Collision-Resistant No More: Hash-and-Sign Paradigm Revisited

Ilya Mironov

Microsoft Research (Silicon Valley Campus)
mironov@microsoft.com

Abstract. A signature scheme constructed according to the hash-and-sign paradigm—hash the message and then sign the hash, symbolically $\sigma(H(M))$—is no more secure than the hash function H against a collision-finding attack. Recent attacks on standard hash functions call the paradigm into question. It is well known that a simple modification of the hash-and-sign paradigm may replace the collision-resistant hash with a weaker primitive—a target-collision resistant hash function (also known as a universal one-way hash, UOWHF). The signer generates a random key k and outputs the pair $(k, \sigma(k||H_k(M)))$ as a signature on M. The apparent problem with this approach is the increase in the signature size. In this paper we demonstrate that for three concrete signature schemes, DSA, PSS-RSA, and Cramer-Shoup, the message can be hashed simultaneously with computing the signature, using one of the signature's components as the key for the hash function. We prove that our constructions are as secure as the originals for DSA and PSS-RSA in the random oracle model and for the Cramer-Shoup signature scheme in the standard model.

Keywords: TCR, UOWHF, collision-resistance, signatures, Cramer-Shoup, DSA, PSS-RSA.

1 Introduction

History of relation between cryptographically secure hash functions and digital signature schemes is one of co-evolution and divergence. Early constructions of dedicated hash functions were motivated by their applications to signature schemes [Riv91, NIS95]; by now the hash functions have extended their application domain to include MACs [BCK96] and public-key encryption [Sho00b]. In turn, unforgeability of many signature schemes crucially depends on security of the underlying hash function. This paper is concerned with divesting signature schemes of their reliance on collision-resistant hash functions by replacing them with a strictly weaker primitive. The general approach is well known; the novelty is in doing so without increasing the signature length.

Hash functions often play a dual role of a *domain extender* and a *random oracle* in constructions of signature schemes.

The first role, that of a domain extender, is due to the fact that it is much easier to design a scheme secure for signing messages of a fixed length than

M. Yung et al. (Eds.): PKC 2006, LNCS 3958, pp. 140–156, 2006.

of unrestricted length. Consider, for example, the RSA signature defined as $\sigma_{\mathrm{RSA}}(M) = M^d \mod N$. If the message domain were unrestricted, a forgery would be trivial since $\sigma_{\mathrm{RSA}}(M) = \sigma_{\mathrm{RSA}}(M + N)$. Virtually all practical signature schemes follow the hash-and-sign paradigm: apply a hash function to the message and sign the result, which we represent symbolically as $\sigma(H(M))$. The natural security requirement for H is that the hash function must be collision-resistant. Otherwise, if two messages have identical hashes $H(M_1) = H(M_2)$ a signature on one of them is a signature on the other. In light of recent attacks on standard hash functions, such as [WY05, WYY05a], feasibility of constructing efficient collision-resistant hash functions appears problematic; bypassing the requirement would make signatures more robust and may potentially increase their efficiency.

The following simple attack on the RSA function, whose domain is restricted to $1 \le M < N$, motivates the second role of hash functions: $\sigma_{\mathrm{RSA}}(M^2 \mod N) = \sigma_{\mathrm{RSA}}(M)^2 \mod N$. Coincidentally, hashing the message before applying the RSA function thwarts this attack, at least in practice. Many practical signature schemes are vulnerable to similar attacks, which are remedied by a judicious application of a hash function. Thus, the Fiat-Shamir heuristic [FS87], which gives a generic way of transforming an identification scheme into a signature, and the full-domain hash [BR93], which is suitable for signatures based on a trapdoor permutation such as RSA or Rabin functions, elevated the status of the hash function from a technical prop to an indispensable element of the construction, in the same time upping the ante for design of the hash function. Not only must the hash be collision-resistant, it should be a real-world implementation of a certain idealized abstraction, called the random oracle. This methodology is adopted by many practical signatures, although there is evidence that it may never be proved secure in the standard model [DOP05, PV05].

By explicitly decoupling the two roles of the hash function we can have more transparent security proofs and more efficient designs of signature schemes. In this paper we relax the collision-resistant requirement, without addressing the need for a random oracle. For the basic Cramer-Shoup signature scheme [CS00], provable in the standard model, this means a strictly better signature scheme (computationally equivalent scheme which relies on a weaker assumption). For discussion of our result as applied to two signature schemes provably secure in the random oracle model, DSA and PSS-RSA, see Section 6.

The primitive, which we prefer to collision-resistant hash functions, is due to Naor and Yung [NY89]. Simultaneous with development of practical signature schemes offering only heuristic security, a series of seminal papers [GMR88, NY89, Rom90] established that provably secure signature schemes can be constructed from one-way functions. An intermediate step of this construction is a family of universal one-way hash functions, also called target-collision resistant (TCR) hashes. TCR hashes is a class of *keyed* hash functions formally defined in Section 2. Further validating this approach, Simon [Sim98] demonstrated that a collision-resistant hash is a fundamentally stronger primitive (and hence more difficult to construct) than a TCR function by proving impossibility of a black-box construction of a collision-resistant hash from a one-way function.

Moreover, a TCR hash may replace a collision-resistant function as the first step of the hash-and-sign signature scheme. Most importantly, it can be done via a little tweak of the hash-and-sign paradigm rather than by going through the theoretically secure but inefficient construction of [NY89]. Informally, if $\sigma(\cdot)$ is secure for signing fixed-length messages and $H_k(\cdot)$ is TCR, the hybrid scheme $(k, \sigma(k||H_k(M)))$, where key k is chosen uniformly at random by the signer, is secure as well. The obvious problem with the scheme is the increase in the signature length, since the key k becomes part of the signature (discussion of the length of the key is deferred to Section 6).

