On Randomizing Hash Functions to Strengthen the Security of Digital Signatures*

Praveen Gauravaram** and Lars R. Knudsen

Department of Mathematics
Technical University of Denmark
Matematiktorvet, Building S303
DK-2800 Kgs. Lyngby
Denmark
P.Gauravaram@mat.dtu.dk, Lars.R.Knudsen@mat.dtu.dk

Abstract. Halevi and Krawczyk proposed a message randomization algorithm called RMX as a front-end tool to the hash-then-sign digital signature schemes such as DSS and RSA in order to free their reliance on the collision resistance property of the hash functions. They have shown that to forge a RMX-hash-then-sign signature scheme, one has to solve a cryptanalytical task which is related to finding second preimages for the hash function. In this article, we will show how to use Dean's method of finding expandable messages for finding a second preimage in the Merkle-Damgård hash function to existentially forge a signature scheme based on a t-bit RMX-hash function which uses the Davies-Meyer compression functions (e.g., MD4, MD5, SHA family) in $2^{t/2}$ chosen messages plus $2^{t/2+1}$ off-line operations of the compression function and similar amount of memory. This forgery attack also works on the signature schemes that use Davies-Meyer schemes and a variant of RMX published by NIST in its Draft Special Publication (SP) 800-106. We discuss some important applications of our attack.

Keywords: Digital signatures, Hash functions, Davies-Meyer, RMX.

1 Introduction

The collision attacks on the MD5 [36] and SHA-1 [30] hash functions in the recent years are one of the most vital contributions in the field of cryptology [40, 41]. Since then, there has been a reasonable amount of research showing how seriously these attacks (in particular on MD5) could undermine the security of the digital signatures [24, 6, 17, 39, 38] in which these hash functions are deployed.

For a long time, it has been suggested to randomize the messages using a fresh random value, also called salt, before they are hashed and signed, in order to free

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

* The work in this paper has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

** Author and this research project has been supported by the Danish Research Council for Technology and Production Sciences grant number 274-08-0052.

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 88–105, 2009.
© Springer-Verlag Berlin Heidelberg 2009

the security of the digital signatures from depending on the collision resistance of the hash functions [14,13,1]. Message randomization before hashing would make an attacker to predict the random value in order to make use of collisions in a hash function to forge a digital signature. In consequence of that, the security of a digital signature would be forced to depend on a property weaker than the collision resistance of the hash function.

At Crypto 2006, Halevi and Krawczyk [19] proposed and analysed two simple message randomization transforms as the front-end tools for any hash-then-sign signature scheme which uses Merkle-Damgård hash functions [26,9]. They have shown that these message randomization techniques would base the security of the hash-then-sign signature schemes on the second preimage resistance of the hash function. They have noted that long message second preimage attack of Kelsey and Schneier [23] can be mounted on the hashes of the randomized messages to find second preimages to existentially forge the signatures of these hashes. Merkle-Damgård hash functions with these front-end tools were also considered as new modes of operation [19]. One of these transforms, called RMX, has the practical viability with the signature schemes such as RSA [37] and DSA [29] and was fully specified in [19, Appendix D], [20, 21]. We call a hash-then-sign signature scheme which uses RMX as the front-end tool, a RMX-hash-then-sign signature scheme.

A signer computes the signature of a message m using a RMX-hash-then-sign signature scheme as follows: He chooses a random value denoted r, and randomizes m by passing the pair (r, m) as input to the RMX transform. The randomized message is given by $M = \mathrm{RMX}(r, m)$. The signer processes the message M using a t-bit hash function H and obtains the t-bit hash value $H(M)$. The signer signs the hash value $H(M)$ using a signature algorithm, denoted SIG, and obtains the signature s. The signer sends the triplet (m, r, s) to the verifier who computes $M = \mathrm{RMX}(r, m)$ and provides the pair (M, s) to the verification procedure to verify s.

The RMX transform requires no change to the signatures algorithms such as RSA and DSA. The implementation of RSA [37] based on RMX-SHA-1 was discussed in [21]. In 2007, NIST has published a variant of RMX as a draft special publication (SP) 800-106 [10] which was superseded in 2008 by a second draft SP 800-106 [11]. In addition, NIST intends that the candidate hash functions in the SHA-3 hash function competition support randomized hashing [31].

1.1 Related Work

Dang and Perlner [12] have shown a generic chosen message forgery attack on the signature schemes based on t-bit RMX-hashes in $2^{t/2}$ chosen messages and $2^{t/2}$ operations of the hash function and similar memory. This attack produces a collision of form $H(\mathrm{RMX}(r, m)) = H(\mathrm{RMX}(r^*, n))$ which implies $\mathsf{SIG}(m) = \mathsf{SIG}(n)$ where $(r, m) \neq (r^*, n)$, m is one of the chosen messages and n is the forgery message of m.

1.2 Our Results

In this article, we first note that the attack of Dang and Perlner [12] does not produce a verifiable forgery if the signer uses the same random value for both RMX hashing and signing such as in DSA [29, 32], ECDSA [3, 32] and RSA-PSS [37]. The re-use of the random value in the signature schemes for the randomized hashing to save the communication bandwidth was addressed in [19, 33, 27, 11, 12]. The attack of [12] also does not work on the signature schemes based on the other randomized hash function analysed by Halevi and Krawczyk [19] wherein both the salt and randomized hash value are signed to produce a signature.

We then show an existential forgery attack under a generic chosen message attack [18] on the RMX-hash-then-sign signature schemes when the compression functions of the hash functions have fixed points. Our attack produces a valid forgery in the above applications wherein the forgery attack of Dang and Perlner does not succeed. Our attack uses Dean's trick of finding fixed-point expandable messages [15, 23] for finding second preimages in the hash functions that use fixed point compression functions. Many popular hash functions, that include, MD4 [35], MD5 [36], SHA family [30] and Tiger [2] have compression functions that use Davies-Meyer construction [22, 34] for which fixed points can be easily found [28]. In an existential forgery attack, the attacker asks the signer for the signatures on a set of messages of his choice and is then able to produce a valid signature on a message which was never signed by the signer. Our forgery attack requires $2^{t/2}$ equal length chosen messages, $2^{t/2+1}$ off-line operations of the compression function and a probability of 2^{-24} to hit the correct bits used to pad the message by the RMX transform. The attack requires about $2^{t/2}$ memory. With this computational work, we can establish a collision of the form $H(\mathrm{RMX}(r, m)) = H(\mathrm{RMX}(r, m\|n))$ where m is one of the chosen messages and $n \neq m$ is a randomized message block which gives a fixed point. This implies $\mathsf{SIG}(m) = \mathsf{SIG}(m\|n)$ and we show the message $m\|n$ as the forgery of m. Our attack also works on the signature schemes that use a variant of RMX published by NIST in its SP 800-106 [11] and in its earlier version [10].

