd_1 and $msg_2 \neq msg_1$, outputs a rnd_2 such that $\text{HASH}(hk, msg_1, rnd_1) = \text{HASH}(hk, msg_2, rnd_2)$.

Note that COLLISION computes an arbitrary second-preimage. In the literature, there is no consensus on how to name this algorithm. Some authors use the name "FORGE" [Ateniese and de Medeiros 2004] because it allows to create forgeries in chameleon signatures, others call it "COLL" [Blazy et al. 2015] as abbreviation of "collision" and some others use HASH^{-1} [Mohassel 2011] meaning that it is the inverse of the hash function $\text{HASH}(hk, msg, \cdot)$. Here the name "COLL" was preferred, but it is used in the non-abbreviated form.

The security of the chameleon hash scheme is based on the following properties (see [Derler et al. 2020]):

- *Uniformity:* For all pair of messages (msg_1, msg_2), given randomly chosen rnd, the random variables $\text{HASH}(hk, msg_1, rnd)$ and $\text{HASH}(hk, msg_2, rnd)$ have probability distributions computationally indistinguishable. For all constructions found in the literature, this property holds statistically: not even inefficient adversaries can differentiate between the output of $\text{HASH}(hk, msg_1, \cdot)$ and $\text{HASH}(hk, msg_2, \cdot)$ for a random rnd, except with probability negligibly next to $1/2$;
- *Weak Collision Resistance:* The property is defined in an attacking game between a challenger and an adversary $\mathcal{A}$ where for all adversaries, the probability of winning the game is negligible. The game has the following parts:
  - **Setup:** The challenger runs $(hk, tk) \leftarrow \text{KEYGEN}(1^n)$ and sends hk to the adversary;
  - **Output:** The adversary returns a distinct pair (msg', rnd') and (msg'', rnd'') and wins if $\text{HASH}(hk, msg', rnd') = \text{HASH}(hk, msg'', rnd'')$;
- *Standard Collision Resistance:* The property is defined by the following attack game which gives more power to the attacker:
  - **Setup:** The challenger runs $(hk, tk) \leftarrow \text{KEYGEN}(1^n)$ and sends hk to the adversary;
  - **Queries:** The adversary requests sample collisions sending a polynomial number of p queries for the challenger. Each query has the format $(msg_{1i}, rnd_{1i}, msg_{2i})$ for $i \in \{1, \ldots, p\}$. The challenger respond for each query with $rnd_{2i} = \text{COLLISION}(tk, msg_{1i}, rnd_{1i}, msg_{2i})$. The adversary can choose its queries adaptively: it can choose each one after examining the previous responses;
  - **Output:** The adversary outputs distinct tuples (msg', rnd') and (msg'', rnd'') and wins the game if $\text{HASH}(hk, msg_{1i}, rnd_{1i}) = \text{HASH}(hk, msg_{2i}, rnd_{2i})$ and $msg' \neq msg''$, $msg' \neq msg_{1i}$ and $msg' \neq msg_{2i}$ for all $i \in \{1, \ldots, p\}$.

All constructions found in the literature have proven weak collision resistance under reasonable assumptions. Constructions with standard collision resistance are presented in [Derler et al. 2020, Camenisch et al. 2017], but with a different definition of chameleon hash.

2.4. Post-Quantum Construction of Chameleon Hash

This construction presented in [Cash et al. 2010] derives its security from the Short Integer Solution Problem (SIS). We define the $SIS_{n,m,q,\beta}$ as the problem of, given $A \in \mathbb{Z}_q^{n \times m}$ randomly chosen, finding a nonzero vector $\mathbf{x} \in \mathbb{Z}_q^m$ such that $A \cdot \mathbf{x} = \mathbf{0} \mod q$ and $||\mathbf{x}|| \leq \beta$. It was shown by [Ajtai 1996] that solving this problem in the average case was as difficult as solving some mathematical problems involving lattices. The problem is believed to be hard in the worst case, even with the help of quantum algorithms. This sparkled interest in lattice-based cryptography.

In order to avoid trivial solutions for the problem and avoid scenarios where solutions do not exist, it is required that $\beta < q$, $\beta > \sqrt{n \log q}$ and $m \geq n \log q$.

From a $SIS_{n,m,q,\beta}$ problem, we can define a corresponding chameleon hash construction defined over $(\mathcal{M}, \mathcal{R}, \mathcal{D})$ where $\mathcal{M} = \{0,1\}^{m'}$, $\mathcal{R} = \{\mathbf{r} \in \mathbb{Z}^{m''} : ||\mathbf{r}|| \leq \frac{1}{2}\sqrt{\beta^2 - m'}\}$ and $D = \mathbb{Z}_q^n$. We have $m = m' + m''$.