We observe that for many signature schemes, such as Cramer-Shoup, those based on Fiat-Shamir heuristic and probabilistic full-domain hash, the signature *already* includes some randomly generated data, which is independent of the message. We demonstrate that this data can double as the key of the TCR hash, thus eliminating the need for extra key material. The resulting schemes retain the signature length of the originals and are at least as secure.

Some schemes and results of this paper were independently discovered and presented by Halevi and Krawczyk [HK05a, HK05b].

2 Definitions

Our definitions of signature schemes and TCRs follow [GMR88] and [NY89] except that we recast the definitions in the language of exact security.

Signature scheme. A signature scheme $\mathcal{S}$ consists of a triple of algorithms:

- Key generation algorithm $\mathit{KeyGen}(1^k) = (\mathrm{PK}, \mathrm{SK})$, a randomized algorithm producing a public-private key pair for a given security parameter.
- Verification algorithm $\mathit{Verify}_{\mathrm{PK}}(M, \sigma) \in \{\texttt{accept}, \texttt{reject}\}$. We say that σ is a valid signature on M if $\mathit{Verify}_{\mathrm{PK}}(M, \sigma) = \texttt{accept}$.
- Signing algorithm $\mathit{Sign}_{\mathrm{SK}}(M) = \sigma \in \{0,1\}^n$. The signing algorithm outputs a valid signature on M with an overwhelming probability taken over its own coin tosses.

A signature scheme $\mathcal{S}$ is (t, ε, q_S)-secure against existential forgery under adaptive chosen-message attack if the attacker running in time less than t and making no more than q_S signing queries cannot succeed with probability more than ε in producing a valid signature on a message M, which has not been previously signed by the signing oracle. In other words, the adversary obtains at most q_S valid signatures on adaptively chosen (queries may depend on the answers to the previous queries) messages. The adversary wins if he can compute a valid signature on a message not in the list of queries.

TCR. A (t, ε)-target collision-resistant hash function (TCR), also known as a universal one-way hash function (UWOHF) is a keyed hash function $H : \{0,1\}^k \times \{0,1\}^* \mapsto \{0,1\}^n$ such that no adversary running in time less than t can win the following game with probability more than ε:

Step 1. Output $X \in \{0,1\}^*$.
Step 2. Receive K randomly chosen from $\{0,1\}^k$.
Step 3. Produce Y so that $H_K(X) = H_K(Y)$.

As a warm-up exercise, we sketch a proof that combining a TCR hash with a signature scheme secure for signing fixed-length strings results in a signature scheme secure for messages of unrestricted length.

Proposition 1. *Assume a signature scheme $\mathcal{S} = (KeyGen, Sign, Verify)$ is (t, ε_S, q_S)-secure against existential forgery under an adaptive chosen-message attack where the messages are restricted to length n. Assume further that $H : \{0,1\}^k \times \{0,1\}^* \mapsto \{0,1\}^{n-k}$ is a (t, ε_H)-TCR. Then there exists a $(t, \varepsilon_S + \varepsilon_H q_S, q_S)$-secure signature scheme for arbitrary-length messages.*

Proof. Let the signature scheme $\mathcal{S}'$ be the following:

Sign':	*Verify'*:
Step 1. Generate $K \xleftarrow{\$} \{0,1\}^k$.	Step 1. Parse signature as (K, σ).
Step 2. $\sigma \leftarrow Sign_{\mathrm{SK}}(K \| H_K(M))$.	Step 2. Run $Verify_{\mathrm{PK}}(K \| H_K(M), \sigma)$.
Step 3. Output signature $K \| \sigma$.	

($KeyGen$ is the same as in $\mathcal{S}$).

Assume that there exists an adversary capable of producing a valid signature $(M, (K, \sigma))$ having queried the signing oracle on messages $M_1, \ldots, M_{q_S}$, such that $M \neq M_i$ for $1 \leq i \leq q_S$. Let the signatures output by the oracle be $(K_1, \sigma_1), \ldots, (K_{q_S}, \sigma_{q_S})$. Two cases are possible. Either $K \| H_K(M) \neq K_i \| H_{K_i}(M_i)$ for all $i \in [1, q_S]$ or there is i such that $K \| H_K(M) = K_i \| H_{K_i}(M_i)$ and $M \neq M_i$. In the former case the adversary can be trivially used to forge a signature for the scheme $\mathcal{S}$. In the latter, draw a random index $j \in [1, q_S]$ and, when the adversary makes query M_j, send M_j as the first message of the TCR-game. Upon receiving key K', set $K_j = K'$. With probability $1/q_S$ the adversary outputs a message-signature pair so that $K \| H_K(M) = K' \| H_{K'}(M_j)$. Since the keys have fixed size k bits, it follows that $K = K'$ and we win the TCR-game by outputting M, which collides with M_j under key K'.

The probability that a t-time adversary forges a signature is less than the sum of ε_S—the probability that he succeeds in breaking the signature scheme $\mathcal{S}$—and $\varepsilon_H q_S$, where ε_H is the probability that it breaks H. □

Pseudo-random generator. We say that a function $F : A \mapsto B$ is (t, ε, q_F)-pseudo-random generator if no adversary running in time less than t and making less than q_F queries of F can distinguish $F(x)$, where $x \xleftarrow{\$} A$, from the uniform distribution on B with probability more than ε. We relax the standard definition [BM82] by dropping the usual requirement that the function stretches its input (i.e., that $|A| < |B|$). Although compressing pseudo-random generators are trivial to construct, the assumption that a particular function, such as SHA-1, is a pseudo-random generator, is substantive.

3 DSA Scheme

We present the original DSA scheme together with our variant, which we call TCR-DSA, see Figure 1. The new signature scheme uses three hash functions: $H\colon \{0,1\}^{\ell_2} \times \{0,1\}^* \mapsto \{0,1\}^{\ell_1}$, which we assume to be a (t_H, ε_H)-TCR, and two functions $F_1\colon \mathbb{Z}_p \mapsto \{0,1\}^{\ell_2}$ and $F_2\colon \{0,1\}^{\ell_1+\ell_2} \mapsto \{0,1\}^n$, which we model as random oracles.

