Short and Stateless Signatures from the RSA Assumption

Susan Hohenberger[*]
Johns Hopkins University

Brent Waters[†]
University of Texas at Austin

June 14, 2009

Abstract

We present the first signature scheme which is "short", stateless and secure under the RSA assumption in the standard model. Prior short, standard model signatures in the RSA setting required either a strong complexity assumption such as Strong RSA or (recently) that the signer maintain state. A signature in our scheme is comprised of one element in $\mathbb{Z}_N^*$ and one integer. The public key is also short, requiring only the modulus N, one element of $\mathbb{Z}_N^*$, one integer and one PRF seed.

To design our signature, we employ the known generic construction of fully-secure signatures from weakly-secure signatures and a chameleon hash. We then introduce a new proof technique for reasoning about weakly-secure signatures. This technique enables the simulator to predict a prefix of the message on which the adversary will forge and to use knowledge of this prefix to embed the challenge. This technique has wider applications beyond RSA.

We also use it to provide an entirely new analysis of the security of the Waters signatures: the only short, stateless signatures known to be secure under the Computational Diffie-Hellman assumption in the standard model.

1 Introduction

Signature schemes are a fundamental building block of modern cryptography. As such, it is imperative to develop and to provide to practitioners efficient schemes in which we have the highest confidence of security. The focus of this work is, therefore, on designing "short" signatures secure under the weakest possible complexity assumptions in the standard model.

Most of today's short signature schemes can be divided into three categories: schemes that use random oracles (e.g., [8, 24, 20, 2, 21, 4, 13, 12]), schemes that require strong complexity assumptions (e.g., Strong RSA [11, 6], q-Strong Diffie-Hellman [3] and LRSW [5]) and (recently) schemes where the signer must maintain state [16]. The one prior anomaly is the short and stateless signature scheme due to Waters [27], which is secure under the Computational Diffie-Hellman (CDH) assumption in bilinear groups in the standard model.

[*] Supported by National Science Foundation grant CNS-0716142 and a Microsoft New Faculty Fellowship.

[†] Supported by NSF CNS-0716199, Air Force Office of Scientific Research (AFOSR) under the MURI award for "Collaborative policies and assured information sharing" (Project PRESIDIO) and the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001. Portions of this work were done while this author was at SRI International.

Our Contribution. We provide the first short and stateless signature scheme secure under RSA in the standard model. While there are several "standard" variants of the RSA assumption, the one we employ is that given a modulus N (chosen as the product of two large safe primes), a random exponent e less than and relatively prime to $\phi(N)$, and a random $y \in \mathbb{Z}_N^*$, it is hard to compute x such that $y = x^e \mod N$. (The restriction to safe primes can be removed.)

In our scheme, a signature is comprised of one element of $\mathbb{Z}_N^*$ and one integer. This is roughly the same size as the Strong RSA signatures of Gennaro, Halevi and Rabin [11]. The Strong RSA signatures of Cramer and Shoup [6] are even larger, requiring two elements in $\mathbb{Z}_N^*$ and one integer (for the basic scheme) or one element in $\mathbb{Z}_N^*$, one prime and one integer (for the trapdoor hash scheme). (We note that Fischlin [10] and Hofheinz-Kiltz [15] provide more efficient versions of Cramer-Shoup signatures.) Our public keys are also short, requiring only the modulus N, one element of $\mathbb{Z}_N^*$, one integer and one PRF seed. In contrast, the Waters' public keys are asymptotically larger, requiring $O(\lambda)$ group elements, where λ is the security parameter.

To realize our new construction, we introduce an entirely new proof technique for digital signatures, which we'll describe in detail shortly. We view this new technique as a major contribution of the work. To demonstrate its usefulness beyond RSA, we show that it can be applied in the CDH setting to obtain a variant of the Waters signatures [27].

Both of these signatures are also *online/offline* signatures, where the majority of the signer's computation can be performed offline before she knows the message. In Section 5, we discuss further computational optimizations and tradeoffs for our RSA scheme.

Intuition behind the Construction and Proof Technique. Our proof strategy begins with the previously-known method for constructing fully-secure signatures from weakly-secure signatures and a chameleon hash. Since chameleon hash functions exist under the hardness of factoring [17] and the RSA assumption [1, 16], one only needs to design an appropriate weakly-secure scheme under RSA. Although, even this has proven an elusive task.

To design a weakly-secure scheme, we do as follows. Suppose the RSA challenge is (N, y, e^*) with the goal of computing $y^{1/e^*} \mod N$. Suppose the adversary provides us with the n-bit messages $M_1, \ldots, M_q$. Denote as w the shortest prefix of M^*, the message on which the adversary will later forge, that is different from all other prefixes of $M_1, \ldots, M_q$. Our strategy is to find w and then at this "point" embed the challenge exponent e^*. Of course, until the end of the game, the simulator does not know what M^* will be.

To find w, the simulator takes a guess as follows. If $q = 0$, meaning the adversary does not request any signatures, then the simulator only needs to guess the first bit of M^* and set w to this. If $q \geq 1$, the simulator may simply guess a pair (i^*, t^*), where $1 \leq i^* \leq q$ and $1 \leq t^* \leq n$. Interpret this pair as saying that M_{i^*} is a message with the longest prefix in common with M^* and the first location at which these two strings differ is t^*. (There may be more than one message in $M_1, \ldots, M_q$ containing the longest common prefix; guessing any one of them will suffice for our analysis.) If $q \geq 1$, then clearly, a valid pair (i^*, t^*) must exist and indeed, the simulator will have at least a $1/(qn)$ chance of guessing it.

Next we turn to embedding the challenge. We need to design a signature scheme that depends on all prefixes of its message. Let the public key contain the modulus N, a random $h \in \mathbb{Z}_N^*$ and a hash function H that maps arbitrary strings to prime numbers. Let $M^{(i)}$ denote the first i bits of M. For $i = 1$ to n, compute $e_i = H(M^{(i)})$. Then let the signature be

$$\sigma = h^{\prod_{i=1}^{n} e_i^{-1}} \mod N.$$

In the proof of security, the simulator selects H so that $H(w) = e^*$. In other words, the simulator designs the public key so that the challenge exponent e^* is used in the forged signature on M^*, but in none of the signatures for $M_1, \ldots, M_q$. Thus, by properly setting h to be y raised to the product of all primes corresponding to all prefixes of $M_1, \ldots, M_q$, the simulator can answer its q signing queries and yet extract from the forgery the RSA solution $y^{1/e^*} \bmod N$.

Brief Background on Short, Standard-Model Signatures. It is worthwhile to briefly compare our results to some existing short signatures in the standard model.

First, Dwork and Naor [**?**] and Cramer and Damgård [**?**] show how to make tree-based signatures shorter by using a "wide" tree (i.e., a larger branching factor) under the RSA assumption in the standard model. Roughly, there exists a trade-off where the tree depth can be decreased by a factor of $\lg w$ if the size of the public parameters is increased by a factor of w. However, the focus of this work is on finding even shorter signatures.

