On Hash Function Firewalls in Signature Schemes

Burton S. Kaliski Jr.

RSA Laboratories
20 Crosby Drive, Bedford, MA 01730, USA
bkaliski@rsasecurity.com

Abstract. The security of many signature schemes depends on the verifier's assurance that the same hash function is applied during signature verification as during signature generation. Several schemes provide this assurance by appending a hash function identifier to the hash value. We show that such "hash function firewalls" do not necessarily prevent an opponent from forging signatures with a weak hash function and we give "weak hash function" attacks on several signature schemes that employ such firewalls. We also describe a new signature forgery attack on PKCS #1 v1.5 signatures, possible even with a strong hash function, based on choosing a new (and suspicious-looking) hash function identifier as part of the attack.

1 Introduction

Nearly all digital signature schemes in practice combine a hash function with other cryptographic operations. In a typical scheme, a signer applies the hash function to the message, then applies a signature primitive to the signer's private key and the hash function output to obtain a signature. The verifier likewise computes a hash value, then applies a verification primitive to the signer's public key, the hash value, and the signature to determine whether the signature is valid. In many schemes, some formatting is applied to the hash value prior to the signature primitive, and in some schemes part or all of the message is included in the input to the signature primitive and can be recovered from the signature.

The security of a signature scheme clearly depends on the security of the hash function. Just as importantly, the security depends on the verifier's assurance that the correct hash function is applied during signature verification. Otherwise, an opponent may be able to take advantage of hash function weaknesses to obtain a forged signature. Suppose that the signer has selected a strong hash function Hash and has signed messages $M_1, \ldots, M_k$, producing signatures $\sigma_1, \ldots, \sigma_k$. Suppose further that the opponent knows a weak hash function $\mathsf{WeakHash}$ and can cause the verifier to apply the weak hash function rather than the strong one. If the opponent can find another message M' such that $\mathsf{WeakHash}(M') = \mathsf{Hash}(M_i)$ for some message M_i then the opponent can present M' and σ_i to the verifier, specifying that the weak hash function is to be applied, and they will be accepted.

The verifier's willingness to accept hash functions other than the one the signer has selected is often in the interest of interoperability. A typical verifier may interact with many signers and thus may need to support a large set of hash

B. Preneel (Ed.): CT-RSA 2002, LNCS 2271, pp. 1–16, 2002.

functions, even if each individual signer only supports a few hash functions. The verifier may not know which hash functions are acceptable for a given signer, nor whether a weakness has been found in a hash function. (By "weakness," here and elsewhere, we mean the ability for an opponent to invert the hash function efficiently for a significant fraction of hash values.)

Clearly, it is desirable to limit the set of hash functions that a verifier accepts for a given signer. One approach is to permit only a small set of well trusted hash functions in a given domain. The Digital Signature Standard [17], for instance, allows only the SHA-1 hash function [16]; ANSI X9.31 [1] and ANSI X9.62 [2] likewise allow only "ANSI-approved" hash functions. Another approach is to convey the set of permitted hash functions as an attribute of the signer's public key, for instance in a public-key certificate or as part of policy for a certificate domain. In this way, signers can select different hash functions but are not affected by one another's choices.

Each of the methods just mentioned has some drawbacks. Limiting the set of hash functions precludes new hash functions that may be faster or otherwise more attractive; conveying the set as an attribute of a public key may rely on more complex certificate management infrastructure than is conveniently available. Note that identifying the hash function in the message itself is not enough; it is likely as easy for an opponent to control the identifier as any other part of a message when forging a signature.

As a result of these concerns, the hash function is often identified within the formatting that is applied to the hash value. This identifier might be an index to a registry of hash functions, such as IS 10118-3:1998 [10], or it might be based on an OSI object identifier (see [23]), which can be assigned by any organization. The signature primitive is then relied upon to bind the hash function identifier securely to the hash value. In this way the hash function identifier serves as a *firewall* between different hash functions.

A hash function firewall prevents an opponent from causing a verifier to verify a signature with a different hash function than the signer intended. But this protection is only for existing signatures; a separate question is whether an opponent can obtain new signatures that exploit a weak hash function. In this paper, we answer that question, offering the following new contributions:

1. We present new *weak hash function attacks* against the signature schemes in IS 9796-2:1997 [9], IS 14888-2:1999 [11], and IS 9796-3:2000 [12]. Each scheme includes an optional hash function firewall and claims that the firewall offers some level of protection against attack. We show that such claims are incorrect, and that if any registered hash function were found to be weak and a verifier supports it, it may be possible to forge a signature. Significantly, our attacks are possible *without the signer's participation.*
2. We show that the signature schemes in ANSI X9.31 [1] and PKCS #1 v1.5 [23], which also include hash function firewalls, appear to protect against weak hash function attacks. However, in the case of PKCS #1 v1.5 signatures, we show that if the opponent is allowed to register a *new* hash function identifier for a given hash function, it is possible to forge signatures even if the hash function is strong. Moreover, the forged signatures are valid for a wide range of public keys, not just for one signer.