DSA	TCR-DSA

Key selection:

p, q—prime, $|p| = n$, $|q| = m$, $p|q-1$
$g \in \mathbb{Z}_q$, ord $g = p$
$a \xleftarrow{\diamondsuit} \mathbb{Z}_p$; $h = g^a \bmod q$
public key: p, q, g, h
private key: a

Hash functions:

DSA	TCR-DSA
$G\colon \{0,1\}^* \mapsto \{0,1\}^n$	$H\colon \{0,1\}^{\ell_2} \times \{0,1\}^* \mapsto \{0,1\}^{\ell_1}$—TCR $\left.\begin{array}{l} F_2\colon \{0,1\}^{\ell_1+\ell_2} \mapsto \{0,1\}^n \\ F_1\colon \mathbb{Z}_p \mapsto \{0,1\}^{\ell_2} \end{array}\right\}$ random oracles

Signature generation:

DSA	TCR-DSA
$k \xleftarrow{\diamondsuit} \mathbb{Z}_p$	$k \xleftarrow{\diamondsuit} \mathbb{Z}_p$
$r = (g^k \bmod q) \bmod p$	$r = (g^k \bmod q) \bmod p$
	$\boxed{r_1 = F_1(r)}$
$s = k^{-1}(\boxed{G(M)} + ra) \mod p$	$s = k^{-1}(\boxed{F_2(r_1, H_{r_1}(M))} + ra) \mod p$
$\sigma = (r, s)$	$\sigma = (r, s)$

Signature verification:

DSA	TCR-DSA
	$\boxed{r_1 = F_1(r)}$
$u = g^{\boxed{G(M)}} h^r \bmod q$	$u = g^{\boxed{F_2(r_1, H_{r_1}(M))}} h^r \bmod q$
$w = u^{s^{-1} \bmod p} \bmod q$	$w = u^{s^{-1} \bmod p} \bmod q$
accept if $w \bmod p = r$	accept if $w \bmod p = r$

Fig. 1. DSA and TCR-DSA (differences are enclosed in boxes)

In this section we tie security of TCR-DSA to that of the DSA instantiated with any concrete function, which is a good $\{0,1\}^{2n} \mapsto \{0,1\}^n$ pseudo-random generator, and under the δ-min-entropy of r assumption (defined below) in the random oracle model.

First, we formulate an assumption on uniformity of r (in both schemes), also discussed in [Bro05].

Assumption of "δ-min-entropy of r". Define the min-entropy of distribution $\mathcal{D}$ as

$$H_\infty(\mathcal{D}) = -\log \max \Pr[x \in \mathcal{D}].$$

Let $\mathcal{R}$ be the distribution of $r = (g^k \mod q) \mod p$, where k is uniform in $\mathbb{Z}_p$. We assume that $H_\infty(\mathcal{R}) > \delta$.

[NS02, Lemma 10] proves that r has min-entropy $O(\delta \log p)$ for some δ that depends on $\log q / \log p$. In practice, we expect r to be distributed much smoother, having min-entropy of the order of $\log p - c \log\log p$ for some small c (consider the occupancy problem applied to p balls and 2^n bins).

Theorem 1. *Under the assumptions that*

- *DSA is $(t, \varepsilon_{\mathrm{DSA}}, q_S)$-secure for some $G\colon \{0,1\}^* \mapsto \{0,1\}^n$;*
- *G restricted to inputs of length $2n$ is (t, ε_G)-pseudo-random generator;*
- *H is (t, ε_H)-TCR;*
- *r has $(\log p - \delta)$-min-entropy;*
- *F_1, F_2 are modeled as random oracles, which together are queried no more than q_F times;*

then TCR-DSA is $(t, 2\varepsilon_{\mathrm{DSA}} + \varepsilon_G + \varepsilon_H q_S + (2^{-\delta} q_S p + q_F)2^{-\delta} q_S, q_S)$-secure.

Proof. We demonstrate how to transform any forgery of TCR-DSA into either an attack on H as a TCR or a forgery of DSA instantiated with G. We do so by defining Game 0 that consists of the challenger interacting with the TCR-DSA adversary $\mathcal{A}$ and the DSA signing oracle. $\mathcal{A}$ queries F_1 and F_2, requests signatures, and attempts to forge a TCR-DSA signature. In the spirit of [Sho04] we describe a sequence of games that transforms the initial game to one whose success probability we can easily analyze.

Game 0. Obtain the public key for the DSA oracle and pass it on as the public key of TCR-DSA. We keep two lists L_1, L_2, initially empty, of inputs on which F_1 and F_2 are defined. Queries to F_1 are answered randomly; queries to F_2 are answered by randomly choosing $M' \xleftarrow{\$} \{0,1\}^{2n}$ and returning $G(M')$ (M' is stored; if M' appeared previously, the process is repeated). Notice that under the assumption of computational indistinguishability of G's output, F_2 cannot be distinguished from a true random oracle with probability more than ε_G. Upon receiving a new signing query M do the following:

Step 1. Generate $M' \xleftarrow{\$} \{0,1\}^{2n}$. Repeat if M' appeared previously.
Step 2. Obtain (r, s) by querying the DSA signing oracle on M'.
Step 3. Fail if $r \in L_1$. Define $r_1 = F_1(r)$ randomly, appending the result to L_1.
Step 4. Fail if $(r_1, H_{r_1}(M)) \in L_2$. Otherwise let $F_2(r_1, H_{r_1}(M)) = G(M')$, add $(r_1, H_{r_1}(M))$ to L_2 and store M' together with $r_1, H_{r_1}(M)$.
Step 5. Output (r, s) as a TCR-DSA signature on M.