1.3 Impact of Our Results

Our forgery attack on the RMX-hash-then-sign signature schemes is totally impractical for the reasonable hash value sizes of 256 bits. Moreover, our attack can not be parallelizable as it requires a real signer to sign a huge set of messages. Our analysis is in no contradiction to that of Halevi and Krawczyk [19]. Moreover, it complements their analysis by showing that RMX-hashes achieve an essential security improvement with respect to off-line birthday attacks in forcing these attacks to work on-line (e.g. requiring $2^{t/2}$ messages signed by the legitimate signer and similar amount of memory). Our analysis demonstrates that the security of RMX-hash-then-sign signature schemes based on the t-bit ideal fixed-point compression functions is equivalent to that of the t-bit standard keyed hash function HMAC [4] based on a t-bit ideal compression function. The attack of [12] has a similar impact, though its application is limited.

1.4 Guide to the Paper

In Section 2, we provide the notation and the background information necessary to understand the paper. In Section 3, we describe randomized hash functions of [19] and outline the RMX specification and its variant published by NIST [11]. In Section 4, we discuss the forgery attack of [12] on the RMX-hash-then-sign schemes and its limitations. In Section 5, we show how to apply Dean's fixed point expandable messages to forge hash-then-sign signatures. In Section 6, we describe our forgery attack on the signature schemes based on the RMX hash mode and its variants. In Section 7, we conclude the paper with some open questions.

2 Preliminaries

In this section, we define some notation and review some fundamentals of hash functions and digital signatures that will be used throughout the paper.

2.1 Notation

The symbol $\|$ represents the concatenation operation. The notation e^{α} represents the concatenation of e bit α times where e is either 0 or 1. For example, $1^4 = 1\|1\|1\|1$. If a is a positive integer, we represent by $a^{[\alpha]}$, the first (from left to right) α bits of a. For example, if $a = 1011011001$ then $a^{[4]} = 1011$. Similarly, if a^b is a positive integer, we represent by $(a^b)^{[\alpha]}$, the first α bits of a^b.

2.2 Merkle-Damgård Hash Functions

Let $H : \{0,1\}^* \rightarrow \{0,1\}^t$ be a Merkle-Damgård hash function based on the compression function $h : \{0,1\}^b \times \{0,1\}^t \rightarrow \{0,1\}^t$. An upper bound in bits (say 2^l) on the length of the message to be hashed is often specified for H. The message m to be processed using H is split into blocks $m_1, m_2, \ldots, m_{L-1}$ and m_L. Let $|m_i| = b$ for $i = 1$ to $L-1$ and q be the number of the message bits in the last block m_L where $q < b$. If $q \leq b - l - 1$ then the message m is padded with $1\|0^{b-l-q-1}\||m|$ where $|m|$ is the l-bit binary representation of the length of the message m. If $q > b - l - 1$ then the message m is padded with $1\|0^{b-q-1}$ and a separate b-bit block $0^{b-l}\|\ |m|$ is concatenated to the padded message $m\|1\|0^{b-q-1}$. Every message block m_i, for $i = 1 \ldots L$, is processed using h as defined by $H_i = h(H_{i-1}, m_i)$ where H_0 is the initial value (IV) of H, H_i is the intermediate hash value of H at iteration i of h and H_L is the hash value of H. We denote by H^{H_0}, a hash function with H_0 as the IV.

Some properties of an ideal hash function H^{H_0}.

1. Collision resistance (CR): It should take about $2^{t/2}$ operations of H^{H_0} to find two messages m and n such that $m \neq n$ and $H^{H_0}(m) = H^{H_0}(n)$.
2. Second preimage resistance (SPR): For a challenged target message m, it should take about 2^t operations of H^{H_0} to find another message n such that $n \neq m$ and $H^{H_0}(m) = H^{H_0}(n)$. However, for a target message of 2^d blocks, second preimages for H^{H_0} can be found in about 2^{t-d} operations of h [23].

Some properties of an ideal compression function h.

1. CR: It should take about $2^{t/2}$ operations of h to find two different pairs (H_{i-1}, m_i) and (H^*_{i-1}, n_i) such that $h(H_{i-1}, m_i) = h(H^*_{i-1}, n_i)$.
2. SPR: For a challenged pair (H_{i-1}, m_i), it should take about 2^t operations of h to find a pair (H^*_{i-1}, n_i) such that $(H_{i-1}, m_i) \neq (H^*_{i-1}, n_i)$ and $h(H_{i-1}, m_i) = h(H^*_{i-1}, n_i)$. This property is also called random-SPR (r-SPR) [19].

2.3 Compression Functions with Fixed Points

A fixed point for a compression function h is a pair (H_{i-1}, m_i) such that $h(H_{i-1}, m_i) = H_{i-1}$. Let h be the Davies-Meyer compression function which is defined by $h(H_{i-1}, m_i) = E_{m_i}(H_{i-1}) \oplus H_{i-1} = H_i$ where m_i is the message block which is used as a key to the block cipher E and the input state H_{i-1} is the plaintext to E. A fixed point for h can be easily found by evaluating the expression $E^{-1}_{m_i}(0)$ for some message block m_i. Davies-Meyer feed-forward can also use addition mod 2^t and fixed points can still be found for this case. This technique to find fixed points for the Davies-Meyer construction has been described in [28], [23, Appendix A.1].

2.4 Existential Forgery Attack on the Signature Schemes

An existential forgery of a signature scheme SIG under a generic chosen message attack [18] (also called weak chosen message attack [8]) is performed as follows:

1. The attacker sends to the challenger (signer) a list of q messages $m^1, \ldots, m^q$.
2. The challenger generates a public and private key pair (Pk, Sk) using a key generation algorithm. The challenger generates the signatures s_i on the messages m^i computed using his private key Sk and the signature algorithm SIG for $i = 1, \ldots, q$. The challenger sends to the attacker the public key Pk and the signatures s_i.
3. The attacker forges the signature scheme SIG by outputting a pair (m, s) if:
   (a) $(m, s) \notin \{(m^1, s_1), \ldots, (m^q, s_q)\}$; and
   (b) The triplet (Pk, m, s) produces a valid verification.

Let Adv be the probability that the adversary wins the above game, taken over the coin tosses made by him and the challenger. The adversary is said to (t, q, ϵ)-existentially forge the signature scheme SIG if he runs in time at most t, makes at most q queries and $\mathsf{Adv} \geq \epsilon$.