The public key hk stores matrices $A_1 \in \mathbb{Z}_q^{n \times m'}$ and $A_2 \in \mathbb{Z}_q^{n \times m''}$. Both are computationally indistinguishable from a random matrix chosen uniformly. Using these keys we compute the hash function as:

$$\textsc{Hash}(hk, \mathbf{x}, \mathbf{r}) = A_1 \cdot \mathbf{x} + A_2 \cdot \mathbf{r} \pmod q$$

Let $f_A(\mathbf{x}) = A \cdot x \mod q$. Let $f^{-1}(\mathbf{y})$ be the set of all preimages of a given $\mathbf{y}$ for the function f. And let SAMPLEPRE be a probabilistic algorithm that given some trapdoor tk (used as secret key in our chameleon hash function) and a given $\mathbf{y}$, samples one element $\mathbf{x}$ from $f^{-1}(\mathbf{y})$ such that $||\mathbf{x}|| \leq \frac{1}{2}\sqrt{\beta^2 - m'}$ with overwhelming probability. The algorithm must not leak information about tk if many of its results are revealed. Methods to define suitable matrices and the SAMPLEPRE algorithm can be found in [Micciancio and Peikert 2012].

We can then define the COLLISION algorithm as:

$$\textsc{Collision}(tk, \mathbf{x}, \mathbf{r}, \mathbf{x'}) = \textsc{SamplePre}(tk, A_1 \cdot \mathbf{x} - A_1 \cdot \mathbf{x'} + A_2 \cdot \mathbf{r}) \pmod q$$

Theorem 1 *The chameleon hash construction presented here is collision-resistant, assuming that the SIS problem is hard for a given set of parameters (n, m, q, β) shared with the chameleon hash construction and that both matrices $A_1 \in \mathbb{Z}_q^{n \times m'}$ and $A_2 \in \mathbb{Z}_q^{n \times m''}$ are computationally indistinguishable from a random matrix chosen uniformly.*

Proof: We show how a collision finder for the presented chameleon hash can be used to create an algorithm to solve $SIS_{n,m,q,\beta}$ with non-negligible probability.

Given the matrix $A \in \mathbb{Z}_q^{n \times m}$ and the other SIS parameters, the algorithm could split the matrix A in A_1 and A_2 such that $A = [A_1|A_2]$. It can then pass the matrices A_1 and A_2 to the collision-finder, which would return $(\mathbf{x'}, \mathbf{r'}, \mathbf{x''}, \mathbf{r''})$ such that $A_1(\mathbf{x'}) +$

$A_2(\mathbf{r'}) = A_1(\mathbf{x''}) + A_2(\mathbf{r''})$ in modulo q. And this means that $A \cdot (\mathbf{x'}||\mathbf{r'}) \equiv A \cdot (\mathbf{x''}||\mathbf{r''}) \pmod q$. This means that $A \cdot (\mathbf{x'}||\mathbf{r'}) - A \cdot (\mathbf{x''}||\mathbf{r''}) \equiv 0 \pmod q$, and so $A(\mathbf{x'} - \mathbf{x''}||\mathbf{r'} - \mathbf{r''}) \equiv 0 \pmod q$. Thus, we found a solution whose norm needs to be proven to not exceed β. Indeed, notice that $||(\mathbf{x'} - \mathbf{x''})|| \leq \sqrt{m'}$, as both vectors are binary and have m' dimensions and in the worst case, one of them is full of 1s and the other full of 0s. We also know that both **r'** and **r''** have norm smaller than $1/2\sqrt{\beta^2 - m'}$ by our requirements. So, by triangle inequality, $||(\mathbf{r'} - \mathbf{r''})|| \leq \sqrt{\beta^2 - m'}$. The norm of our solution is indeed lesser than the required value: $\sqrt{||\mathbf{x'} - \mathbf{x''}||^2 + ||\mathbf{r'} - \mathbf{r''}||^2} \leq \sqrt{m' + \beta^2 - m'} = \beta$. ■

In conclusion, finding a collision is at least as hard as solving the $\text{SIS}_{n,m,q,\beta}$ problem.