Finally, if $\mathcal{A}$ outputs $M^*, (r^*, s^*)$ as a forgery of TCR-DSA, do the following:

Step 6. Compute $r_1^* = F_1(r^*)$.
Step 7. Fail if F_2 has not been queried on $(r_1^*, H_{r_1^*}(M^*))$.
Step 8. Fetch M_0^* such that $F_2(r_1^*, H_{r_1^*}(M^*)) = G(M_0^*)$.

Step 9. Fail if the DSA oracle has been queried on M_0^*.
Step 10. Output $M_0^*, (r^*, s^*)$ as a DSA forgery.

Observe that if Game 0 succeeds, the challenger aided by $\mathcal{A}$ queried the DSA oracle no more than q_S times and successfully forged a DSA signature. The probability of this event is no more than $\varepsilon_{\mathrm{DSA}}$. In order to complete the proof we shall bound the probability that Game 0 fails (Steps 3, 4, 7, 9).

To bound the failure probability of Step 3 of Game 0 we need the following lemma.

Lemma 1. *Let $\mathcal{D}$ be a distribution on set X. Let $\tau = 2^{-H_\infty(\mathcal{D})}|X|$. We claim that for any set $A \subset X$ and any $x_1, \ldots, x_n \xleftarrow{\text{¢}}_{\mathcal{D}} X$ (n elements chosen from X independently at random according to $\mathcal{D}$) the following holds:*

$$\Pr[\exists i, j (i \neq j, x_i = x_j) \bigvee \exists i (x_i \in A)] < (\tau^2 n^2)/|X| + \tau|A|n/|X|.$$

Proof. Observe that

$$\Pr[\exists i, j \colon i \neq j, x_i = x_j] \leq E[\#\{i < j \colon x_i = x_j\}] = \sum_{i<j} E[x_i = x_j] < n^2\tau^2/|X|. \quad (1)$$

To analyze the probability that $x_i \in A$ for some i, consider $p = \Pr[x \xleftarrow{\text{¢}}_{\mathcal{D}} X \colon x \in A]$. Then, $\Pr[\exists i \colon x_i \in A] = 1 - (1-p)^n < pn$. Further, $p = \sum_{a \in A} \Pr[x \xleftarrow{\text{¢}}_{\mathcal{D}} X \colon x = a] < \tau|A|/|X|$, which, together with (1), completes the proof. □(Lemma 1)

By applying Lemma 1 to the distribution of r and A defined as the set of inputs on which F_1 is queried directly, we obtain that Step 3 fails with probability at most $2^{-2\delta} q_S^2 p + 2^{-\delta} q_F q_S = (2^{-\delta} q_S p + q_F) 2^{-\delta} q_S$.

Since r_1 never repeats, the probability that Step 4 fails is at most $q_F^2 2^{-\ell_2/2}$.

If Step 7 fails, it means that the forger produced a valid signature (r^*, s^*) without knowing the value of $(r_1^*, H_{r_1^*}(M^*))$. Since the value is distributed randomly, same forger can be used against DSA. The probability of the failure is thus at most $\varepsilon_{\mathrm{DSA}}$.

Now we rewrite Step 3 of Game 0, replacing it with the following:

Step $3a'$. Submit M as the first move of the TCR game.
Step $3b'$. Obtain $\kappa \in \{0,1\}^{\ell_2}$ as the key. Set $r_1 = F_1(r) = \kappa$.

Step 9 fails if the DSA oracle has been previously called on M_0^*, which can only happen if $(r_1^*, H_{r_1^*}(M^*)) = (r_1', H_{r_1'}(M'))$ for some other M'. It implies that $r_1^* = r_1'$ and the result is obviously a collision under $H_{r_1}(\cdot)$, which we output by rewriting Step 9:

Step $9'$. If DSA oracle has been queried on M_0^*, fetch $(r_1', H_{r_1'}(M')) = (r_1^*, H_{r_1^*}(M^*))$. Find the game (started in Step $3a'$), where the M' was the first move and r_1' was the key. Complete the game by outputting M^*. Fail.

The new game fails with exactly the same probability as Game 0. To complete the proof we notice that Step 9′ fails with probability at most $q_S \varepsilon_H$. □[Theorem 1]

We proved that TCR-DSA signature scheme is as secure as DSA for *arbitrary G*, which can, in particular, be modeled as a random oracle, or be extremely slow and provably (under, say, the discrete-logarithm assumption) collision-resistant. Although a direct proof of security of TCR-DSA under some standard assumptions would be tempting, we are concerned with the *tightness* of the reduction. Best known reductions, even in the random oracle model, tie the forgery probability of DSA variants to the hardness of discrete logarithm with q_F (number of random oracle queries) factor [BPVY00]. In order to shave off the factor, we may either assume additionally that the underlying group can be accurately modeled in the generic group model [Bro05], or make some non-standard assumptions [PV05]. Our reduction is tight in respect to the forgery probability of DSA and loose with respect to the security of H. The latter is hardly a bottleneck, since neither the key nor the output length of H affects the length of the signature and therefore boosting security of H should only be constrained by efficiency considerations.

Our proof does rely on one non-standard assumption, that of δ-min entropy of r. For our result to be meaningful, δ should be sufficiently high (of the order of $\log p$), which ensures that all values of r are unique with high probability. Two points are in order. First, for $\delta \approx -\log \varepsilon$ the assumption can be derived from $(t, \varepsilon, 0)$-security of DSA, where t is the time required to do 2^δ DSA verifications. Second, the proof of Theorem 1 can be restructured to accommodate $\delta \approx -\log \varepsilon$. The proof will appear in the full version of the paper.

4 RSA-PSS Signature Scheme

RSA-based probabilistic signature scheme (PSS-RSA) was proposed by Bellare and Rogaway [BR96] as a strengthening of the full domain hash scheme [BR93]. PSS-RSA enjoys *tight security* reduction to the underlying hard problem—the RSA assumption—in the random oracle model. In other words, assuming that certain hash functions are ideal, forging PSS-RSA is computationally equivalent to inverting the RSA function. Later, Coron proved an even tighter reduction to the RSA assumption [Cor02], which we use as a basis for our security claim.