One approach has been to consider schemes under stronger complexity assumptions, such as Strong RSA [11, 6], q-Strong Diffie-Hellman [3] and LRSW [5]. All of these schemes rely on the hardness of problems which, for any given instance, there are an *exponential* number of valid solutions. This stands in sharp contrast to problems such as RSA and CDH, where for any given instance, there is only *one* solution. Moreover, the latter two schemes require that the number of elements in problem input grows with the number of signing queries made by the adversary. For instance, the q-Strong Diffie-Hellman assumption requires that given a generator g of prime order p and the tuple $(g^x, g^{x^2}, \ldots, g^{x^q})$, it is hard to compute $(c, g^{1/(x+c)})$ for any $c \in \mathbb{Z}_p^*$. Thus, if the adversary asks for q signatures, then the problem must remain hard when q powers of x are released. In RSA and CDH (and Strong RSA), the number of input elements is always a small constant, independent of the adversary's behavior. To obtain high confidence in the security of our schemes, we should based them on the simplest and weakest assumptions possible. In fairness, these excellent works have laid the foundation of our result and they are still unrivaled in their computational efficiency by our RSA scheme. Now that we have "short" RSA signatures, it would be of great interest to reduce the cost of signing and verification. See Section 5.

Another type of strong complexity assumption is to assume RSA is secure against *sub-exponential*-time attackers and then apply complexity leveraging techniques. Micali, Rabin and Vadhan [18] did this to give verifiable unpredictable functions (VUFs), which immediately give rise to a signature scheme. In contrast, we only assume the standard RSA problem is hard for polynomial-time attackers; in other words, all our reductions are polynomial in the security parameter.

Earlier this year, Hohenberger and Waters [16] presented short RSA and CDH based schemes, secure in the standard model, but where the signer had to maintain a counter value as state. This counter was incremented with each signature issued. Unfortunately, their scheme was compromised if the signer accidentally issued two signatures with the same counter value. Indeed, while early signatures, such as those of Goldwasser, Micali and Rivest [14], were stateful, the concept of the *stateless* signature has become so ingrained in practice that it is really more of a requirement than an extra feature. Moreover, stateful signatures are harder for systems designers to work with because, in addition to protecting the secret key, they must also safeguard a counter value (in writable memory) from being maliciously rolled back by an adversary.

2 Generic Transformation of Weakly-Secure Signatures to Fully-Secure Signatures using Chameleon Hashes

2.1 Signature Schemes

A signature scheme is a tuple of the following algorithms:

KeyGen(1^λ) : the key generation algorithm outputs a keypair (PK, SK).
Sign(SK, M) : the signing algorithm takes in a secret key SK, and a message M, and produces a signature σ.
Verify(PK, M, σ) : the verification algorithm takes in a public key PK, a message M, and a purported signature σ, and returns 1 if the signature is valid and 0 otherwise.

2.2 GMR Unforgeability

The basic security notion for signatures is *existential unforgeability with respect to adaptive chosen-message attacks* as formalized by Goldwasser, Micali and Rivest [14]. It is defined using the following game between a challenger and an adversary $\mathcal{A}$ over message space $\mathcal{M}$:

Setup: The challenger runs the algorithm **KeyGen**(1^λ) to obtain the public key PK and the secret key SK, and gives PK 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 $\sigma \leftarrow \mathbf{Sign}(\mathrm{SK}, M)$. 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 $\mathbf{Verify}(\mathrm{PK}, M, \sigma) = 1$.

We define $\mathbf{Adv}_{\mathcal{A}}$ to be the probability that adversary $\mathcal{A}$ wins in the above game.

Definition 2.1 (Unforgeability against Adaptive Chosen Message Attacks [14]) *A signature scheme* (**KeyGen**, **Sign**, **Verify**) *is existentially unforgeable with respect to adaptive chosen message attacks if for all probabilistic polynomial time adversaries $\mathcal{A}$,* $\mathbf{Adv}_{\mathcal{A}}$ *is negligible in λ.*

2.3 Weak Unforgeability

Several works (e.g., [3]) have considered a weaker definition called *existential unforgeability with respect to weak chosen-message attacks.* It is defined using the following game between a challenger and an adversary $\mathcal{A}$ 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 **KeyGen**(1^λ) to obtain the public key PK and the secret key SK. Next, the challenger signs each queried message as $\sigma_i \leftarrow \mathbf{Sign}(\mathrm{SK}, M_i)$ for $i = 1$ to n. The challenger then sends PK, $\sigma_1, \ldots, \sigma_n$ 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 $\mathbf{Verify}(\mathrm{PK}, M, \sigma) = 1$.

We define $\mathbf{Adv}_{\mathcal{A}}^{weak}$ to be the probability that adversary $\mathcal{A}$ wins in the above game.

Definition 2.2 (Unforgeability against Weak Chosen Message Attacks) *A signature scheme* (**KeyGen**, **Sign**, **Verify**) *is existentially unforgeable with respect to weak chosen message attacks if for all probabilistic polynomial time adversaries $\mathcal{A}$,* $\mathbf{Adv}_{\mathcal{A}}^{weak}$ *is negligible in λ.*

2.4 Chameleon Hashes

As formalized by Krawczyk and Rabin [17], a chameleon hash function H takes two inputs: a message m and randomness r. It is collision-resistant with the additional property that, given special trapdoor information, any target y and any message m', it is possible to efficiently find a value r' such that $H(m', r') = y$. Secure constructions exist in the standard model under the discrete logarithm assumption [17], the hardness of factoring [17], and the RSA assumption [1, 16]. See Appendix C for further details.

2.5 Generic Transformation

We now recall a generic construction for building unforgeable signatures out of weak unforgeable signatures and chameleon hashes, as used in many prior signature constructions such as [17, 26, 3, 16]. Let (G, S, V) be a weak unforgeable scheme for n-bit messages. Let chameleon hash family $\mathcal{H}$ map inputs as $\{0,1\}^\ell \times \{0,1\}^k \rightarrow \{0,1\}^n$. Consider a scheme for ℓ-bit messages constructed as:

KeyGen(1^λ): Select a random chameleon hash $H \in \mathcal{H}$. Run $G(1^\lambda)$ to obtain the keypair (pk, sk). The public key is $\mathrm{PK} = (pk, H)$ and the secret key is $\mathrm{SK} = (sk, H)$.

Sign($\mathrm{SK}, M \in \{0,1\}^\ell$): Pick a random $r \in \{0,1\}^k$. Compute $x = H(M, r)$, and then $\sigma' \leftarrow S(sk, x)$. Output the signature $\sigma = (\sigma', r)$.

Verify(PK, M, σ): Parse σ as (σ', r). Compute $x = H(M, r)$ and then output $V(pk, x, \sigma')$.

Lemma 2.3 *If (G, S, V) is a weakly-secure scheme according to Definition 2.2 and $\mathcal{H}$ is a secure chameleon hash family, then the above scheme is a fully-secure scheme according to Definition 2.1.*

While this construction is well known (e.g., [17, 26, 3, 16]), we were unable to find an explicit proof of the above lemma and thus provide one in Appendix A for completeness.

3 Algebraic Settings and Complexity Assumptions

3.1 RSA Assumption and other Facts

We begin by recalling some basic facts and complexity assumptions.