3. Our Proposal

3.1. Preimage Chameleon Hash Schemes

This proposed scheme is a tuple of 3 efficient algorithms $(\text{KEYGEN}, \text{HASH}, \text{PREIMAGE})$ defined over message space $\mathcal{M}$ and randomness space $\mathcal{R}$. The algorithms KEYGEN and HASH work equal as in the simple chameleon hash scheme. But the algorithm PREIMAGE has the following property:

- **PREIMAGE** is an efficient probabilistic algorithm that on input sk, msg and dgt, outputs a rnd such that $\text{HASH}(hk, msg, rnd) = dgt$.

In the literature, there are at least four chameleon hash constructions which could be used as a preimage chameleon hash: Krawczyk and Rabin's construction based on trapdoor permutation functions [Krawczyk and Rabin 1998], Bellare and Ristov construction based on Fiat-Shamir sigma protocol [Bellare and Ristov 2014], Ateniese and Medeiros construction based on Nyberg-Rueppel signature scheme [Ateniese and de Medeiros 2005] and Cash, Hofheinz, Kiltz and Peikert construction with security based on lattice problems [Cash et al. 2010]. The first two of them suffer from the key exposition problem: given a collision in the HASH function, an adversary could easily extract the secret key used in the scheme. This limits the applicability of the first two constructions. The idea of using a scheme of chameleon hashes with preimage computation also appeared in [Lu et al. 2019], where it was called chameleon hash+.

Similarly to the original chameleon hash scheme, we require the uniformity property and weak collision resistance property. But to capture the notion that this scheme allows for first preimage computation instead of second preimage, the standard collision-resistance could be defined with the following attack game:

- **Setup:** The challenger runs $(hk, tk) \leftarrow KeyGen(1^n)$ and send hk to the adversary;
- **Queries:** The adversary request sample preimages sending q queries for the challenger. Each query has the format (msg_i, dgt_i) for $i \in \{1, \ldots, q\}$. The challenger respond for each query with rnd_i such that $rnd_i = \text{PREIMAGE}(tk, msg_i, dgt_i)$. The adversary can adaptively choose its queries;
- **Output:** The adversary outputs distinct pairs (msg', rnd') and (msg'', rnd''), winning the game if $\text{HASH}(hk, msg', rnd') = \text{HASH}(hk, msg'', rnd'')$ and $msg' \neq msg_i$ and $msg'' \neq msg_i$ for all $i \in \{1, \ldots, q\}$.

The difference in the output restriction for the adversary prevents trivial attacks where it could compute $dgt_0 = \text{HASH}(hk, msg_0, rnd_0)$ over random values msg_0 and rnd_0, obtain rnd_1 from the query' (msg', dgt_0) and return $(msg_0, rnd_0, msg', rnd_1)$ as answer.

Notice that from a preimage chameleon hash, we can easily define a traditional chameleon hash using the following definition for the $Collision$ function:

$$\text{COLLISION}(tk, msg, rnd, msg') = \text{PREIMAGE}(tk, msg', Hash(pk, msg, rnd))$$

The converse is not true: one cannot necessarily generalize a chameleon hash scheme to this preimage chameleon hash variant, as being able to compute collisions with a trapdoor does not imply the ability to compute preimages with that trapdoor.

However, the additional power of this scheme also means that creating secure constructions of this scheme is even more challenging than for the classical chameleon hash scheme. If a classical chameleon hash had standard collision-resistance and we had a PREIMAGE algorithm to compute preimages given its trapdoor, this would not necessarily give us a preimage chameleon hash with enhanced collision-resistance. The reason is that in the attack game defined here, we give to the adversary the power to query preimages, something more powerful than querying for second-preimages.

3.2. Signatures from Preimage Chameleon Hash Schemes

Given a preimage chameleon hash $CH = (\text{KEYGEN}, \text{HASH}, \text{PREIMAGE})$ over $(\mathcal{M}, \mathcal{R}, \mathcal{D})$, it is possible to define a signature scheme $\Sigma = (\text{GEN}, \text{SIGN}, \text{VERIFY})$ over $(\mathcal{M}, \mathcal{R})$ using the following construction based in the signature scheme proposed in [Mohassel 2011].

The GEN is constructed like in Algorithm 1. The keys (pk, sk) store the preimage chameleon hash keys and also a target digest $dgt \in \mathcal{D}$. The SIGN algorithm is Algorithm 2 where SIGN is computing a $sig \in \mathcal{R}$ such that for the message $msg \in \mathcal{M}$, $\text{HASH}(hk, msg, sig) = dgt$. Finally, the VERIFY algorithm is Algorithm 3 which computes the hash for the pair (msg, sig) and checks if the digest is really dgt.