3 Randomized Hashing

A family of hash functions $\{H_r\}_{r \in R}$ for some set R is target collision resistant (TCR) [5, 19] if no efficient attacker after choosing a message m and receiving the salt $r \in_R R$ can find a second preimage n such that $m \neq n$ and $H_r(m) = H_r(n)$

except with insignificant probability. The usage of the family $\{H_r\}_{r\in R}$ for the digital signatures necessitates the users to sign the salt r along with the hash of the message. However, hash-then-sign signature schemes such as DSA [29] and RSA [37] do not support signing the salt in addition to $H_r(m)$. In order to free such signature schemes from signing the salt, Halevi and Krawczyk introduced the notion of enhanced TCR (eTCR) hash function family [19]. The hash function family $\{\tilde{H}_r\}_{r\in R}$ is eTCR if there exists no efficient attacker who after committing to a message m and receiving the salt r, can find a pair $(r^*, n) \neq (r, m)$ such that $\tilde{H}_r(m) = \tilde{H}_{r^*}(n)$ except with insignificant probability.

Halevi and Krawczyk [19] presented two randomized hash function modes for H. The first t-bit scheme, denoted H_r, XORs every block m_i of the message m with a b-bit random value r as shown below:

$$H_r^{H_0}(m) = H_r^{H_0}(m_1\| \ldots \|m_L) \stackrel{\text{def}}{=} H^{H_0}(m_1 \oplus r\|m_2 \oplus r\| \ldots \|m_L \oplus r).$$

The second t-bit scheme, denoted $\tilde{H}_r$, prepends r to $m_i \oplus r$ for $i = 1 \ldots, L$ as shown below:

$$\tilde{H}_r^{H_0}(m) = \tilde{H}_r^{H_0}(m_1\| \ldots \|m_L) \stackrel{\text{def}}{=} H^{H_0}(r\|m_1 \oplus r\|m_2 \oplus r\| \ldots \|m_L \oplus r).$$

The functions $\tilde{H}_r$ and H_r are eTCR and TCR respectively if the compression function h is either chosen-SPR (c-SPR) or Evaluated-SPR (e-SPR) [19]. These properties for the compression function h are defined below:

1. c-SPR: For a given message block m_i, find $(H_{i-1}, H^*_{i-1}, n_i)$ such that $h(H_{i-1}, m_i) = h(H^*_{i-1}, n_i)$.
2. e-SPR: Choose $u \geq 1$ values $\Delta_1, \ldots, \Delta_u$ each of length b bits. Receive a random value $r \in \{0,1\}^b$ and then define $m_i = r \oplus \Delta_u$ and $H_{i-1} = H^{H_0}(r \oplus \Delta_1\| \ldots \|r \oplus \Delta_{u-1})$. Find (H^*_{i-1}, n_i) such that $h(H_{i-1}, m_i) = h(H^*_{i-1}, n_i)$.

A generic birthday attack can be mounted on the c-SPR property of h and it does not work on the r-SPR and e-SPR properties of h [19]. The eTCR construction $\tilde{H}_r$ was proposed as the preferred hash function mode for use in digital signatures as it does not require explicit signing of the salt r and allows for better implementation flexibility. A concrete specification of $\tilde{H}_r$ called RMX and its usage with the digital signatures and its implementation details were discussed in [19, Appendix D] [20] and [21] respectively. RMX was also considered as a message randomization transform.

3.1 RMX Specification

The RMX scheme randomizes an input message m of at most $2^l - b$ bits using a random value r of length between 128 and b bits to an output message M. The RMX algorithm is defined below following [19, Appendix D], [20, 21]:

1. Three random values r_0, r_1 and r_2 are computed from r as follows:
   (a) $r_0 = r\|0^{b-|r|}$ such that $|r_0| = b$ bits.

   (b) $r_1 = \underbrace{r\|r\|\ldots\|r}_{b \text{ bits}}$ such that $|r_1| = b$ and the last repetition of r is truncated if needed.
   (c) $r_2 = r_1^{[b-l-8]}$ (The first $b-l-8$ bits of r_1).
2. Split the input message m into $L-1$ b-bit blocks $m_1, m_2 \ldots, m_{L-1}$ and a last block m_L of length b' where $1 \leq b' \leq b$.
3. Set $M_0 = r_0$.
4. For $i = 1$ to $L-1$:
   (a) $M_i = m_i \oplus r_1$.
5. Let $lpad$ (meaning last block pad) be a 16-bit string, representing the bit length b' of m_L in the big-endian notation. If $lpad_0$ and $lpad_1$ are the first and second bytes of $lpad$, respectively, and each of these bytes represents a number between 0 and 255, then $b' = 256 \times lpad_1 + lpad_0$.
   (a) If $b' \leq b-l-24$ then set $M_L^* = m_L\|0^k\|lpad$ where $k = b-b'-16-l$. Set $M_L = M_L^* \oplus r_2$.
   (b) If $b' > b-l-24$ then set $M_L^* = m_L\|0^{b-b'}$ and $M_{L+1}^* = 0^{b-l-24}\|lpad$. Set $M_L = M_L^* \oplus r_1$ and $M_{L+1} = M_{L+1}^* \oplus r_2$.
6. Output the randomized message $M = \text{RMX}(r, m) = M_0\|\ldots\|M_L$ in the case of (5a) and $M = \text{RMX}(r, m) = M_0\|\ldots\|M_L\|M_{L+1}$ in the case of (5b).

Remark 1. We note that when $b' \leq b-l-24$, the padding rule designed for RMX with $k = b-b'-16-l$ requires a separate block to accommodate the padding and length encoding (at least $l+1$) bits of the message M required by the hash function H^{H_0} as illustrated in Appendix A. In addition, when $b' \leq b-l-24$, with $k = b-b'-16-l$, $|M_L^*| = b-l$ and hence $|r_2| = b-l$ bits which means $r_2 = r_1^{[b-l]}$. For example, let $|m_L| = b' = b-l-24$ bits. Then $k = b-b'-l-16 = b-(b-l-24)-l-16 = 8$ bits. Now $M_L^* = m_L\|0^8\|lpad$ and $|M_L^*| = b-l-24+8+16 = b-l$ bits. Hence, r_2 should also be $b-l$ bits. Hence, in this paper, we set $r_2 = r_1^{[b-l]}$ for $b' \leq b-l-24$ bits.