Strictly speaking, PSS-RSA does not follow the hash-and-sign paradigm, since the message is concatenated with some random *salt* and only then is hashed using a hash function, modeled as a random oracle. We propose to hash the message, whose length is unrestricted, using a conventional TCR function keyed with the salt, and hash the short TCR function's output with a conservatively designed "oracle" (see Section 6).

Security of TCR-PSS-RSA (see Figure 2) relies on the following:

(t, ε)-RSA assumption. No algorithm running in time less than t can solve $x^r = y \bmod N$ for x with probability more than ε, where N is a random RSA modulus, $y \xleftarrow{\$} \mathbb{Z}_N^*$, and r is fixed.

Key selection:

p, q—prime, $|p| = |q| = n/2$, $N = pq$
$e, d \in Z_N$, $ed = 1 \pmod{N}$
public key: N, d
private key: e

Hash functions:

RSA-PSS	TCR-RSA-PSS
	$H\colon \{0,1\}^{k_0} \times \{0,1\}^* \mapsto \{0,1\}^{k_1}$—TCR
$h\colon \boxed{\{0,1\}^*} \mapsto \{0,1\}^{k_1}$	$h\colon \boxed{\{0,1\}^{k_0+k_2}} \mapsto \{0,1\}^{k_1}$
$g_1\colon \{0,1\}^{k_1} \mapsto \{0,1\}^{k_0}$	$g_1\colon \{0,1\}^{k_1} \mapsto \{0,1\}^{k_0}$
$g_2\colon \{0,1\}^{k_1} \mapsto \{0,1\}^{n-k_0-k_1-1}$	$g_2\colon \{0,1\}^{k_1} \mapsto \{0,1\}^{n-k_0-k_1-1}$

h, g_1, g_2—random oracles

Signature generation:

$r \xleftarrow{\$} \{0,1\}^{k_0}$	$r \xleftarrow{\$} \{0,1\}^{k_0}$
$\boxed{w = h(M\|\|r)}$	$\boxed{w = h(r\|\|H_r(M))}$
$r^* = g_1(w) \oplus r$	$r^* = g_1(w) \oplus r$
$y = 0\|\|w\|\|r^*\|\|g_2(w)$	$y = 0\|\|w\|\|r^*\|\|g_2(w)$
$\sigma = y^d \mod N$	$\sigma = y^d \mod N$

Signature verification:

$y = \sigma^e \mod N$	$y = \sigma^e \mod N$
check $y = b\|\|w\|\|r^*\|\|\gamma$	check $y = b\|\|w\|\|r^*\|\|\gamma$
$r = r^* \oplus g_1(w)$	$r = r^* \oplus g_1(w)$
accept if $\begin{cases} \boxed{h(M\|\|r) = w} \\ g_2(w) = \gamma \\ b = 0 \end{cases}$	accept if $\begin{cases} \boxed{h(r\|\|H_r(M)) = w} \\ g_2(w) = \gamma \\ b = 0 \end{cases}$

Fig. 2. PSS-RSA and TCR-PSS-RSA (differences are enclosed in boxes)

Theorem 2. *If all of the following hold:*

- $(t, \varepsilon_{\mathrm{RSA}})$*-RSA assumption;*
- H *is* (t, ε_H)*-TCR;*
- h, g_1, g_2 *are modeled as random oracles and queried no more than* q_F *times;*

then TCR-PSS-RSA is $(t, \varepsilon_{\mathrm{RSA}}(1 + 6q_S 2^{-k_0} + 2(q_S + q_F)^2 2^{-k_1} + \varepsilon_H q_S, q_S)$*-secure.*

Proof [sketch]. Let $M_1, \ldots, M_{q_S}$ be the signing queries made by the adversary, $r_1, \ldots, r_{q_S}$ be the r values from the corresponding signatures output by the signer, $M' \neq M_i$ for $1 \leq i \leq q_S$ be the message with forged signature, and r' be its r-value. The proof from [Cor02] of PSS-RSA applies virtually without any changes, where $H_r(M)$ replaces M. We have to make sure that Coron's proof also rules out "weak" forgeries (which corresponds to $H_{r'}(M') = H_{r_i}(M_i)$, and $r' \neq r_i$ for some i), which it does, and bound the probability that $H_{r_i}(M_i) = H_{r'}(M')$ and $r_i = r'$.

To bound the probability of the latter event, consider a simulator that knows the private key of the signature scheme. Upon receiving a signing query for M_i, it starts a TCR game, submitting M_i as the first message. It receives a random key r_i, which it uses in computing a signature on M_i. If the adversary succeeds in creating a collision, the simulator is able to complete one of the TCR games. □

5 Cramer-Shoup Signature Scheme

Historically, the Cramer-Shoup signature scheme [CS00] was the first efficient signature scheme provably secure in the standard model (i.e., without random oracles). The scheme relies on the strong RSA assumption, introduced in [BP97].

The basic Cramer-Shoup signature scheme, which uses a collision-resistant hash function, was presented in [CS00] together with a variant, where the hash