Assumption 3.1 (RSA [23]) *Let k be the security parameter. Let positive integer N be the product of two k-bit, distinct odd primes p, q. Let e be a randomly chosen positive integer less than and relatively prime to $\phi(N) = (p-1)(q-1)$. Given (N, e) and a random $y \in \mathbb{Z}_N^*$, it is hard to compute x such that $x^e \equiv y \mod N$.*

In the **Strong RSA** assumption, the adversary is given (N, y) and succeeds by producing any integer pair (e, x) such that $e > 1$ and $x^e \equiv y \mod N$. The standard RSA version is much more restrictive on the adversary.

In Section 4, we will restrict ourselves to the RSA assumption where $N = pq$ is the product of two safe primes $p = 2p' + 1$ and $q = 2q' + 1$. (Technically, we will want that the several prime exponents used during signing do not divide $\phi(N)$. While safe primes will make this argument simpler, they are not strictly necessary.)

Our RSA-based scheme will require a primality test, such as the efficient test of Miller and Rabin [19, 22]. We will also use the following facts.

Lemma 3.2 (Shamir [25]) *Given $x, y \in \mathbb{Z}_n$ together with $a, b \in \mathbb{Z}$ such that $x^a = y^b$ and $\gcd(a,b) = 1$, there is an efficient algorithm for computing $z \in \mathbb{Z}_n$ such that $z^a = y$.*

Theorem 3.3 (Prime Number Theorem) *Define $\pi(x)$ as the number of primes $\leq x$. For $x > 1$,*

$$\pi(x) > \frac{x}{\lg x}.$$

3.2 Bilinear Groups and the CDH Assumption

Let $\mathbb{G}$ and $\mathbb{G}_T$ be groups of prime order p. A *bilinear map* is an efficient mapping $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ which is both: (*bilinear*) for all $g \in \mathbb{G}$ and $a, b \leftarrow \mathbb{Z}_p$, $e(g^a, g^b) = e(g,g)^{ab}$; and (*non-degenerate*) if g generates $\mathbb{G}$, then $e(g,g) \neq 1$.

Assumption 3.4 (Computational Diffie-Hellman [7]) *Let g generate a group $\mathbb{G}$ of prime order $p \in \Theta(2^\lambda)$. For all p.p.t. adversaries $\mathcal{A}$, the following probability is negligible in λ:*

$$\Pr[a, b, \leftarrow \mathbb{Z}_p; z \leftarrow \mathcal{A}(g, g^a, g^b) : z = g^{ab}].$$

4 An RSA-Based Construction

4.1 A Weakly-Secure Scheme

Setup(1^λ) The setup algorithm chooses an RSA modulus N, such that $2^\ell < \phi(N) < 2^{\ell+2}$, where ℓ is another security parameter derived from 1^λ. It then chooses a random value $h \in \mathbb{Z}_N^*$.

Next, it chooses a random key K for the PRF function $F : \{0,1\}^* \rightarrow \{0,1\}^\ell$ and a random $c \in \{0,1\}^\ell$. It then establishes a function $H_{(\cdot)} : \{0,1\}^* \rightarrow \{0,1\}^\ell$ as follows:

$$H_{K,c}(z) = F_K(i, z) \oplus c,$$

where i, called the *resolving index* for z, is the smallest $i \geq 1$ such that $F_K(i,z) \oplus c$ is odd and prime.

The public key PK is (N, h, c, K), where anyone can compute $H()$ using c and K from the public key. The secret key SK is the factorization of N together with the (public) values (c, K), which are necessary for the signer to compute $H()$.

Sign(SK, $M \in \{0,1\}^n$) To sign messages larger than n bits, one could first apply a collision-resistant hash function to the message. Let $M^{(i)}$ denote the first i bits of M; that is, the length i prefix of M. For $i = 1$ to n, it computes $e_i = H_{K,c}(M^{(i)})$. Finally, it outputs the signature

$$\sigma = h^{\prod_{i=1}^n e_i^{-1}} \mod N.$$

Note: if any e_i divides $\phi(N)$, then σ may not be defined. In this event, the signer will output SK as the signature, since we are using safe primes and thus $2e_i + 1$ divides N. We will later argue that this event occurs with negligible probability.

Verify(PK, M, σ) The verification algorithm first computes the appropriate primes as follows: for $i = 1$ to n, it computes $e_i = H_{K,c}(M^{(i)})$. The algorithm accepts if and only if

$$\sigma^{\prod_{i=1}^n e_i} \equiv h \mod N.$$

4.2 Proof of Security

Theorem 4.1 (Weak Security under RSA) *If the RSA assumption holds when N is the product of two safe primes, then the above signature scheme is weakly unforgeable as in Definition 2.2.*

Proof. As in the stateful signatures of [16], our reduction will disregard all RSA challenges (N, e^*, y) where e^* is *not* an odd *prime* less than 2^ℓ. We recall from [16] that good challenges will occur with polynomial probability. By construction, $\phi(N) < 2^{\ell+2}$. We also know, by Theorem 3.3, that the number of primes $\leq 2^\ell$ is $\geq \frac{2^\ell}{\ell}$. Thus, a loose bound on the probability of e^* being a prime in the proper range is $(\frac{2^\ell}{\ell})/2^{\ell+2} = \frac{1}{4\ell}$.

Suppose there is an adversary $\mathcal{A}$ against the above signature scheme for n-bit messages that makes at most $q(\lambda)$ queries where $q()$ is a polynomial and succeeds in forging with probability ϵ. (We say q queries where it is clear from context.) We show that this adversary can be used to break (good challenges for) RSA with probability approximately $\epsilon/(qn\ell\lambda)$, where q, n, ℓ are all polynomial in the security parameter λ. On input (N, e^*, y), where e^* is an odd prime $< 2^\ell$, our RSA solver $\mathcal{B}$ proceeds as:

Setup: Adversary $\mathcal{A}$ must first provide $\mathcal{B}$ with the messages $M_1, \ldots, M_q$ on which it will request to see signatures. $\mathcal{B}$ wishes to guess the shortest prefix of M^*, the message on which the adversary will later forge, that is different from all other prefixes of $M_1, \ldots, M_q$.

- If $q = 0$, $\mathcal{B}$ guesses $w \in \{0, 1\}$ at random and sets value $t^* = 1$. When $\mathcal{A}$ does not ask for any signatures, then the first prefix (i.e., bit) of the forgery message M^* will be used later to embed the challenge, and $\mathcal{B}$ need only guess it with probability 1/2.
- If $q \geq 1$, the simulator guesses at random $1 \leq i^* \leq q$ (a message with the longest prefix in common with the forgery message[1]) and $1 \leq t^* \leq n$ (the length of the longest common prefix plus one). We will later argue that $\mathcal{B}$'s guesses are correct with probability $\geq 1/(qn)$. The values (i^*, t^*) define the t^*-bit string w comprised of the first $(t^* - 1)$ bits of M_{i^*} followed by the complement of M_{i^*}'s t^* bit. In other words, if $\mathcal{B}$'s guesses are correct, then we know that w is the t^*-bit prefix of the message on which the adversary will forge, and moreover, that no other signatures will be issued with this prefix.