Algorithm 1: $\text{GEN}(1^n, dgt)$

1 $(hk, tk) \xleftarrow{\$} \text{KEYGEN}(1^n)$;
2 $pk \leftarrow (hk, dgt)$;
3 $sk \leftarrow (tk, dgt)$;
4 **Return** (pk, sk);

Algorithm 2: $\text{SIGN}(sk, msg)$

1 $(tk, dgt) \leftarrow sk$;
2 $sig \leftarrow \text{PREIMAGE}(tk, msg, dgt)$;
3 **Return** sig;

The difference between this construction and what is proposed in [Mohassel 2011] is that as here we are using preimage chameleon hashes, we verify our signature comparing $\text{HASH}(hk, msg, sign)$ not with a randomly chosen value from $\mathcal{D}$, but with any value specifically chosen during key creation (dgt).

Algorithm 3: VERIFY(pk, msg, sig)

```
(hk, dgt) ← pk;
if HASH(hk, msg, sig) = dgt then
    Return accept;
else
    Return reject;
end
```

This opens the possibility to use this signature construction in the following new scenarios and applications:

1. Instead of using the VERIFY algorithm, in some contexts, the HASH algorithm could be used to check the validity of a signature. The verifier could extract using HASH$(hk, msg, sign)$ some fixed information about the signer, which, if valid and correct, attests the validity of the signature.
2. User-friendly digital signatures could be created to mimic the appearance of a handwritten signature. A valid signature could produce with HASH$(hk, msg, sign)$ a compressed image of the signer's handwritten signature to be shown on the screen by some software.

As the target verification value is not random, to define security, in the underlying attack game, we require that the adversary chooses its value:

- **Setup:** Let Λ be any system parameters needed during key creation. The challenger send Λ to the adversary, the adversary replies with a target verification value dgt. The challenger runs $(pk, sk) \leftarrow$ GEN$(1^n, dgt)$ and sends pk to the challenger.
- **Queries:** The adversary requests sample signatures for the messages msg_i with $i \in \{1, \ldots, p\}$. For each message msg_i requested, the challenger respond sending sig_i such that $sig_i =$ SIGN(sk, msg_i). It chooses the queries adaptively.
- **Output:** The adversary outputs a pair (msg, sig) and wins the game if VERIFY$(pk, msg, sig) =$ `accept` and $(msg, sig) \neq (msg_i, sig_i)$ for all $i \in \{1, \ldots, p\}$.

Notice that instantiating this scheme with a preimage chameleon hash with standard collision resistance ensures its security in the sense of preventing existential forgeries against chosen message attacks: an existential forgery yields collisions involving new messages for the preimage chameleon hash.

4. The Post-Quantum Construction of Preimage Chameleon Hash

We overcome the problem of not having a preimage chameleon hash with standard collision resistance using as starting point the construction presented in Section 2.4 and prove that this construction could provide a signature scheme that is SUF-CMA (strongly unforgeable under chosen message attack) after making the following changes:

- Given $f_{A_2}(\mathbf{r}) = A_2 \cdot \mathbf{r}$, we define our hash algorithm as:

$$\text{HASH}(hk, \mathbf{x}, \mathbf{r}) = \text{H}(\mathbf{x}) + f_{A_2}(\mathbf{r}) \mod q$$

where H now is a generic hash function different than f_{A_1}.

- We define the PREIMAGE algorithm as:

$$\text{PREIMAGE}(tk, \mathbf{x}, \mathbf{d}) = \text{SAMPLEPRE}(tk, \mathbf{d} - H(\mathbf{x})) \mod q$$

  where SAMPLEPRE samples one of the preimages of f_{A_2} given the trapdoor tk, returning a preimage $\mathbf{y}$ with small $||\mathbf{y}||$ with overwhelming probability.
- The PREIMAGE algorithm now has a state. It stores all previously computed return values, and if given the same input more than once, after computing the result the first time, it returns the already computed result. Like in the GPV signature proposed in [Gentry et al. 2008], this could be simulated deriving all probabilistic decisions in SAMPLEPRE from a pseudorandom generator (PRG) always using the same value stored in tk as seed.
  This restriction is necessary for our security proof and to avoid that an adversary could find collisions in f_{A_2} sending the same signing query in the attack game and expecting different signatures.

  We use as system parameters $\Lambda = (n, m, q)$ such that $A_2 \in \mathbb{Z}_q^{n \times m}$.