If we set $k = b-b'-24-l$ for $b' \leq b-l-24$ bits then the padding and length encoding bits required by H^{H_0} can be accommodated in the last block of M as illustrated in Appendix A. When $k = b-b'-24-l$, with $b' \leq b-l-24$, $|M_L^*| = b-l-8$ bits and hence $|r_2| = b-l-8$ bits which means $r_2 = r_1^{[b-l-8]}$ which is the same as in RMX specification. For example, let $|m_L| = b' = b-l-24$ bits. Then $k = b-b'-l-24 = b-(b-l-24)-l-24 = 0$ bits. Now $M_L^* = m_L\|0^0\|lpad$ and $|M_L^*| = b-l-24+0+16 = b-l-8$ bits and hence $|r_2| = b-l-8$ bits and we can set $r_2 = r_1^{[b-l-8]}$.

A variant of RMX. NIST has published a variant of RMX in its SP 800-106 [11] replacing its previous draft SP 800-106 [10]. We call the latest variant of RMX in SP 800-106 [11] by RMX_{SP} whose specification is placed in Appendix B.

Remark 2. We note that RMX and RMX_{SP} differ in the padding rule defined for the messages. In addition, in RMX, the prepended random value r is extended to a block of b bits by padding it with 0 bits, whereas, in RMX_{SP}, it is directly concatenated with the XOR of the message blocks and random value.

4 Generic Forgery Attack on the RMX-Hash-Then-Sign Signature Schemes

Dang and Perlner [12] proposed an on-line birthday forgery attack on the signature schemes based on t-bit RMX-hashes in $2^{t/2}$ chosen messages, $2^{t/2}$ off-line hash function operations and a similar amount of memory as outlined below:

- *On-line phase:*
  1. Query the signer for the signatures of $2^{t/2}$ chosen messages m^i where $i = 1, \ldots, 2^{t/2}$. Store every m^i in a Table L_1.
  2. The signer chooses a fresh random value r_i to compute the signature s_i of every message m^i using SIG. The signer first computes $\mathrm{RMX}(m^i)$ and then computes $\mathsf{SIG}(H^{H_0}(\mathrm{RMX}(m^i))) = s_i$. The signer returns the pair (r_i, s_i) where $i = 1, \ldots, 2^{t/2}$.
- *Off-line phase:*
  1. For $i = 1, \ldots, 2^{t/2}$, using r_i, compute the hash values $H^{H_0}(\mathrm{RMX}(r_i, m^i))$ and store them together with (r_i, s_i) in L_1.
  2. Choose random pairs (r_j, m_j) and compute the hash values $H^{H_0}(\mathrm{RMX}(r_j, m_j))$ where j is in increments of 1. While computing a hash value, check whether it collides with any of the hash values in the Table L_1. After about $j = 2^{t/2}$ attempts, with a good probability, we can find one collision. Let that random pair be (r_y, m_y). That is, we can find (r_x, m^x, H_x) from L_1 where $(r_x, m^x) \neq (r_y, m_y)$ and $H_x = H^{H_0}(\mathrm{RMX}(r_y, m_y)) = H^{H_0}(\mathrm{RMX}(r_x, m^x))$ where $x, y \in \{1, \ldots, 2^{t/2}\}$. Hence, $\mathsf{SIG}(m^x) = \mathsf{SIG}(m_y)$.
  3. Output message m_y as the forgery of m^x.

4.1 Limitations of the Forgery Attack

We note that the above attack does not produce a valid forgery in the following signature applications. These applications were noted in [19, 12].

- The random component that already exists in the signature schemes such as RSA-PSS, DSA and ECDSA, can also be used for randomized hashing (e.g., RMX hash mode) to save the bandwidth. The above attack does not succeed on such signature schemes during the forgery verification as the random value used by the signer for RMX hashing and signing matches the arbitrary value chosen by the attacker with a negligible probability.
- When the signature schemes based on TCR hashing H_r are used to sign a message m, both r and $H_r(m)$ have to be signed. For a valid forgery on such signatures, the attacker should use the same random value as the signer; hence this attack does not succeed.

5 Application of Dean's Fixed Point Expandable Messages to Forge Hash-Then-Sign Signature Schemes

Dean [15,23] has shown that if it is easy to find fixed points for the compression function then a fixed point expandable message, a multicollision using different length messages, can be constructed for the hash function. Using Dean's trick, we show that, one out of $2^{t/2}$ signatures obtained on the equal length messages from a legitimate signer can be forged by finding a collision which looks like $H^{H_0}(m) = H^{H_0}(m\|n)$ where n is the fixed point message block. This implies $\mathsf{SIG}(m\|n) = \mathsf{SIG}(m)$ and we show the message $m\|n$ as the forgery of m in $2^{t/2+1}$ invocations of H^{H_0} and one chosen message query to the signer. We assume that h is the Davies-Meyer compression function. The attack is outlined below:

1. Consider $2^{t/2}$ equal-length messages m^i of length $(c \times b) - (l+1)$ bits where c is the number of b-bit blocks and $i = 1, \ldots, 2^{t/2}$. Compute the hash values H_i of m^i under H^{H_0} where each m^i is padded with a bit 1 followed by l bits that represent the binary format of the length $(c \times b) - (l+1)$ bits. Let these $l+1$ bits be *pad* bits. Store m^i and H_i in a table L_1.
2. For $j = 1, \ldots, 2^{t/2}$, compute $2^{t/2}$ fixed points for h such that $h(H_j^*, n^j) = H_j^*$ where the last $l+1$ bits of every block n^j are fixed with a padding bit 1 and l bits that represent the binary format of the length of the message $m^i\|pad\|(n^j)^{[b-(l+1)]}$ where $(n^j)^{[b-(l+1)]}$ represents the first $b-(l+1)$ bits of n^j. Let the last $l+1$ bits of n^j be *padf* bits where by *padf* we mean padding bits in the fixed point block. Store H_j^* and $(n^j)^{[b-(l+1)]}$ in a table L_2.
3. According to the birthday paradox, with a significant probability, we can find a hash value H_x from the list L_1 and a hash value H_y^* from the list L_2 such that $H^{H_0}(m^x\|pad) = H_x = H_y^* = h(H_y^*, n^y)$ for some $x \in \{1, \ldots, 2^{t/2}\}$ and $y \in \{1, \ldots, 2^{t/2}\}$. This implies $H^{H_0}(m^x\|pad\|n^y) = H^{H_0}(m^x\|pad) = H_x$. Let $m = m^x$ and $n = (n^y)^{[b-(l+1)]}$.
4. Ask the signer for the signature on the message m. The signer hashes the message $m\|pad$ using H^{H_0} and then signs the hash value $H^{H_0}(m\|pad)$ using the signature algorithm SIG to obtain the signature $s = \mathsf{SIG}(m)$.
5. Now, $H^{H_0}(m\|pad\|n\|padf) = H^{H_0}(m\|pad)$. Hence, $\mathsf{SIG}(H^{H_0}(m\|pad)) = \mathsf{SIG}(H^{H_0}(m\|pad\|n))$ which implies $\mathsf{SIG}(m) = \mathsf{SIG}(m\|pad\|n)$. Note that *padf* bits are the padded bits to the message $m\|pad\|n$ when it is hashed with H^{H_0}.
6. Output the message $m\|pad\|n$ as the forgery of the chosen message m.