Armed with this information, $\mathcal{B}$ proceeds to set up the public key as:

1. Select a random PRF seed K.
2. Select a random index $1 \leq j \leq \ell\lambda$ and set $c = F_K(j, w) \oplus e^*$.
3. Abort if any of the following conditions hold:
   (a) j is not the resolving index of $H_{K,c}(w)$.
   (b) Some prime is not locally unique or divides $\phi(N)$. Let $P(M_i)$ be the vector of n primes derived as $H_{K,c}(M_i^{(k)})$ for $k = 1$ to n. Abort if, for any i, $P(M_i)$ contains a repeated prime or a prime that divides $\phi(N)$ (i.e., a prime p such that $2p + 1$ divides N).
   (c) $e^* \in S$, where S is defined as the set of all unique primes across all vectors $P(M_i)$ for $i = 1$ to q.

[1] More than one message in $M_1, \ldots, M_q$ may share this longest common prefix. Guessing any one of them will suffice for this analysis.

4. Set
$$h = y^{\prod_{e_i \in S} e_i} \mod N.$$
The maximum size of S is $qn - 1$. To $\mathcal{A}$, h will appear to be distributed randomly in $\mathbb{Z}_N^*$.
5. Send the public key $\mathrm{PK} = (N, h, c, K)$ to $\mathcal{A}$.

Sign: $\mathcal{B}$ can create a signature on any message M provided during the Setup as follows.

1. Compute the vector of n primes $P(M)$ (i.e., the set $H_{K,c}(M_i^{(k)})$ for $k = 1$ to n).
2. Compute the signature as
$$\sigma = y^{\prod_{e_i \in [S - P(M)]} e_i} \mod N.$$

Extract from Forgery: Eventually, $\mathcal{A}$ will output a forgery (M^*, σ). If $M^{*(t^*)} \neq w$, then abort; the Setup guess was not correct.

Now, we wish to extract the RSA solution. Consider the vector of primes $P(M^*)$. If any member of $P(M^*)$ divides $\phi(N)$ (i.e., a prime p such that $2p+1$ divides N), then $\mathcal{B}$ can factor N and compute the RSA solution $y^{1/e^*} \mod N$.

Otherwise, let α be the number of times e^* appears in $P(M^*)$. We know from our Setup that $\alpha \geq 1$. Now, consider the following settings:

$$x = \sigma^{(e^*)^{\alpha-1} \prod_{e_i \in P(M^*), e_i \neq e^*} e_i}, \quad y = y, \quad a = e^*, \quad b = \prod_{e_i \in S} e_i$$

First, we see that $x^a = y^b$. Second, we know that $\gcd(a, b) = 1$, since all values are primes and $e^* \notin S$. Thus, $\mathcal{B}$ can apply Lemma 3.2 to efficiently compute a value $z \in \mathbb{Z}_N$ such that $z^a = y$. $\mathcal{B}$ outputs z as the RSA solution.

Analysis. We now argue that any successful adversary $\mathcal{A}$ against our scheme will have success in the game presented by $\mathcal{B}$. To do this, we first define a sequence of games, where the first game models the real security game and the final game is exactly the view of the adversary when interacting with $\mathcal{B}$. We then show via a series of claims that if a $\mathcal{A}$ is successful against Game j, then it will also be successful against Game $j+1$.

Game 1: This game is defined to be the same as the security game of the scheme.

Game 2: The same as Game 1, with the exception that $\mathcal{A}$ fails if some prime is not locally unique or divides $\phi(N)$ (as described in Setup abort condition (b)).

Game 3: The same as Game 2, with the exception that $\mathcal{A}$ fails if $e^* \in S$.

Game 4: The same as Game 3, with the exception that at the beginning of the game $\mathcal{B}$ guesses w as follows:

- if $q = 0$, w is chosen at random from $\{0, 1\}$;
- otherwise, a random pair (i^*, t^*) is chosen, where $1 \leq i^* \leq q$ and $1 \leq t^* \leq n$. Together with $M_1, \ldots, M_q$, this defines the string w as comprised of the first $(t^* - 1)$ bits of M_{i^*} followed by the complement of M_{i^*}'s t^*th bit.

Now $\mathcal{A}$ fails if the message on which he forges does not have prefix w.

Game 5: The same as Game 4, with the exception that $\mathcal{A}$ fails if the resolving index of $H_{K,c}(w)$ is greater than $\ell\lambda$.

Game 6: The same as Game 5, with the exception that at the beginning of the game $\mathcal{B}$ guesses an index $1 \leq j^* \leq \ell\lambda$ and $\mathcal{A}$ fails is the resolving index of $H_{K,c}(w)$ is not j^*.

Game 7: The same as Game 6, with the exception that at the beginning of the game $\mathcal{B}$ chooses a random PRF seed K (as before) and a random $e \in \{0,1\}^\ell$ and then sets $c = F_K(j^*, w) \oplus e$.

Game 8: The same as Game 7, with the exception that c is set as $c = F_K(j^*, w) \oplus e^*$, where e^* is the ℓ-bit prime from the RSA challenge.

Notice that Game 8 is exactly the view of the adversary when interacting with $\mathcal{B}$. In Appendix B, we complete this argument by linking the probability of $\mathcal{A}$'s success in these games via a series of claims. The only non-negligible probability gaps come between Games 3 and 4, where there is a factor $1/(qn)$ loss, and between Games 5 and 6, where there is a factor $1/(\ell\lambda)$ loss. □

4.3 Short, Fully-Secure RSA Signatures

We obtain a fully secure signature scheme by combining our RSA-based weakly unforgeable signatures with any suitable chameleon hash function. Standard model chameleon hashes exist under the hardness of factoring [17] and RSA [1]. The following result is immediate from Theorem 4.1 and Lemma 2.3.

Corollary 4.2 (Full Security under RSA) *Let (G', S', V') be the signature scheme described in Section 4.1. Let $\mathcal{H}$ be a chameleon hash function family secure under the RSA assumption. Let (G, S, V) be the signature scheme resulting from the generic transformation in Section 2.5 on (G', S', V') and $\mathcal{H}$. Then (G, S, V) is a fully-secure signature scheme, according to Definition 2.1, under the RSA assumption.*

The resulting signatures are very short. A signature contains one element from $\mathbb{Z}_N^*$ and one k-bit integer, where k is derived from the security parameter and the choice of the chameleon hash when using the standard model, RSA-based hash in [1, 16]. We provide more details in Appendix C.

5 Optimizations for the RSA Construction

While the main efficiency focus of this work is on simultaneously achieving a short public key and signature under RSA, we now briefly turn our attention to methods for improving the computational efficiency of these signatures. A significant computational overhead for both the signer and the verifier in our RSA scheme is the generation and testing of primes necessary to compute the hash function $H()$. The signer also must perform one exponentiation, where the exponent may be reduced modulo $\phi(N)$, while the verification cost is roughly n exponentiations of ℓ-bit exponents.