The public key is the matrix A_2 and the access to the hash function H. The secret key is the trapdoor tk necessary to compute SAMPLEPRE to sample preimages from f_{A_2}.

Theorem 2 *Our construction of the preimage chameleon hash is correct.*

Proof. If $\mathbf{r} = \text{PREIMAGE}(tk, \mathbf{x}, \mathbf{d})$, then:

$$\begin{aligned} \text{HASH}(hk, \mathbf{x}, \mathbf{r}) &= \mathrm{H}(\mathbf{x}) + f_{A_2}(\mathbf{r}) \mod q \\ &= \mathrm{H}(\mathbf{x}) + f_{A_2}(f_{A_2}^{-1}(\mathbf{d} - \mathrm{H}(\mathbf{x}))) \mod q \\ &= \mathrm{H}(\mathbf{x}) + \mathbf{d} - \mathrm{H}(\mathbf{x}) \mod q = \mathbf{d} \end{aligned}$$

■

This construction also is collision-resistant, as F is collision-resistant and summing the result of this function to a value returned by a random oracle keeps the collision resistance.

Theorem 3 *Let f_{A_2} be a collision-resistant many-to-one function such that there exists an algorithm* SAMPLEDOM *which samples $\boldsymbol{x} \in \mathbb{Z}^m$ from a distribution possibly non-uniform such that $f_{A_2}(\boldsymbol{x})$ is uniform. Then, the signature constructed using the preimage chameleon hash defined here is strongly unforgeable under the chosen message attack (SUF-CMA) in the random-oracle model.*

Proof. The proof is a generalization of the security proof for the GPV signature from [Gentry et al. 2008]. We show that given an efficient adversary that could forge signatures, we could use it to build an efficient algorithm to find collisions in f_{A_2} such that if the adversary succeeds with non-negligible probability, then this also happens with our algorithm.

The algorithm simulates the challenger in the following way.

- **Setup:** Our collision-finder algorithm receives the matrix A_2 and the parameters $\Lambda = (n, m, q)$. It sends Λ to the adversary $\mathcal{A}$, which replies with a target verification $\mathbf{d}$. The algorithm proceeds sending to $\mathcal{A}$ the public key $pk = (hk, \mathbf{d})$ with $hk = A_2$.

Next, as we are in the random oracle model, the adversary can make two kinds of queries: queries to the oracle H, sending $\mathbf{x}_i$ and receiving $\mathrm{H}(\mathbf{x}_i)$ and signing queries, sending $\mathbf{x}_i$ and receiving r_i such that $r_i = \textsc{Sign}(sk, \mathbf{x}_i)$. Without loss of generality, we require that the adversary $\mathcal{A}$ always makes an oracle query for each message $\mathbf{x}_i$ before sending this $\mathbf{x}_i$ in a signing query or before outputting $\mathbf{x}_i$ as a forgery in the end.

The queries are handled by our algorithm in the following way:

- **Oracle Query:** For each query $\mathbf{x}_i$, if this value was never queried before, choose $\mathbf{r}_i \xleftarrow{\$} \textsc{SampleDom}(pk)$. Let $\mathbf{h}_i \leftarrow \mathbf{d} - f_{A_2}(\mathbf{r}_i)$. The algorithm stores $\mathbf{r}_i$ in a dictionary using $\mathbf{x}_i$ as the key. Finally, it sends $\mathbf{h}_i$ to $\mathcal{A}$ as result.
  If a query $\mathbf{x}_i$ was already made in the past, get the value $\mathbf{r}_i$ associated with key $\mathbf{x}_i$ from the dictionary. Set $\mathbf{h}_i \leftarrow \mathbf{d} - f_{A_2}(\mathbf{r}_i)$ and send $\mathbf{h}_i$ as response to $\mathcal{A}$.
- **Signing Query:** For each query $\mathbf{x}_i$, look up for $\mathbf{r}_i$ in the dictionary using $\mathbf{x}_i$ as the key. Send $\mathbf{r}_i$ as the answer to $\mathcal{A}$.

This part of the algorithm simulates a random oracle choosing a random signature $\mathbf{r}_i$ to $\mathbf{x}_i$ and then deriving the random digest $\mathrm{H}(\mathbf{x}_i) = \mathbf{d}_i$ from the signature.

- **Output:** After all the queries, the adversary $\mathcal{A}$ returns a possible forgery as $(\mathbf{x}, \mathbf{r})$. Our algorithm then searches in the dictionary a value $\mathbf{r}^*$ stored with the key $\mathbf{x}$. It returns $(\mathbf{r}, \mathbf{r}^*)$ as a collision for f_{A_2}.