Remark 3. Our attack technique subtly differs from Dean's trick [15, 23] as we exert control over the fixed point message blocks by integrating the padding and length encoding bits that represent the length of the forgery message in the last few bits of the fixed point block. Whereas in Dean's trick to find expandable messages, all bits in the fixed point block can be random. Hence, our trick would also work to find expandable messages for the hash functions when the message inputs are XORed with a random value.

6 Existential Forgery Attack on Some RMX-Hash-Then-Sign Signatures

Here we extend the technique presented in Section 5 to provide an existential forgery attack on the hash-then-sign signatures that use t-bit fixed point compression functions and RMX transform specified in Section 3.1 and Remark 1. Our forgery attack also works in the applications outlined in Section 4.1 wherein the generic forgery attack described in Section 4 does not succeed.

In this attack, we first determine the length of the message to be forged and also length of the message to be produced as a forgery. We then compute $2^{t/2}$ fixed point message blocks for the compression function by integrating padding and length encoding bits required for the forgery message into the fixed point blocks. We then ask the signer to sign $2^{t/2}$ equal length chosen messages and collect their signatures and random values used for signing. We use those $2^{t/2}$ random values and messages to compute $2^{t/2}$ RMX-hashes and find a collision with the $2^{t/2}$ pre-computed fixed point hash values. We will find one of the chosen messages (along with its random value and signature) and one fixed point message block whose respective hashes collide. We then XOR the fixed point block and the random value to obtain a new block and concatenate it as a suffix block to the chosen message. Finally, we produce this concatenated message as the forgery of the chosen message. The attack involves some subtleties due to the fact that the RMX transform has a padding layer and hence, the attack has an additional negligible complexity of 2^{24} which adds to the complexity of $2^{t/2}$ chosen messages and $2^{t/2+1}$ operations of the compression function.

The pseudocode for the existential forgery attack on the signature scheme SIG which uses a hash function H^{H_0} based on the Davies-Meyer compression function h and RMX as the message randomization algorithm is given below:

- *Pre-computation phase:*
  1. Determine the length of the message to be forged. Let it be a message m of $2b-l-24$ bits. Let m^* be the forgery of m to be produced whose length can be pre-determined using $|m|$ as given by $|m^*| = |m|+l+24+b+(b-l-24-1) = 2b-l-24+l+24+b+b-l-25 = 4b-l-25$ bits.
  2. Pre-compute $2^{t/2}$ fixed points for the compression function h using message blocks N^j, each of size b bits, such that $H^*_{j-1} = h(H^*_{j-1}, N^j)$ for $j = 1, \ldots, 2^{t/2}$. While finding fixed points, fix the last $l+1$ bits of each message block N^j for the pad bit 1 and l bits to represent the pre-determined length encoding of the message m^* of length $4b-l-25$ bits to be produced as forgery. Let these $l+1$ bits be *padf* bits. Store the pairs (N^j, H^*_{j-1}) for $j = 1, \ldots, 2^{t/2}$ in a Table L_1.
- *On-line phase:*
  1. Query the signer with $2^{t/2}$ equal length chosen messages m^i for $i = 1, \ldots, 2^{t/2}$ (the message can also be the same in these queries). Let $|m^i| = b+b-l-24$ bits. Every message m^i can be represented as $m^i = m^i_1 \| m^i_2$ where $|m^i_1| = b$ bits and $|m^i_2| = b-l-24$ bits. Store these $2^{t/2}$ messages in a Table L_2.

For $i = 1, \ldots, 2^{t/2}$, the signer computes the signatures s_i on the equal-length messages m^i as follows:

(a) The signer chooses a fresh random value r_i for every query i independent of the message m^i. The signer calculates three random values $r_{0,i}$, $r_{1,i}$ and $r_{2,i}$ for every chosen random value r_i following the RMX specification in Section 3.1 and Remark 1 as follows:
   i. $r_{0,i} = r_i \| 0^{b-|r_i|}$
   ii. $r_{1,i} = r_i \| r_i \| \ldots \| r_i$ such that $|r_{1,i}| = b$ bits and the last repetition of r_i is truncated if needed.
   iii. $r_{2,i} = r_{1,i}^{[b-l]}$ (as noted in Remark 1).

(b) The signer splits every message m^i as $m^i = m_1^i \| m_2^i$ where $|m_1^i| = b$ bits and $|m_2^i| = b - l - 24$ bits.

(c) The signer randomizes every message m^i as follows:
   i. $M_0^i = r_{0,i}$
   ii. $M_1^i = m_1^i \oplus r_{1,i}$
   iii. $M_2^i = (m_2^i \| 0^8 \| lpad) \oplus r_{2,i}$

(d) Let $padm$ represents the l-bit binary encoded format of the length of the message $M_0^i \| M_1^i \| M_2^i$. For every randomized message $M_0^i \| M_1^i \| M_2^i$, the signer computes the hash value by passing this message as input to the hash function H^{H_0}. The hash value for every randomized message is $\tilde{H}_{r_i}^{H_0}(m^i) = H^{H_0}(M_0^i \| M_1^i \| (M_2^i \| 1 \| 0^{l-1}) \| (0^{b-l} \| padm))$. The signer returns the signature $s_i = \mathsf{SIG}(\tilde{H}_{r^i}^{H_0}(m^i))$ of every message m^i.