5.1 Online/Offline Signatures

In an *online/offline* signature as introduced by Even, Goldreich and Micali [9], the scheme is designed so that the signer can do the bulk of his computational work *before* the message is known to him. This paradigm is extremely useful for many applications which require a quick response time once a message comes in, but where the device may otherwise have relatively longer periods of inactivity. Fortunately, our RSA scheme (as well as our later CDH scheme) have this property.

To see this, recall the generic structure of our fully-secure signature scheme from Section 2.5. The signer can, offline, choose a random n-bit message X, sign X using the weakly-secure scheme, and then later use the trapdoor of the chameleon hash to link this signature to any desired message M. Thus, all of the expensive primality testing and the single exponentiation for our scheme can be performed offline by the signer. Indeed, this use of a chameleon hash to obtain online/offline signatures was previously suggested by Shamir and Tauman [26].

5.2 Send Resolving Indices with the Signature

One of the main verification costs is searching for the n resolving indices. Each signature verification requires an expected $n\ell$ primality tests; i.e., an expected ℓ per evaluation of $H_{K,c}(M^{(i)})$, for $i = 1$ to n. The number of primality tests could be reduced to n by sending a vector of resolving indices along with the signature. While the size of the signature would increase by roughly $n \cdot \log(\ell\lambda)$ bits (i.e., the number of resolving indices n by the representation of their maximum likely value $\ell\lambda$; see Claim B.4 for more), this is still considerably smaller than several prior tree-based approaches.

The danger of attack on this scheme is that a malicious signer will send a higher index than the resolving index. However, suppose that a maximum resolving index $T = \ell\lambda$ is posted in the public key and that honest verifiers reject if any of the sender's indices exceed this value. Then we can alter our prior proof to fit this variation as well. The simulator $\mathcal{B}$ behaves as before, except that she guesses which resolving index the adversary $\mathcal{A}$ will choose for the evaluation of w, between 1 and $T = \ell\lambda$, and later uses this value to extract the RSA solution. As $\mathcal{B}$ is already guessing the resolving index in our current reduction (see Game 6), there is no additional concrete security loss.

5.3 Using a Larger Alphabet

In the current scheme, the message is broken up into n 1-bit chunks, which each correspond to a prime exponent. Finding these n primes is costly for both the signer and the verifier, and the verification then requires n exponentiations (of one prime each). Suppose we break the message up into larger, k-bit chunks. The benefit of this approach would be that only n/k primes need now be found and used in verification. The drawback is that the concrete security of the scheme would decrease by a factor of $1/(2^k - 1)$, because now the simulator must guess within the chunk pointed to by (i^*, t^*), which of the $2^k - 1$ values the forger will later use. In the binary case, the bit pointed to by (i^*, t^*) was always the complement of M_{i^*}'s t^*th bit.

Considering $k = 2$, however, we cut the cost of signature generation and verification in half, for only a 1/3 reduction in concrete security. In some scenarios, this may be acceptable.

5.4 Using Smaller Prime Exponents

In the current scheme, primes are chosen to be of ℓ bits where 2^ℓ is roughly the size of $\phi(N)$. We could instead select ℓ to be smaller, but where 2^ℓ is still exponential in the security parameter. The main benefit of this approach would be a slightly more efficient scheme at the cost of a security proof with respect to a different variant of the RSA problem, namely, inverting RSA with a random prime exponent of bit-length less than or equal to ℓ.

6 A CDH-Based Construction

Our RSA proof techniques can be translated into the CDH setting as well. Interestingly, this provides new insights about the security of the only prior (stateless) scheme known to be secure under CDH in the standard model: the Waters signatures [27]. We present an entirely new method for reasoning about the *weak* unforgeability of these signatures under CDH. By adding a chameleon hash, we obtain a fully-secure scheme which is a derivative of the Waters signatures. The main contribution here is a much shorter, cleaner security argument as well as a demonstration that our RSA proof techniques are likely to be useful in other settings.

6.1 The Waters Signatures

Recall the Waters signature scheme [27], which is known to be fully-secure under the Computational Diffie-Hellman assumption in the standard model.

Setup(1^λ) The setup algorithm selects a bilinear group $\mathbb{G}$ of prime order $p > 2^\lambda$. It chooses a random exponent $a \in \mathbb{Z}_p$. Let n be a security parameter derived from λ. It chooses random group elements $g, v_0, v_1, \ldots, v_n \in \mathbb{G}$. The secret key is a and the public key is output as:

$$g, v_0, v_1, \ldots, v_n, e(g, g)^a.$$

Sign(SK, $M \in \{0,1\}^n$) The message space is treated as n-bits; to sign arbitrarily long messages one could first apply a collision-resistant hash function. Here M_i denotes the ith bit of M. The signer chooses a random $r \in \mathbb{Z}_p$ and then outputs the signature as:

$$\sigma_1 = g^a \left(v_0 \prod_{i=1}^{n} v_i^{M_i} \right)^r, \quad \sigma_2 = g^r.$$

Verify(PK, $M \in \{0,1\}^n, \sigma = (\sigma_1, \sigma_2)$) The verification algorithm uses the bilinear map to verify the signature by checking that

$$e(\sigma_1, g) = e(g, g)^a e(v_0 \prod_{i=1}^{n} v_i^{M_i}, \sigma_2).$$

6.2 Proof of Security

Theorem 6.1 (Weak Security under CDH) *If the CDH assumption holds in $\mathbb{G}$, then the Waters signature scheme is weakly unforgeable as in Definition 2.2.*

Proof. Suppose we have an adversary $\mathcal{A}$ against the above signature scheme that makes at most $q(\lambda)$ queries where $q()$ is a polynomial and succeeds in forging with probability ϵ. (We say q queries where it is clear from context.) We show that this adversary can be used to break CDH with probability $\geq \epsilon/(qn)$. On input (g, g^a, g^b), our CDH solver $\mathcal{B}$ proceeds as follows:

Setup: Adversary $\mathcal{A}$ must first provide $\mathcal{B}$ with the messages $M_1, \ldots, M_q$ on which it will request to see signatures. $\mathcal{B}$ wishes to guess the shortest prefix of M^*, the message on which the adversary will later forge, that is different from all other prefixes of $M_1, \ldots, M_q$.

- If $q = 0$, $\mathcal{B}$ guesses $w \in \{0, 1\}$ at random and sets value $t^* = 1$. When $\mathcal{A}$ does not ask for any signatures, then the first prefix (i.e., bit) of the forgery message M^* will be used later to embed the challenge, and $\mathcal{B}$ need only guess it with probability $1/2$.
- If $q \geq 1$, the simulator guesses at random $1 \leq i^* \leq q$ (a message with the longest prefix in common with the forgery message[2]) and $1 \leq t^* \leq n$ (the length of the longest common prefix plus one). We will later argue that $\mathcal{B}$'s guesses are correct with probability $\geq 1/(qn)$. The values (i^*, t^*) define the t^*-bit string w comprised of the first $(t^* - 1)$ bits of M_{i^*} followed by the complement of M_{i^*}'s t^* bit. In other words, if $\mathcal{B}$'s guesses are correct, then we know that w is the t^*-bit prefix of the message on which the adversary will forge, and moreover, that no other signatures will be issued with this prefix.