CS	TCR-CS
Key selection:	
p, q—strong primes, $\lvert p\rvert = \lvert q\rvert = \ell_1$, $n = pq$	
$h, x \xleftarrow{\text{¢}} \mathrm{QR}_n$; $e' \xleftarrow{\text{¢}} \mathbb{P}_{\ell+1}$	
$k' \xleftarrow{\text{¢}} \{0,1\}^{\ell_2}$	
public key: n, h, x, e', k'	
private key: p, q	
Hash function	
$H\colon \{0,1\}^{\ell_2} \times \{0,1\}^* \mapsto \{0,1\}^{\ell}$—TCR	
	$\mu\colon \{0,1\}^{\ell+1} \mapsto \{0,1\}^{\ell_2}$—projection
Signature generation:	
generate $e \in \mathbb{P}_{\ell+1}$, $e \neq e'$	generate $e \in \mathbb{P}_{\ell+1}$, $e \neq e'$
$y' \xleftarrow{\text{¢}} \mathrm{QR}_n$	$y' \xleftarrow{\text{¢}} \mathrm{QR}_n$
$\boxed{k \xleftarrow{\text{¢}} \{0,1\}^{\ell_2}}$	
$x' = (y')^{e'} h^{-\boxed{H_k(M)}}$	$x' = (y')^{e'} h^{-\boxed{H_{\mu(e)}(M)}}$
$y = \left(x h^{-\boxed{H_{k'}(k,x')}}\right)^{1/e}$	$y = \left(x h^{\boxed{H_{k'}(x')}}\right)^{1/e}$
$\boxed{\sigma = (e, y, y', k)}$	$\boxed{\sigma = (e, y, y')}$
Signature verification:	
check e is odd, $\lvert e\rvert = \ell + 1$, $e \neq e'$	check e is odd, $\lvert e\rvert = \ell + 1$, $e \neq e'$
check $x' = (y')^{e'} h^{-\boxed{H_k(M)}}$	check $x' = (y')^{e'} h^{-\boxed{H_{\mu(e)}(M)}}$
check $x = y^e h^{-\boxed{H_{k'}(k,x')}}$	check $x = y^e h^{-\boxed{H_{k'}(x')}}$

Fig. 3. CS and TCR-CS (differences are enclosed in boxes). $\mathbb{P}_\ell$ is the set of prime numbers of length ℓ; QR_n is the set of quadratic residues modulo n. Function $\mu(\cdot)$ returns most significant ℓ_2 bits of the input.

function is presumed to be a TCR. Our scheme combines the short signature length of the former and security of the latter. The TCR-based variant (CS) and our scheme (TCR-CS) are compared in Figure 3.

CS scheme makes use of a TCR by applying the generic transformation, outlined in the introduction,—the TCR's key is generated by the signer and transmitted as part of the signature. We propose to derive the hash function's key from the randomly chosen prime number e, which is already included in the signature.

Our proof follows closely Cramer-Shoup's original proof [CS00]. The main technical difficulty in constructing the reduction, which is not present in the original proof, consists in incorporating the hash function's key into a prime with a special structure. We expand on it below.

In addition to standard modular arithmetic, in the CS scheme the signer generates a *fresh* $(l+1)$-bit prime e with each signature (following [CS00] we assume $\ell = 160$). The prime numbers need not be uniformly distributed; the only requirement is that the probability that the same prime number is generated twice be negligible. Efficiency of the scheme (especially relative to [GHR99], which is otherwise comparable in terms of security and signature length) critically depends on the signer's ability to generate primes quickly. To this end, [CS00] proposes to use primes of special structure, namely $e = 2PR + 1$, where P is a 53-bit prime, which can be tested for primality much faster than the average $(\ell+1)$-bit integer. For TCR-CS to be as efficient, we would prefer to use the same procedure. On the other hand, our reduction technique prescribes taking a random key obtained as part of the TCR game and using it in the signature. If e were a random 161-bit prime, doing so would be trivial—a random 161-bit number is prime with probability approximately 1/112, hence the reduction would only suffer a factor of 112. Primes of the special structure $2PR + 1$, where $|P| = 53$, are more rare (a random 161-bit number has this structure with probability no more than 2^{-13}), and testing for it is prohibitively expensive.

Instead of using e as a key, we solve the problem by taking $\mu(e)$, where $\mu\colon \{0,1\}^{161} \mapsto \{0,1\}^{106}$ is a projection function, which simply drops 55 least significant bytes of its input. To reverse the procedure, which is what the reduction is to do, we adapt the prime generation algorithm from [CS00]. For a given key $k \in \{0,1\}^{106}$, generate a random prime P in the range $(2^{52}, 2^{53})$, take a random number R in the range $((k2^{55} - 1)/2P, (k+1)2^{55} - 2)/2P)$, and accept $e = 2PR + 1$ if e is prime. If the procedure completes, $\mu(e) = k$ trivially holds. [CS00] shows that the expected number of trials until P is prime is 64 (for the purpose of the reduction, a pool of 53-bit long primes can be precomputed), and for any fixed P the expected probability that e is prime is at least 1/128.

Before can sketch the proof of the following theorem, whose exact security claim is based on [SS00], we introduce the strong RSA assumption.

(t, ε)-strong RSA assumption. No algorithm running in time less than t can solve $x^r = y \bmod N$ for x and $r > 1$ with probability more than ε given random RSA modulus N, and random $y \in \mathbb{Z}_N^*$.

Theorem 3. *Fix $\ell = 160$. Let T_e is the time required to do 161-bit exponentiation. If the following holds:*

- $(t + T_e q_S \log q_S, \varepsilon_{\mathrm{RSA}})$*-RSA assumption;*
- $(t + T_e q_S \log q_S, \varepsilon_{\mathrm{SRSA}})$*-strong RSA assumption;*
- *A concrete pseudo-random number generator used for generating e and y' is (t, ε_G)-secure;*
- *H is (t, ε_H)-TCR;*

then TCR-CS is $(t, q_S, \varepsilon_{\mathrm{RSA}}(q_S+1)+\varepsilon_{\mathrm{SRSA}}\cdot 1.01+\varepsilon_H q_S 128+\varepsilon_G+q_S^2 2^{-145}+2^{-80})$*-secure.*

Proof [sketch]. For a detailed proof we refer the reader to [CS00, SS00]. Consider an adversary that makes q_S signing queries M_i, obtains signatures $\sigma_i = (e_i, y_i, y'_i)$, and then forges a signature $\sigma = (e, y, y')$ on $M \neq M_i$ for $1 \leq i \leq q_S$. Let $x'_i = (y'_i)^{e'} h^{-H_{\mu(e_i)}(M_i)}$ and $x' = (y')^{e'} h^{-H_{\mu(e)}(M')}$. We distinguish between three kinds of forgeries:

Type I. There is $1 \leq i \leq q_S$, such that $e = e_i$ and $x' = x'_i$.
Type II. There is $1 \leq i \leq q_S$, such that $e = e_i$ and $x' \neq x'_i$.
Type III. For all $1 \leq i \leq q_S$, $e \neq e_i$.