If the adversary $\mathcal{A}$ returned a forgery, then we have that $\mathrm{H}(\mathbf{x}) + f_{A_2}(\mathbf{r}) = \mathrm{H}(\mathbf{x}) + f_{A_2}(\mathbf{r}^*) \pmod q$. This algorithm fails if $\mathbf{r} = \mathbf{r}^*$ and succeeds otherwise. As f_{A_2} is a function many-to-one, in the worst case, there are only two possible values that f_{A_2} maps to the same result. Thus, the collision finder succeeds with at least half the probability that $\mathcal{A}$ wins its attack game. If this probability is non-negligible, then so is our probability of finding a collision. ■

4.1. Ring-based construction

The relevant property to ensure the security of our preimage chameleon hash, exactly like in the GPV signature, is the existence of the function f_A with the properties described in the last section. However, the description using matrices described there and used in the classical chameleon hash construction from Section 2.4 is not the only option.

We can define a function $f_{\hat{\mathbf{a}}}(\hat{\mathbf{r}}) = \hat{\mathbf{a}} \cdot \hat{\mathbf{r}}$ with operations defined over vectors of m polynomials from the ring $\mathbb{Z}_q[x]/(f(x))$ where $f(x)$ has degree n. This operation is collision-resistant, but instead of reducing the security to the SIS problem described in subsection 2.4, we reduce it to the ring-based SIS problem, the $R\text{-}SIS$.

The $R\text{-}SIS_{m,q,\beta}$ is defined as the problem of finding $\hat{\mathbf{x}} \in R^m$ such that $\hat{\mathbf{a}} \cdot \hat{\mathbf{x}} = 0$. As in the SIS problem, we require that $||\hat{\mathbf{x}}|| < \beta$ for sufficiently small values of β. The average case of this problem was also reduced to the worst case of some problems involving lattices, believed to be hard even for quantum algorithms by [Micciancio 2002], [Peikert and Rosen 2006] and [Lyubashevsky and Micciancio 2006], provided that $R_q = \mathbb{Z}_q[x]/(f(x))$ and $f(x)$ is an irreducible polynomial.

If one can compute distinct $\hat{\mathbf{r}}$ and $\hat{\mathbf{s}}$ such that $f_{\hat{\mathbf{a}}}(\hat{\mathbf{r}}) = f_{\hat{\mathbf{a}}}(\hat{\mathbf{s}})$, then we have $(\hat{\mathbf{r}} - \hat{\mathbf{s}})$ as a solution to the $R\text{-}SIS$ problem. If we restrict the domain of $f_{\hat{\mathbf{a}}}$ to values with norm

smaller than $\frac{1}{2}\beta$, then by triangle inequality, a collision will reveals a $R\text{-}SIS$ solution with norm smaller than β.

Methods of generating $\hat{\mathbf{a}}$ computationally indistinguishable from a vector chosen randomly and uniformly with a trapdoor tk such that we have the algorithms SAMPLEPRE and SAMPLEDOM were presented in [Micciancio and Peikert 2012].

The main advantage of the ring-based construction is that instead of requiring $m \geq n \log q$ to ensure the security, we only need to require that $m \geq \log q$. In the matrix version, A_2 would need to have approximately $n^2 \log q$ elements, but in the ring-based version, $\hat{\mathbf{a}}$ would need to store approximately $n \log q$ elements. As shown in [El Bansarkhani and Buchmann 2014] for the GPV signature, it is also possible to achieve faster implementations with a ring-based version.

5. Implementation, Results and Discussion

To evaluate the performance of our proposed construction, we implemented the preimage chameleon hash (PCH) using the library by [Rohloff et al. 2020] implementing the ring-based construction from [El Bansarkhani and Buchmann 2014] using the same method to derive the key $\hat{\mathbf{a}}$ from the trapdoor. The polynomial ring was R_q with q having $k = 27$ bits. The same parameters were used both for the GPV signature and our construction.

To compare the execution time, we measured in the same machine the following signatures: RSA and ECDSA from OpenSSL 1.1.1 library, CRYSTALS-Dilithium (proposed in [Ducas et al. 2018]) and FALCON (implementation proposed in [Pornin 2019]) from code submitted the NIST standardization project and a BLISS-B (described in [Ducas 2014]) implementation from strongSwan library version 5.8.4 [strongSwan 2020]. The ECDSA used the curve B-233.