Armed with this information, $\mathcal{B}$ proceeds to set up the public key as:

1. Set $e(g, g)^\alpha = e(g^a, g^b)$, thus the secret key will implicitly be set to $\alpha = ab$.
2. Pick random values $y_0, \ldots, y_n \in \mathbb{Z}_p$.
3. Set $v_0 = g^{y_0} \prod_{i=1}^{t^*} (g^a)^{w_i}$.
4. For $i = 1$ to n, set
$$v_i = \begin{cases} g^{y_i} & \text{if } i > t^*; \\ g^{-a} g^{y_i} & \text{else if } w_i = 1; \\ g^a g^{y_i} & \text{otherwise } (w_i = 0). \end{cases}$$
Here w_i denotes the ith bit of w. The key observation here is that all g^a terms will cancel out for signatures with prefix w and that this won't be true for any other t^*-bit prefix.
5. Send $\mathrm{PK} = (g, v_0, \ldots, v_n, e(g, g)^\alpha)$ to $\mathcal{A}$.

Sign: $\mathcal{B}$ can create a signature on any message M provided during the Setup. Let $\beta = \sum_{i=1}^{t^*} w_i$ be the number of 1's in w. Let $\gamma = \sum_{i=1}^{t^*} m_i(1 - 2w_i)$, where m_i denotes the ith bit of M. Notice that $\beta + \gamma = \sum_{i=1}^{t^*} w_i + m_i(1 - 2w_i)$; this is equal to the number of bits that differ between w and the first t^* bits of M. By our setup, $\beta + \gamma \neq 0$ for all messages provided by the adversary.

1. Select a random value $r' \in \mathbb{Z}_p$.
2. Set $\sigma_2 = (g^{-b})^{1/(\beta+\gamma)} g^{r'}$; this implicitly sets $\sigma_2 = g^r$ with $r = -b/(\beta + \gamma) + r'$.
3. Set $\sigma_1 = \sigma_2^{y_0 + \sum_{i=1}^n m_i y_i} g^{ar'(\beta+\gamma)}$. To see that this is properly formed relative to σ_2, note that the value we want is:
$$\sigma_1 = g^{ab} \left(v_0 \prod_{i=1}^{n} v_i^{m_i} \right)^r = g^{ab} \left(g^{y_0 + \beta a} g^{\gamma a} g^{\sum_{i=1}^n m_i y_i} \right)^r = g^{ab} \left(g^{a(\beta+\gamma)} \right)^r \left(g^{y_0 \sum_{i=1}^n m_i y_i} \right)^r =$$
$$\left(g^{a(\beta+\gamma)} \right)^{r'} \left(g^{y_0 \sum_{i=1}^n m_i y_i} \right)^r = \sigma_2^{y_0 + \sum_{i=1}^n m_i y_i} g^{ar'(\beta+\gamma)}.$$
4. Output the signature (σ_1, σ_2).

[2] More than one message in $M_1, \ldots, M_q$ may share this longest common prefix. Guessing any one of them will suffice for this analysis.

Extract from Forgery: Eventually, $\mathcal{A}$ will output a forgery $(M, \sigma = (\sigma_1, \sigma_2))$. If $M^{(t^*)} \neq w$, then abort; the Setup guess was not correct. From the construction, one can see that $\mathcal{B}$'s guesses are correct with probability $\geq 1/(qn)$, because the distribution of the public key and signature responses is the same for all possible guesses. Now, to extract the CDH solution g^{ab}, the main idea is that the forgery is of the form $\sigma_1 = g^{ab} g^{zr}, \sigma_2 = g^r$ for a value z known to $\mathcal{B}$, and thus it can compute $\sigma_1/\sigma_2^z = g^{ab}$. To see this, let m_i denote the ith bit of M and observe that:

$$g^z = v_0 \prod_{i=1}^{n} v_i^{m_i} = g^{y_0 + \beta a} \prod_{i=1}^{t^*} g^{a m_i (1 - 2w_i)} \prod_{i=1}^{n} g^{m_i y_i},$$

and thus that

$$z = y_0 + \left(\beta + \sum_{i=1}^{t^*} m_i (1 - 2w_i) \right) a + \sum_{i=1}^{n} m_i y_i.$$

Observe that up to t^*, it holds that $m_i = w_i$. Recall that $\beta = \sum_{i=1}^{t^*} w_i$ was chosen as the number of 1's in w. Thus, $\beta + \sum_{i=1}^{t^*} w_i(1 - 2w_i) = \beta - \sum_{i=1}^{t^*} w_i$ is equal to zero and z simplifies to $y_0 + \prod_{i=1}^{n} g^{m_i y_i}$. □

6.3 Short, Fully-Secure CDH Signatures

We obtain a fully secure signature scheme by combining the above CDH-based weakly unforgeable signatures with any suitable chameleon hash function. Standard model chameleon hashes exist under the discrete-logarithm assumption [17] (and thus under CDH). The following result is immediate from Theorem 6.1 and Lemma 2.3.

Corollary 6.2 (Full Security under CDH) *Let (G', S', V') be the signature scheme described in Section 6.1. Let $\mathcal{H}$ be a chameleon hash function family secure under the CDH assumption. Let (G, S, V) be the signature scheme resulting from the generic transformation in Section 2.5 on (G', S', V') and $\mathcal{H}$. Then (G, S, V) is a fully-secure signature scheme, according to Definition 2.1, under the CDH assumption.*

The resulting signatures are fairly short. A signature contains two elements from $\mathbb{G}$ and one k-bit integer, where k is derived from the security parameter and the choice of the chameleon hash. Weak signing requires only two exponentiations, since the signer can choose $v_0, \ldots, v_n$ such that she knows their discrete logarithms base g. Verification requires only two pairings. Of course, this is mostly a theoretical exercise as the Waters signatures are more efficient on all counts.

7 Conclusion and Open Problems

In this work, we presented the first stateless signatures with short public keys and short signatures secure under the RSA assumption in the standard model. This answers a long-standing open problem as to whether or not such a construction was possible. Indeed, this is the only known scheme to satisfy all of the above requirements under a computational assumption with a short input and a single valid output.

The construction requires a new proof technique for reasoning about the security of signature schemes. We demonstrate that this technique is of broader interest, by showing how to apply it in the CDH setting to obtain a new security proof for the Waters signatures [27]. Interestingly, both our constructions are also *online/offline* signatures, where the vast majority of the signing computation can be done offline before the signer knows the message.

We leave open several interesting problems. The Waters signatures and our variant here offer short signatures, but a public key of $O(\lambda)$ elements, where λ is the security parameter. It is still unknown how to realize standard model CDH signatures where both the signatures and the public key are short. While we offer many computational optimizations for our RSA scheme in Section 5, it would also be of great practical significance to obtain faster signing and verification times. Finally, given the usefulness of signatures in designing stronger encryption, anonymous credentials, e-cash, etc., it would be worthwhile to revisit some of these systems and try to weaken the complexity assumptions on which they are founded.

Acknowledgments

We thank Amit Sahai for helpful discussions and the anonymous reviewers for their helpful comments.