Proof from [CS00, SS00] applies without change for Type II and III forgeries (it suffices to check that nowhere in the proof does the choice of e depend on the hash of the message). To invoke the original proof for Type I forgery we have to bound the probability that $H_{\mu(e)}(M) = H_{\mu(e)}(M')$. We argue that such a forgery cannot happen with probability more than $\varepsilon_H q_S 128$, where ε_H is the security parameter of H. The proof is analogous to Theorem 2, with the only difference being the embedding process, described earlier in the section, that is used to map a random TCR key to a prime of $2PR + 1$ form. □

Finally, we observe that our modification applies to Fischlin's variant of the Cramer-Shoup scheme [Fis03], which is optimized for the size of the signature.

6 Discussion

In this section we address two points often raised in discussions of using TCR hashes as a building block of signature schemes: the problem of hash function's keylength, which is message size-dependent, and the random oracle assumption, which directly implies existence of collision-resistant functions.

Keylength of TCR hash. Bellare and Rogaway observed in [BR97] that adapting the iterative Merkle-Damgård [Mer90, Dam90] paradigm for TCR construction is not straightforward. Namely, even the second iteration of a TCR hash may be insecure (in contrast with a composition of collision-resistant hash functions, which is provably collision-resistant). They proposed interleaving applications of the compression function with XORing the chaining variable with independent masking keys, which increases the key length logarithmically with the size of the message. Their method was improved by Shoup [Sho00a], whose scheme was shown to be optimal among a concrete class of algorithms in [Mir01, Sar03]. For example, the key length required to hash a 1Gb message by going through the Shoup method applied to a keyed variant of the SHA-1 compression function is more than 4.8Kb. The proofs of optimality are exact and hence leave no hope of reducing the keylength if we are to stay within the existing paradigm.

We emphasize that the proofs of optimality only apply to a specific class of "masking-based" domain extenders. There are two potential ways to beat the lower bounds: design a dedicated TCR function, whose security is not degraded by chaining, or demonstrate a provably secure generic way of composing TCR hashes without key expansion. Both approaches are reasonable (see, for instance, [HPL04] which strengthens the definition of TCR to allow application of the Merkle-Damgård construction), and we expect that the interest in TCR functions rekindled by recent attacks on collision-resistant hash functions will spur further research in this area.

Random oracles and TCR functions. Pondering on the difference between collision-resistant hash functions and TCRs might appear rather pointless in the presence of the random oracle paradigm. Indeed, if we assume that a concrete hash function instantiates a random oracle, it is implicit that the function is collision-resistant and its domain can be trivially extended by going through the Merkle-Damgård construction, hence obliterating the need for TCRs. We claim that although this reduction is sound in theory, it may not be practical and may lead to bad design choices.

Hash functions must work for message lengths ranging from a few bytes to several hundred megabytes, which forces hash function designers to make certain trade-offs and defend against new classes of attacks. Designing an "oracle-like" hash function that accepts long inputs is inherently more challenging than designing a short-input function (for theoretical analysis of some of the difficulties see [KS05, CDMP05], for practical attacks that span several blocks see [WY05, BCJ$^+$05, WYY05b]). It suffices to point out that the latest generation of hash functions, such as SHA-256,512 or Whirlpool, works at a fraction of the speed of MD5 and is much slower than AES [NES03, NM02].

Constructing an "oracle"—a function that thoroughly but slowly hashes one block (0.5–1Kb long) is conceivable, but it would be inadequate as a general-purpose hash function. It is plausible that the trend towards slower hash functions can be reversed if, instead of designing one-size-fits-all collision-resistant hash functions, we settle for fast TCR functions able to handle long messages and relatively slow "oracles" for fixed-length inputs.

In support of our view we cite the performance characteristics of signature schemes and hash functions from the NESSIE report [NES03, Table 50]. For three signature schemes (Cramer-Shoup, ECDSA, and RSA-PSS) the speed of the sign and verify operations ranged from 1.6M (RSA-PSS verify) to 62M (RSA-PSS sign) CPU cycles on Pentium IV. For comparison, one-block SHA-256 evaluation, which takes about 1.5K CPU cycles [NM02], is faster by approximately three orders of magnitude. It means that the hash function (applied to one block!)

can be slowed down by two orders of magnitude without the schemes' overall performance taking notice.

Finally, we note that the "oracle" functions may find applications in practical solutions to the problem raised in the beginning of this section, namely the keylength of TCR functions. Shoup's masking-based solution to domain expansion requires a long key, which can be derived from a shorter key using an "oracle." The resulting scheme would be provably secure in the random oracle world and enjoy efficiency of a cheap TCR construction.

7 Conclusions

For any hash-and-sign signature scheme a collision-finding attack on the underlying hash function is devastating. Recent attacks on MD5 and SHA-1 [WY05, WYY05a] suggest that designing efficient collision-resistant hash functions is harder than it has been commonly thought.

TCR hashes provide a good alternative to collision-resistant hash functions in the context of digital signatures. Traditionally, replacing collision-resistant hashes with TCRs, which are by definition *keyed* hash functions, resulted in an increase in the signature size, which has to additionally accommodate the hash function's key. We argue that for specific signature schemes the key can be derived from the already present part of the signature. For the Cramer-Shoup signature scheme we prove in the standard model our variant of the scheme, which provides a shorter signature while offering the same security.

Security of signature schemes provable in the random oracle model relies on the assumption that some concrete hash functions are real-world implementations of a certain ideal functionality. We revisit two popular signature schemes, DSA and PSS-RSA, and propose their TCR-based variants, whose proofs of security, while still dependent on random oracles, only require short-input ones. We argue that a short-input oracle might be easier to construct, since it can afford to be much slower than a conventional hash function.