All tests were run in a notebook LG S43 with a Dual-Core Intel Pentium B980 2.40GHz (without AVX2 support) with 4GB of memory and running Ubuntu 18.04.4. While measuring running time, the tests were performed 1000 times, and the mean was extracted. Given the measured standard deviation, we computed the error margin given an interval of confidence of 95%, assuming a normal distribution. For all schemes, the signature time also includes the time to perform a hash on the messages to be signed. All of them use SHA256, except for CRYSTALS-Dilithium and FALCON, which used SHAKE256. The code used to measure running times and sizes can be checked in [Astrizi 2020].

Table 1. Running time comparison [ms]. The confidence interval is 95%.

Scheme	Security level	KEYGEN	SIGN/PREIMAGE	VERIFY/HASH
RSA 2048	112	160.541 ± 6.540	2.620 ± 0.001	0.053 ± 0.000
ECDSA 233	112	0.661 ± 0.009	0.689 ± 0.001	1.338 ± 0.002
Dilithium 1280×1024	128	0.475 ± 0.023	1.719 ± 0.076	0.453 ± 0.000
BLISS-B I	128	916.577 ±14.900	2.448 ± 0.221	0.224 ± 0.020
FALCON-512	≈100	15.088 ± 0.542	0.629 ± 0.018	0.079 ± 0.000
GPV (n=512, k=27)	≈100	5.521 ± 0.026	32.005 ± 0.033	0.241 ± 0.008
PCH (n=512, k=27)	≈100	5.520 ± 0.021	32.232 ± 0.070	0.244 ± 0.008

Table 2. Size comparison [bytes].

Scheme	Digest	σ	pk	sk
RSA 2048	-	256	259	512
ECDSA 233	-	58	31	29
Dilithium 1280×1,024	-	2,829	1,472	3,504
BLISS-B I	-	732	933	1182
FALCON-512	-	651	897	1281
GPV ($n = 512, k = 27$)	-	61,440	61,440	114,688
PCH ($n = 512, k = 27$)	2048	61,440	61,440	114,688

The results of the running times are summarized in Table 1. The sizes for signatures, keys, and chameleon hash digests are summarized in Table 2. As expected, our construction has comparable performance and the same key size as a GPV signature.

The tests and implementation prove the viability of the construction. Despite slower and with bigger keys than classical and more modern post-quantum signature schemes, our presented construction is, at the present moment, the only one known to implement post-quantum digital signature securely based on preimage chameleon hash. Finding new preimage chameleon hash schemes with performance on par with modern signature schemes is an open research problem.

6. Conclusion

In this paper, we propose a new post-quantum digital signature scheme, where the signature is a value which (together with the message), yields via hash a value predefined by the signatory. To do this, we use a new feature added to the traditional chameleon hash function that allows first preimage computation using a trapdoor.

The new signature scheme, which we call the preimage signature, is proven here to be strongly unforgeable under a chosen message attack (SUF-CMA). The proposed scheme was coded and compared to traditional digital signature algorithms. The comparison shows that the new signature algorithm is viable; that is, it can be used in practice to sign electronic documents.

As the signature is the preimage of any value chosen by the signer during key creation, this opens new possibilities, like creating signatures where a signed message is verified comparing its hash after concatenated with the signature to check if the result is a given value with some special interest, for example, a representation of a handwritten signature. Such an approach can improve the user experience in terms of trust in the signature, not only regarding cryptographic verification, but also visual verification by recipients.

7. Acknowledgments

Thiago Astrizi was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior — Brasil (CAPES) — Finance Code 001. Lucia Moura was supported by the program PV-CNPq (Brazil) and by an NSERC (Canada) discovery grant.

References

Ajtai, M. (1996). Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 99–108.

Astrizi, T. L. (2020). Repository with preimage chameleon hash test source code. `https://github.com/thiagoharry/test_preimage_chameleon_hash`.

Ateniese, G., Chou, D. H., de Medeiros, B., and Tsudik, G. (2005). Sanitizable signatures. In *Computer Security – ESORICS 2005, LNCS 3679*, pages 159–177, Berlin, Heidelberg. Springer.

Ateniese, G. and de Medeiros, B. (2004). Identity-based chameleon hash and applications. In *Financial Cryptography, LNCS 3110*, pages 164–180, Berlin, Heidelberg. Springer.

Ateniese, G. and de Medeiros, B. (2005). On the key exposure problem in chameleon hashes. In *Security in Communication Networks, LNCS 3352*, pages 165–179, Berlin, Heidelberg. Springer.