References

[1] Giuseppe Ateniese and Breno de Medeiros. Identity-based chameleon hash and applications. In *Financial Cryptography*, volume 3110 of *Lecture Notes in Computer Science*, pages 164–180. Springer, 2004.

[2] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security (CCS)*, pages 62–73. ACM Press, 1993.

[3] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology – EUROCRYPT '04*, volume 3027 of *Lecture Notes in Computer Science*, pages 382–400. Springer, 2004.

[4] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004.

[5] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology – CRYPTO '04*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer, 2004.

[6] Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. *ACM Transactions on Information and System Security*, 3(3):161–185, 2000.

[7] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[8] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology — CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984.

[9] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital schemes. In *CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 263–275. Springer, 1990.

[10] March Fischlin. The Cramer-Shoup Strong-RSA signature scheme revisited. In *Public Key Cryptography (PKC)*, volume 2567 of *Lecture Notes in Computer Science*, pages 116–129. Springer, 2003.

[11] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 123–139. Springer, 1999.

[12] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Symposium on the Theory of Computing (STOC)*, pages 197–206, 2008.

[13] Eu-Jin Goh, Stanislaw Jarecki, Jonathan Katz, and Nan Wang. Efficient signature schemes with tight reductions to the Diffie-Hellman problems. *J. of Cryptology*, 20(4):493–514, 2007.

[14] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2), 1988.

[15] Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. In *Advances in Cryptology – CRYPTO '08*, volume 5157 of *Lecture Notes in Computer Science*, pages 21–38. Springer, 2008.

[16] Susan Hohenberger and Brent Waters. Realizing hash-and-sign signatures under standard assumptions. In *Advances in Cryptology – EUROCRYPT '09*, volume 5479 of *Lecture Notes in Computer Science*, pages 333–350. Springer, 2009.

[17] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *Network and Distributed System Security Symposium*, 2000.

[18] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *Symposium on Foundations of Computer Science (FOCS)*, pages 120–130. IEEE Computer Society, 1999.

[19] Gary L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and System Sciences*, 13:300–317, 1976.

[20] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Advances in Cryptology – CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer, 1992.

[21] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer, 1996.

[22] Michael O. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12:128–138, 1980.

[23] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. of the ACM*, 21(2):120–126, February 1978.

[24] Claus P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.

[25] Adi Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Transaction on Computer Systems*, 1:38–44, 1983.

[26] Adi Shamir and Yael Tauman. Improved online/offline signature schemes. In *Advances in Cryptology – CRYPTO '01*, volume 2139 of *Lecture Notes in Computer Science*, pages 355–367. Springer, 2001.

[27] Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology – EUROCRYPT '05*, volume 3494 of *Lecture Notes in Computer Science*, pages 320–329. Springer, 2005.

A Proof of Lemma 2.3

The proof of Lemma 2.3 appears implicitly in several works, and is closely related to an online/offline signature proof due to Shamir and Tauman [26]. We provide it here for completeness.

Proof. Suppose that adversary $\mathcal{A}$ has advantage ϵ in breaking the full security of the (**KeyGen**, **Sign**, **Verify**) signature scheme as given in the construction in Section 2.5. Then, we will construct an adversary $\mathcal{B}$ that either finds a collision in the chameleon hash function or breaks the weak security of the scheme (G, S, V).

Observe that $\mathcal{A}$ can make two types of forgeries. Let $(M, (\sigma', r))$ be the forgery output by $\mathcal{A}$. Either it is the case that the chameleon hash value corresponding to this forgery $x = H(M, r)$ was used in a previous signature or it was not. $\mathcal{B}$ may simply guess which type of forgery $\mathcal{B}$ will make.

If dealing with a type I forger, then $\mathcal{B}$ will find a collision for hash $H \in \mathcal{H}$ as follows.

Setup: Run G to obtain the keypair (pk, sk). Send $\mathcal{A}$ the PK $= (pk, H)$.

Queries: When $\mathcal{A}$ asks to see a signature on $M \in \{0,1\}^\ell$, pick a random $r \in \{0,1\}^k$, compute $x = H(M, r)$, compute $\sigma' = S(sk, x)$ and output the signature (σ', r). Record (M, r) in a table T.

Output: Eventually, $\mathcal{A}$ will output a forgery $(M^*, (\sigma^*, r^*))$, where $M^* \neq M_i$ for all $M_i \in T$ (since a valid forgery) and yet $H(M^*, r^*) = H(M', r')$ for at least one $(M', r') \in T$ (since a type I forgery). $\mathcal{B}$ outputs the pairs (M^*, r^*) and (M', r') as a collision.

If dealing with a type II forger that makes at most q queries, then $\mathcal{B}$ will break (G, S, V) on input pk as follows.

Setup: Pick a random chameleon hash $H \in \mathcal{H}$ with its trapdoor information. Output random values $x_1, \ldots, x_q \in \{0,1\}^\ell$ to the challenger, and obtain the response $(pk, \sigma_1, \ldots, \sigma_q)$. Then send $\mathcal{A}$ the PK $= (pk, H)$.

Queries: When $\mathcal{A}$ makes its ith signature request on $M \in \{0,1\}^\ell$, use the trapdoor information to find an r such that $x_i = H(M, r)$, then output the signature (σ_i, r). Record M in T.

Output: Eventually, $\mathcal{A}$ will output a forgery $(M^*, (\sigma^*, r^*))$, where $M^* \neq M_i$ for all $M_i \in T$ (since a valid forgery) and yet $H(M^*, r^*) \neq x_i$ for all x_i from the setup (since a type II forger). $\mathcal{B}$ outputs the forgery $(H(M^*, r^*), \sigma^*)$.

In both cases, $\mathcal{B}$ succeeds in his break, whenever $\mathcal{A}$ succeeds in hers. □

B Analysis of $\mathcal{A}$'s Probability of Success in the Section 4 Games

Define $\mathbf{Adv}_{\mathcal{A}}[\text{Game } x]$ as the advantage of adversary $\mathcal{A}$ in Game x.

Claim B.1 *If F is a secure pseudorandom function, then*

$$\mathbf{Adv}_{\mathcal{A}}[\text{Game 2}] = \mathbf{Adv}_{\mathcal{A}}[\text{Game 1}] - \mathit{negl}(\lambda).$$

Proof. First, consider the probability of a repeat occurring when n ℓ-bit primes are chosen at random. By Theorem 3.3, we know that the number of ℓ-bit primes is at least $2^\ell/\ell$. Thus, a repeat will occur with probability $< \sum^n n/(2^\ell/\ell) = n^2\ell/2^\ell$. The probability of no repeats within a batch of n when this experiment is repeated q times is therefore at most $qn^2\ell/2^\ell$. Since n, q and ℓ are all polynomial in the security parameter, this is a negligible amount.

Second, consider the probability of choosing one of the two odd primes that divide $\phi(N)$ when n ℓ-bit primes are chosen at random. Suppose, as is the case in our simulation, that N is not revealed until after the primes are chosen. Then this probability is $2\ell/2^\ell$.