2. For every queried message m^i where $i = 1, \ldots, 2^{t/2}$, the signer returns the pair (r_i, s_i).

– *Offline phase:*

1. Add the received pair (r_i, s_i) to the table L_2.
2. Using the random value r_i, compute three random values $r_{0,i}$, $r_{1,i}$ and $r_{2,i}$ following the RMX specification in Section 3.1 and Remark 1.
3. Randomize the message m^i as follows:
   (a) $M_0^{i'} = r_{0,i}$
   (b) $M_1^{i'} = m_1^i \oplus r_{1,i}$
   (c) $M_2^{i'} = (m_2^i \| 0^8 \| lpad) \oplus r_{2,i}$
4. Let $M^{i'} = M_0^{i'} \| M_1^{i'} \| M_2^{i'}$. The validity of the signatures s_i returned by the signer during the *on-line* phase of the attack ensures that $M_0^{i'} \| M_1^{i'} \| M_2^{i'} = M_0^i \| M_1^i \| M_2^i$.
5. Compute $\tilde{H}_{r_i}^{H_0}(m^i) = H^{H_0}(M_0^i \| M_1^i \| (M_2^i \| 1 \| 0^{l-1}) \| (0^{b-l} \| padm))$ and add the hash values $\tilde{H}_{r_i}^{H_0}(m^i)$ to the Table L_2. Let $\tilde{H}_{r_i}^{H_0}(m^i) = H_i$ for $i = 1, \ldots, 2^{t/2}$. Now the Table L_2 contains the values (m^i, r_i, s_i, H_i).
6. Find a collision between the $2^{t/2}$ hash values stored in the Tables L_1 and L_2. With a significant probability, we can find a hash value H_x from the Table L_2 and a hash value H_y^* from the Table L_1 such that:

$$H^{H_0}(M_0^x \| M_1^x \| (M_2^x \| 1 \| 0^{l-1}) \| (0^{b-l} \| padm)) = H_x = h(H_y^*, N^y) = H_y^*$$

where
(a) $M_0^x = r_{0,x}$ and $|M_0^x| = b$
(b) $M_1^x = m_1^x \oplus r_{1,x}$ and $|M_1^x| = b$
(c) $M_2^x = (m_2^x \| 0^8 \| lpad) \oplus r_{2,x}$ and $|M_2^x| = b - l$
(d) $N^y = (N^y)^{[b-l-1]} \| padf$
and $x \in \{1, \ldots, 2^{t/2}\}$, $y \in \{0, \ldots, 2^{t/2} - 1\}$. This step is illustrated in Figure 1(a) and 1(b).

7. Let $r'_{2,x}$ be the last l bits of $r_{1,x}$. Then $r_{1,x} = r_{2,x} \| r'_{2,x}$. Note that $|r_{2,x}| = b-l$. Let $padr = r'_{2,x} \oplus (1 \| 0^{l-1})$. Let $padr1 = (r_{1,x}^{[b-l]} \oplus 0^{b-l}) \| (r'_{2,x} \oplus padm)$. Note that $|padr1| = b$ bits.
8. Calculate $(N^y)^{[b-l-1]} \oplus r_{1,x}^{[b-l-1]} = n^y$ as shown in Figure 1(c). The probability that the last 24 bits of n^y are $0^8 \| lpad$ is 2^{-24}. These 24 bits represent the padding bits of the message randomized by RMX.
9. Let $m^* = m_1^x \| (m_2^x \| 0^8 \| lpad \| padr) \| padr1 \| (n^y)^{[|n^y|-24]}$ and $m = m_1^x \| m_2^x$. Note that $|m| = 2b - l - 24$ bits and $m^* = 4b - l - 25$ bits which is the same as predetermined in Step (1) of the *pre-computation phase* of the attack. The signature $\mathsf{SIG}(m)$ on the message m is also valid on m^* as $\tilde{H}_r^{H_0}(m) = \tilde{H}_r^{H_0}(m^*)$.
10. Finally, output the message m^* as the forgery of the message m.

Complexity: It takes $2^{t/2}$ operations of h to *precompute* fixed points; $2^{t/2}$ chosen message queries to the signer during the *on-line phase*; $2^{t/2}$ operations of h and XOR operations during the *off-line phase* and a probability of 2^{-24} to hit the correct padding bits of the RMX transform. Total complexity of the attack is approximately $2^{t/2+1}$ operations of the compression function and $2^{t/2}$ chosen messages. The memory requirements of the attack are as follows assuming that the size of the signature is four times that of the security level of $t/2$ bits of a t-bit hash (which is possible in DSA): $b \times 2^{t/2} + t \times 2^{t/2}$ bits in the *precomputation phase*; $2b \times 2^{t/2}$ bits in the *on-line phase* and $|r| \times 2^{t/2} + t \times 2^{t/2} + 2t \times 2^{t/2}$ bits in the *off-line phase*. The total memory required for the attack is equal to $(3b + 4t + |r|) \times 2^{t/2}$ bits. This is approximately $2^{t/2+3}$ memory of t-bit values.

Illustration: Forging a signature scheme based on RMX-SHA-256 requires about 2^{129} operations of the SHA-256 compression function, 2^{128} chosen messages and a probability of 2^{-24}. Assuming that $|r| = 128$ bits, this attack requires a memory of about 2^{131}. In comparison, as noted in [19], forging a signature scheme based on RMX-SHA-256 using second preimage attack of [23] requires about 2^{201} SHA-256 compression function operations, more than 2^{55} memory and one chosen message.

Remark 4. Our forgery attack on the RMX-hash-then-sign signature schemes is independent of the size of the random value r. Our analysis assumes that in the RMX specification, $b' = b - l - 24$ bits. The attack also works for about the same complexity when $b' < b - l - 24$ bits. However, when $b' > b - l - 24$ bits, the padding bits of the RMX transform are placed in the last two blocks M_L and M_{L+1}. Since, the last block M_{L+1} does not contain any message bits, it is not possible to generate a fixed point block which can represent this block and hence, the attack does not work.

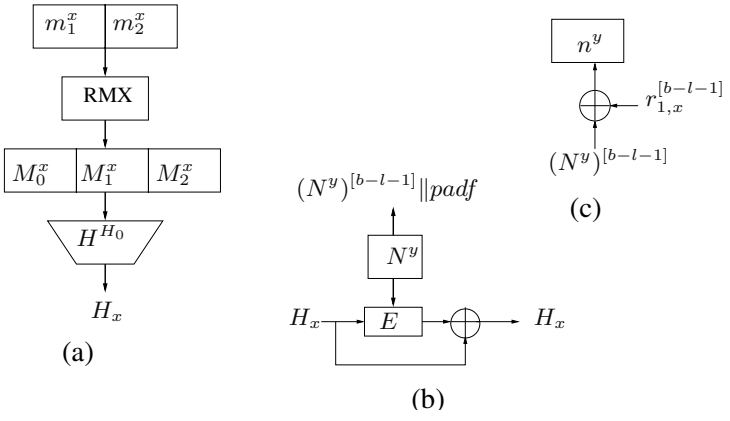

Fig. 1. Forgery attack on the RMX-hash-then-sign scheme based on Davies-Meyer

6.1 Applications of Our Forgery Attack

Our existential forgery attack on SIG based on RMX-hashes that use fixed point compression functions also works on SIG based on RMX_{SP}-hashes that use fixed point compression functions. When $|m| + 1 \geq |r|$ for RMX_{SP}, the complexity of our forgery attack on SIG using RMX_{SP} is similar to the one on SIG using RMX with the exception that it requires a success probability of 1/2 to hit the correct padding bit "1" used to pad the message by RMX_{SP}. The same attack also works on the signature schemes that use the previous version of RMX_{SP} [10] by assuming $|m| + 16 \geq |r|$. The attack has similar complexity as when RMX is used except that it has a success probability of 2^{-16} to hit the 16 padding bits used to pad the message by this variant.