Ateniese, G., Magri, B., Venturi, D., and Andrade, E. (2017). Redactable blockchain–or–rewriting history in bitcoin and friends. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 111–126. IEEE.

Bellare, M. and Ristov, T. (2008). Hash functions from sigma protocols and improvements to vsh. In *Advances in Cryptology - ASIACRYPT 2008, LNCS 5350*, pages 125–142, Berlin, Heidelberg. Springer.

Bellare, M. and Ristov, T. (2014). A characterization of chameleon hash functions and new, efficient designs. *Journal of cryptology*, 27(4):799–823.

Blazy, O., Kakvi, S. A., Kiltz, E., and Pan, J. (2015). Tightly-secure signatures from chameleon hash functions. In *Public-Key Cryptography – PKC 2015, LNCS 9020*, pages 256–279, Berlin, Heidelberg. Springer.

Camenisch, J., Derler, D., Krenn, S., Pöhls, H. C., Samelin, K., and Slamanig, D. (2017). Chameleon-hashes with ephemeral trapdoors. In *Public-Key Cryptography – PKC 2017, LNCS 10175*, pages 152–182, Berlin, Heidelberg. Springer.

Cash, D., Hofheinz, D., Kiltz, E., and Peikert, C. (2010). Bonsai trees, or how to delegate a lattice basis. In *Advances in Cryptology – EUROCRYPT 2010, LNCS 6110*, pages 523–552, Berlin, Heidelberg. Springer.

Chien, H.-Y. (2018). Dynamic public key certificates for IoT and WSN scenarios. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 646–651. IEEE.

Derler, D., Samelin, K., and Slamanig, D. (2020). Bringing order to chaos: The case of collision-resistant chameleon-hashes. In *Public-Key Cryptography – PKC 2020, LNCS 12110*, pages 462–492, Cham. Springer.

Ducas, L. (2014). Accelerating bliss: the geometry of ternary polynomials. *IACR Cryptol. ePrint Arch.*, 2014:874.

Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., and Stehlé, D. (2018). Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238–268.

El Bansarkhani, R. and Buchmann, J. (2014). Improvement and efficient implementation of a lattice-based signature scheme. In *Selected Areas in Cryptography – SAC 2013, LNCS 8282*, pages 48–67, Berlin, Heidelberg. Springer.

Gentry, C., Peikert, C., and Vaikuntanathan, V. (2008). Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206.

Goldwasser, S., Micali, S., and Rivest, R. L. (1988). A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on computing*, 17(2):281–308.

Guo, S., Zeng, D., and Xiang, Y. (2013). Chameleon hashing for secure and privacy-preserving vehicular communications. *IEEE Transactions on Parallel and Distributed Systems*, 25(11):2794–2803.

Krawczyk, H. and Rabin, T. (1998). Chameleon hashing and signatures. `http://eprint.iacr.org/1998/010`.

Lu, X., Au, M. H., and Zhang, Z. (2019). Raptor: A practical lattice-based (linkable) ring signature. In *Applied Cryptography and Network Security, LNCS 11464*, pages 110–130, Cham. Springer.

Lyubashevsky, V. and Micciancio, D. (2006). Generalized compact knapsacks are collision resistant. In *Automata, Languages and Programming, LNCS 4052*, pages 144–155, Berlin, Heidelberg. Springer.

Micciancio, D. (2002). Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 356–365. IEEE.

Micciancio, D. and Peikert, C. (2012). Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology – EUROCRYPT 2012, LNCS 7237*, pages 700–718, Berlin, Heidelberg. Springer.

Mohassel, P. (2011). One-time signatures and chameleon hash functions. In *Selected Areas in Cryptography, LNCS 6544*, pages 302–319, Berlin, Heidelberg. Springer.

Peikert, C. and Rosen, A. (2006). Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *Theory of Cryptography, LNCS 3876*, pages 145–166, Berlin, Heidelberg. Springer.

Pornin, T. (2019). New efficient, constant-time implementations of Falcon. *IACR Cryptol. ePrint Arch.*, 2019:893.

Rohloff, K., Cousins, D., and Polyakov, Y. (2020). PALISADE Lattice Cryptography Library (release 1.9.2). `https://palisade-crypto.org/`.

strongSwan (2020). Bimodal lattice signature scheme (BLISS). https://wiki.strongswan.org/projects/strongswan/wiki/BLISS. Accessed on 29/07/2020.