In the actual experiments, primes are chosen as follows: given a random PRF key K and a constant c, the PRF is evaluated in increments until its output and the exclusive or of c are prime. However, the probability of a collision or of dividing $\phi(N)$ in the random experiments cannot differ non-negligibly from the probability of a collision or of dividing $\phi(N)$ in the PRF experiments or this fact would admit a feasible distinguishing attack against the PRF.

Thus, the overall difference in advantage between these games is negligible in λ. □

Claim B.2 *If F is a secure pseudorandom function, then*

$$\mathbf{Adv}_{\mathcal{A}}[\text{Game 3}] = \mathbf{Adv}_{\mathcal{A}}[\text{Game 2}] - \mathit{negl}(\lambda).$$

Proof. First, consider the probability that $e^* \in S'$, where S' is a set of nq randomly chosen ℓ-bit primes. By Theorem 3.3, we know that the number of ℓ-bit primes is at least $2^\ell/\ell$. Thus, the probability that $e^* \in S'$ is at most $nq\ell/2^\ell$. Since n, q and ℓ are all polynomial in the security parameter, this is a negligible amount.

In the set S, primes are chosen as follows: given a random PRF key K and a constant c, the PRF is evaluated in increments until its output and the exclusive or of c are prime. Thus, the probability that $e^* \in S$ cannot differ in a non-negligible amount from the probability that $e^* \in S'$ or this fact would admit a feasible distinguishing attack against the PRF.

Thus, the overall difference in advantage between these games is negligible in λ. □

Claim B.3

$$\mathbf{Adv}_{\mathcal{A}}[\text{Game 4}] \geq \frac{\mathbf{Adv}_{\mathcal{A}}[\text{Game 3}]}{qn}.$$

Proof. The only difference between these games is that $\mathcal{B}$ guesses a random prefix w. This value is used no where in the game. Logically, however, think of w as the shortest prefix of M^*, the forgery message, which differs from all prefixes of $M_1, \dots, M_q$. This prefix clearly must exist. Once the adversary makes a forgery, we only declare him successful if the prefix of his forged message matches $\mathcal{B}$'s guess, which will occur with $1/(qn)$ probability when $q \geq 1$ and with $1/2$ probability when $q = 0$. □

Claim B.4 *If F is a secure pseudorandom function, then*

$$\mathbf{Adv}_{\mathcal{A}}[\text{Game 5}] = \mathbf{Adv}_{\mathcal{A}}[\text{Game 4}] - \mathit{negl}(\lambda).$$

Proof. For randomly chosen PRF seed K, a random $c \in \{0,1\}^\ell$ and arbitrary input $w \in \{0,1\}^{\leq \ell}$, we want to show that the resolving index of $H_{K,c}(w)$ is less than $\ell\lambda$ with all but negligible probability.

First, consider the probability that after $T = \ell\lambda$ random choices in $\{0,1\}^\ell$ no primes are found. Let p be the probability that a random ℓ-bit number is a prime. Let X_i be a random variable representing whether an ℓ-bit number is prime, where $\Pr[X_i = 1] = p$ and $\Pr[X_i = 0] = 1 - p$. Let X be the sum of T X_i and $\mu = E[X] = \sum_{i=1}^{T} X_i$. Chernoff bounds (lower tail) where $0 < \delta \leq 1$ provide the following

$$\Pr[X < (1-\delta)\mu] < e^{\frac{-\mu\delta^2}{2}}.$$

By Theorem 3.3, we know that $p \geq \frac{1}{\ell}$. Set $\delta = \frac{\lambda - 1}{\lambda}$ and $T = \ell\lambda$, which implies that $\mu = \lambda$. Plugging this into the formula, we have that at least one prime is found with high probability:

$$\Pr[X < 1] < e^{-\frac{(\lambda-1)^2}{2\lambda}}.$$

In game 5, finding the resolving index of $H_{K,c}(w)$ involves computing $F_K(i, w) \oplus c$ for $i = 1$ until a prime is found. However, since we know that a random function would produce at least one prime in $T = \ell\lambda$ trials with high probability, then the same must hold true of $F_K(\cdot) \oplus c$ or this admits a distinguishing attack against the PRF. □

Claim B.5

$$\mathbf{Adv}_{\mathcal{A}}[\text{Game 6}] = \frac{\mathbf{Adv}_{\mathcal{A}}[\text{Game 5}]}{\ell\lambda}.$$

Proof. From game 5, we know that the resolving index of $H_{K,c}(w)$ is between 1 and $\ell\lambda$. In game 6, at the beginning of the game, $\mathcal{B}$ takes a random guess $j^* \in [1, \ell\lambda]$ and $\mathcal{A}$ is said to fail if j^* is not the resolving index of $H_{K,c}(w)$. Thus, $\mathcal{B}$'s guess is correct with probability $\geq 1/(\ell\lambda)$. □

Claim B.6

$$\mathbf{Adv}_{\mathcal{A}}[\text{Game 7}] = \mathbf{Adv}_{\mathcal{A}}[\text{Game 6}].$$

Proof. These games are identical. The only difference is in the *presentation* of how c is chosen. In both games, c is chosen uniformly at random in $\{0,1\}^\ell$. □

Claim B.7

$$\mathbf{Adv}_{\mathcal{A}}[\text{Game 8}] \geq \mathbf{Adv}_{\mathcal{A}}[\text{Game 7}].$$

Proof. Given the prior series of games, we know that $\mathcal{A}$ only succeeds in game 7 when the randomly chosen $z \in \{0,1\}^{\ell}$ happens to be prime. This must hold for the resolving index of $H_{K,c}(w)$ to be j^*. In game 8, we make it easier on the adversary by always substituting a random prime for z. Thus, we consider only the distributions in game 7 where $\mathcal{A}$ has a non-zero chance of success. □

C Chameleon Hash based on RSA

Krawczyk and Rabin [17] provide a chameleon hash function based on the hardness of factoring. For better efficiency, one could combine the following chameleon hash, due to Ateniese and de Mederios [1], with our RSA construction in Section 4 as was employed by the RSA signatures in [16]. The hash works as follows.

Let ℓ, α be security parameters, where α is a constant fraction of ℓ. Let N be an RSA modulus such that $2^{\ell} < \phi(N) < 2^{\ell+2}$. Choose a random, positive $e \in \{0,1\}^{\ell}$ which is relatively prime to $\phi(N)$ and a random value $J \in \mathbb{Z}_N$. Set the public key as (N, e, J) and keep as the trapdoor the factorization of N as well as a value d such that $ed \equiv 1 \mod \phi(N)$.

The hash $H : \{0,1\}^{\alpha} \times \mathbb{Z}_N \rightarrow \mathbb{Z}_N$ takes two inputs and produces one output. The hash is computed as $H(m,r) = J^m r^e \mod N$.

Given the trapdoor, anyone can compute a collision for any message m' by solving the following equation for r':

$$J^m r^e = J^{m'} r'^e \text{ as } r' = r(J^d)^{m-m'} \mod N.$$

Theorem C.1 ([16]) *The above chameleon hash function is secure under the RSA assumption in the standard model.*

Observe that by employing this scheme in Section 4.3, we add only the value r (an integer in $\mathbb{Z}_N$) to the size of the fully-secure signature.