Our forgery attack also works on the signatures based on the proposal $H^{H_0}(r\|H_r^{H_0}(m))$ [19] and on the signature schemes that use RMX transform together with the hash functions that use split padding [42] (assures that a minimum number of message bits are used in every block including the padding and length encoding block) to pad the message input to the hash function. Adding sequential counters to the RMX hash function (similar to the HAIFA hash mode [7]) also does not add any protection against our attack nor to the one in Section 4 as the counter inputs can be controlled in both the attacks. Note that sequential counters to the RMX hash function would still prevent the attempts to forge the RMX-hash-then-sign schemes using second preimage attacks of [23, 15].

Remark 5. Our on-line birthday forgery attack does not work on the signature schemes that use wide-pipe hash construction [25] with the internal state size $w \geq 2t$ based on the fixed point compression functions as the attack requires at least 2^t chosen messages and 2^{t+1} operations of the compression function. For example, Grøstl hash function [16], one of the selected candidates for the first round of NIST's SHA-3 hash function competition, uses a compression function for which fixed points can be easily found and has $w \geq 2t$ for a t-bit hash value.

6.2 Attack on the e-SPR Property of the Compression Functions

Our forgery attack on the RMX-hash-then-sign signature schemes translates into a birthday collision attack on the e-SPR property of the compression function h for which fixed points can be easily found. Recall that in the e-SPR game, we choose $u \geq 1$ values $\Delta_1, \ldots, \Delta_u$, each of length b bits. We then receive a random value $r \in \{0,1\}^b$ and define $m_i = r \oplus \Delta_u$ and $H_{i-1} = H^{H_0}(r \oplus \Delta_1 \| \ldots \| r \oplus \Delta_{u-1})$. Finally, we aim to find a pair (H^*_{i-1}, n_i) such that $(H^*_{i-1}, n_i) \neq (H_{i-1}, m_i)$ and $h(H_{i-1}, m_i) = h(H^*_{i-1}, n_i)$. The attack is outlined below:

1. Collect $2^{t/2}$ fixed point pairs (H_{i-1}, m^i) for h in a Table L where $i = 1, \ldots, 2^{t/2}$. Play the e-SPR game $2^{t/2}$ times always with $\Delta_1 = \Delta_2 = 0$ and every time we receive a fresh random value r_j for $j = 1, \ldots, 2^{t/2}$.
2. We check if $H_{i-1} = H^{H_0}(r_j \| r_j)$ for some i and j. Let that $r_j = r$ and fixed point be (H_{i-1}, m_i) where $m_i = m^i$ for some i.
3. Let $H^*_{i-1} = H^{H_0}(r)$, $n_i = r$, $H_{i-1} = H^{H_0}(r \| r)$. Now $h(H^*_{i-1}, n_i) = H^{H_0}(r \| r) = H^{H_0}(r \| r \| m_i) = h(H_{i-1}, m_i)$.

Thus, after an expected number of $2^{t/2}$ games, we win one game. Note that the forgery attack in Section 4 also translates into an e-SPR attack on any compression function after an expected number of $2^{t/2}$ e-SPR games.

7 Conclusion

Our research opens an interesting question on how to improve RMX SHA family without degrading its performance much to protect the signatures based on it against our forgery attack. One solution is not to mix the message bits processed using RMX transform with the padding and length encoding bits of the hash function by having only the latter bits in the last block. However, this patch introduces insufficient amount of randomness in the last block and requires some changes to the implementations of SHA family. It is an interesting problem to study this design. Recently, it has been suggested to use RMX-MD5 as a countermeasure to prevent impersonation attacks on the websites using collision attacks on MD5 [38]. Considering that one out of 2^{64} legitimate signatures based on RMX-MD5 can be forged following our results, it is an interesting problem to combine our techniques with those of [38] to analyse the security of RMX-MD5 over MD5 in the website certificates. Our research shows that randomized hashing is not easy to implement safely and we recommend NIST to consider our research during the SHA-3 hash function competition. It is clear from our attacks and those of [15,23] that it is well-worth investigating the SPR properties of the compression functions to identify possible weaknesses that may affect the randomized hashing setting.

Acknowledgments. Many thanks Quynh Dang and Ray Perlner for valuable discussions on this research and to Chris Mitchell for bringing to our awareness some important early works [14, 13, 1] on randomized hashing to strengthen the

digital signature security. Many thanks to our friends in the Information Security Group, RHUL for hosting Praveen Gauravaram to present preliminary results of this work and for their valuable discussions. Many thanks to the anonymous reviewers of Eurocrypt 2009 for some interesting comments and corrections; especially to the reviewer who suggested us to provide Section 6.2 and reminding us of the work of Dean [15]. Many thanks to our Crypto colleagues at MAT, DTU and Pierre-Alain Fouque for valuable discussions on this subject.