References

[BCJ$^+$05] Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, and William Jalby. Collisions of SHA-0 and reduced SHA-1. In Cramer [Cra05], pages 36–57.

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *Advances in Cryptology—CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1996.

[BM82] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *23rd Annual Symposium on Foundations of Computer Science*, pages 112–117, Chicago, Illinois, 3–5 November 1982. IEEE.

[BP97] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *Advances in Cryptology—EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. Springer, 1997.

[BPVY00] Ernest F. Brickell, David Pointcheval, Serge Vaudenay, and Moti Yung. Design validations for discrete logarithm based signature schemes. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography—PKC 2000*, volume 1751 of *Lecture Notes in Computer Science*, pages 276–292. Springer, 2000.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures—how to sign with RSA and Rabin. In Ueli M. Maurer, editor, *Advances in Cryptology—EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer, 1996.

[BR97] Mihir Bellare and Phillip Rogaway. Collision-resistant hashing: Towards making UOWHFs practical. In Burton S. Kaliski Jr., editor, *Advances in Cryptology—CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 470–484. Springer, 1997.

[Bra90] Gilles Brassard, editor. *Advances in Cryptology—CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20–24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*. Springer, 1990.

[Bro05] Daniel R. L. Brown. Generic groups, collision resistance, and ECDSA. *Designs, Codes and Cryptography*, 35(1):119–152, 2005.

[CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function. In Shoup [Sho05], pages 430–448.

[Cor02] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In Knudsen [Knu02], pages 272–287.

[Cra05] Ronald Cramer, editor. *Advances in Cryptology—EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22–26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*. Springer, 2005.

[CS00] Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. *ACM Trans. on Information and System Security (TISSEC)*, 3(3):161–185, 2000.

[Dam90] Ivan Damgård. A design principle for hash functions. In Brassard [Bra90], pages 416–427.

[DOP05] Yevgeniy Dodis, Roberto Oliveira, and Krzysztof Pietrzak. On the generic insecurity of the full domain hash. In Shoup [Sho05], pages 449–466.

[Fis03] Marc Fischlin. The Cramer-Shoup Strong-RSA signature scheme revisited. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 116–129. Springer, 2003.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology—CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1987.

[GHR99] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In Jacques Stern, editor, *Advances in Cryptology—EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 123–139. Springer, 1999.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17:281–308, 1988.

[HK05a] Shai Halevi and Hugo Krawczyk. Strengthening digital signatures via randomized hashing. Internet-Draft, Crypto Forum Research Group, May 2005.

[HK05b] Shai Halevi and Hugo Krawczyk. Strengthening digital signatures via randomized hashing. Talk at Cryptographic Hash Workshop (NIST), Oct 31–Nov 1. 2005.

[HPL04] Deukjo Hong, Bart Preneel, and Sangjin Lee. Higher order universal one-way hash functions. In Pil Joong Lee, editor, *Advances in Cryptology—ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 201–213. Springer, 2004.

[Knu02] Lars R. Knudsen, editor. *Advances in Cryptology—EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28–May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*. Springer, 2002.

[KS05] John Kelsey and Bruce Schneier. Second preimages on n-bit hash functions for much less than 2^n work. In Cramer [Cra05], pages 474–490.

[Mer90] Ralph C. Merkle. One way hash functions and DES. In Brassard [Bra90], pages 428–446.

[Mir01] Ilya Mironov. Hash functions: From Merkle-Damgård to Shoup. In Birgit Pfitzmann, editor, *Advances in Cryptology—EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 166–181. Springer, 2001.

[NES03] NESSIE Consortium. Performance of optimized implementations of the NESSIE primitives, version 2.0. Deliverable report D21, February 2003. NES/DOC/TEC/WP6/D21/2.

[NIS95] NIST. Secure hash standard. FIPS PUB 180-1, National Institute of Standards and Technology, April 1995.

[NM02] Junko Nakajima and Mitsuru Matsui. Performance analysis and parallel implementation of dedicated hash functions. In Knudsen [Knu02], pages 165–180.

[NS02] Phong Q. Nguyen and Igor E. Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *J. Cryptology*, 15(3):151–176, 2002.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 33–43, 15–17 May 1989.

[Pre00] Bart Preneel, editor. *Advances in Cryptology—EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14–18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*. Springer, 2000.

[PV05] Pascal Paillier and Damien Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In Bimal Roy, editor, *Advances in Cryptology—ASIACRYPT 2005*, Lecture Notes in Computer Science, pages 1–20. Springer, 2005.

[Riv91] Ronald L. Rivest. The MD4 message digest algorithm. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology—CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 303–311. Springer, 1991.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 387–394, 14–16 May 1990.

[Sar03] Palash Sarkar. Masking based domain extenders for UOWHFs: Bounds and constructions. Cryptology ePrint Archive, Report 2003/225, 2003. `http://eprint.iacr.org/`.

[Sho00a] Victor Shoup. A composition theorem for universal one-way hash functions. In Preneel [Pre00], pages 445–452.

[Sho00b] Victor Shoup. Using hash functions as a hedge against chosen ciphertext attack. In Preneel [Pre00], pages 275–288.

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/`.

[Sho05] Victor Shoup, editor. *Advances in Cryptology—CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14–18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*. Springer, 2005.

[Sim98] Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In Kaisa Nyberg, editor, *Advances in Cryptology—EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 334–345. Springer, 1998.

[SS00] Thomas Schweinberger and Victor Shoup. ACE: The advanced cryptographic engine. Manuscript, 2000. `http://shoup.net/papers/ace.pdf`.

[WY05] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Cramer [Cra05], pages 19–35.

[WYY05a] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Shoup [Sho05], pages 17–36.

[WYY05b] Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient collision search attacks on SHA-0. In Shoup [Sho05], pages 1–16.