References

1. Akl, S.G.: On the Security of Compressed Encodings. In: Chaum, D. (ed.) Advances in Cryptology: Proceedings of Crypto 1993, pp. 209–230. Plenum Press, New York (1983)
2. Anderson, R., Biham, E.: Tiger: A Fast New Hash Function. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 89–97. Springer, Heidelberg (1996)
3. ANSI. ANSI X9.62:2005: Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA) (2005)
4. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
5. Bellare, M., Rogaway, P.: Collision-resistant hashing: Towards making uOWHFs practical. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 470–484. Springer, Heidelberg (1997)
6. Bellovin, S., Rescorla, E.: Deploying a New Hash Algorithm. In: Proceedings of NDSS. Internet Society (Feburary 2006)
7. Biham, E., Dunkelman, O.: A Framework for Iterative Hash Functions - HAIFA. Cryptology ePrint Archive, Report 2007/278 (2007) (Accessed on May 14, 2008), http://eprint.iacr.org/2007/278
8. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. Journal of Cryptology 21(2), 149–177 (2008)
9. Damgård, I.B.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
10. Dang, Q.: NIST Special Publication 800-106 Draft Randomized Hashing Digital Signatures (2007) (Accessed on July 21, 2008), http://csrc.nist.gov/publications/drafts/Draft-SP-800-106/Draft-SP800-106.pdf
11. Dang, Q.: Draft NIST Special Publication 800-106 Draft Randomized Hashing Digital Signatures (2008) (Accessed on August 6, 2008), http://csrc.nist.gov/publications/drafts/800-106/2nd-Draft_SP800-106_July2008.pdf
12. Dang, Q., Perlner, R.: Personal communication (October 2008)
13. Davies, D., Price, W.: Security for Computer Networks. John Wiley, Chichester (1984)
14. Davies, D.W., Price, W.L.: The Application of Digital Signatures Based on Public-Key Cryptosystems. In: Proc. Fifth Intl. Computer Communications Conference, pp. 525–530 (October 1980)
15. Dean, R.D.: Formal Aspects of Mobile Code Security. PhD thesis, Princeton University (1999)
16. Gauravaram, P., Knudsen, L.R., Matusiewicz, K., Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: Grøstl – A SHA-3 Candidate. First Round of NIST's SHA-3 Competition (2008) (Accessed on January 5, 2009), http://www.groestl.info/Groestl.pdf

17. Gauravaram, P., McCullagh, A., Dawson, E.: Collision Attacks on MD5 and SHA-1: Is this the "Sword of Damocles" for Electronic Commerce? In: Clark, A., McPherson, M., Mohay, G. (eds.) AusCERT Conference Refereed R & D Stream, pp. 1–13 (2006)
18. Goldwasser, S., Micali, S., Rivest, R.L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM Journal on Computing 17(2), 281–308 (1988)
19. Halevi, S., Krawczyk, H.: Strengthening digital signatures via randomized hashing. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 41–59. Springer, Heidelberg (2006), http://www.ee.technion.ac.il/~hugo/rhash/rhash.pdf
20. Halevi, S., Krawczyk, H.: The RMX Transform and Digital Signatures (2006) (Accessed on July 30, 2008), http://www.ee.technion.ac.il/~hugo/rhash/rhash-nist.pdf
21. Halevi, S., Shao, W., Krawczyk, H., Boneh, D., McIntosh, M.: Implementing the Halevi-Krawczyk Randomized Hashing Scheme (2007) (Accessed on July 28, 2008), http://www.ee.technion.ac.il/~hugo/rhash/implementation.pdf
22. Hohl, W., Lai, X., Meier, T., Waldvogel, C.: Security of Iterated Hash Functions Based on Block Ciphers. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 379–390. Springer, Heidelberg (1994)
23. Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
24. Lenstra, A.K., de Weger, B.: On the Possibility of Constructing Meaningful Hash Collisions for Public Keys. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 267–279. Springer, Heidelberg (2005)
25. Lucks, S.: A failure-friendly design principle for hash functions. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 474–494. Springer, Heidelberg (2005)
26. Merkle, R.C.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
27. Mironov, I.: Collision-Resistant No More: Hash-and-Sign Paradigm Revisited. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 140–156. Springer, Heidelberg (2006)
28. Miyaguchi, S., Ohta, K., Iwata, M.: Confirmation that Some Hash Functions Are Not Collision Free. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 326–343. Springer, Heidelberg (1991)
29. NIST. FIPS PUB 186-2: Digital Signature Standard (DSS) (January 2000) (Accessed on August 15, 2008), http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf
30. NIST. FIPS PUB 180-2-Secure Hash Standard (August 2002) (Accessed on May 18, 2008), http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf
31. NIST. Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. Docket No: 070911510-7512-01 (November 2007)
32. NIST. Draft FIPS PUB 186-3: Digital Signature Standard (2008) (Accessed on January 4, 2008), http://csrc.nist.gov/publications/drafts/fips_186-3/Draft_FIPS-186-3_November2008.pdf
33. Pasini, S., Vaudenay, S.: Hash-and-Sign with Weak Hashing Made Secure. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 338–354. Springer, Heidelberg (2007)

34. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: A synthetic approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994)
35. Rivest, R.L.: The MD4 Message Digest Algorithm. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 303–311. Springer, Heidelberg (1991)
36. Rivest, R.: The MD5 Message-Digest Algorithm. Internet Request for Comment RFC 1321, Internet Engineering Task Force (April 1992)
37. RSA Laboratories. *PKCS #1 v2.1: RSA Cryptography Standard*. RSA Data Security, Inc. (June 2002) (Accessed on August 15, 2008), `ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf`
38. Sotirov, A., Stevens, M., Appelbaum, J., Lenstra, A., Molnar, D., Osvik, D.A., de Weger, B.: MD5 Considered Harmful Today Creating A Rogue CA Certificate. Presented at 25th Annual Chaos Communication Congress (2008) (Accessed on January 3, 2009), `http://www.win.tue.nl/hashclash/rogue-ca/`
39. Stevens, M., Lenstra, A.K., de Weger, B.: Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 1–22. Springer, Heidelberg (2007)
40. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
41. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
42. Yasuda, K.: How to Fill Up Merkle-Damgård Hash Functions. In: Pieprzyk, J. (ed.) Advances in Cryptology - ASIACRYPT 2008. LNCS, vol. 5350, pp. 272–289. Springer, Heidelberg (2008)

A Observation in the Padding Rule of RMX

Consider hashing of a message m using RMX-SHA-256. For SHA-256, $b = 512$ bits. Let $|m| = 512 + 424 = 936$ bits where $|m_1| = 512$ and $|m_2| = 424$ bits. Following the specification of RMX given in Section 3.1, $b' = b - l - 24 = 512 - 64 - 24 = 424$ bits and $k = b - b' - 16 - l = 512 - 424 - 16 - 64 = 8$ bits. Let $|r| = 128$ bits. Now the randomized message M is defined as follows:

1. $M_0 = r_0$
2. $M_1 = m_1 \oplus r_1$
3. Calculation of M_2:
   (a) $M_2^* = m_2 \| 0^8 \| lpad$ where $|M_2^*| = 448$ bits
   (b) $M_2 = M_2^* \oplus r_2$

Therefore, $\text{RMX}(r, m) = M = M_0 \| M_1 \| M_2$. A hash function H^{H_0} used to process M requires at least $l+1$ bits for padding and length encoding. For SHA-256, $l = 64$ bits and hence it requires at least 65 bits for padding and length encoding. It is difficult to accommodate more than 64 bits in the remaining l-bit positions in the last block M_2 as it already has 448 bits. Therefore to process M using SHA-256, M is padded as follows: $M = M_0 \| M_1 \| \underbrace{(M_2 \| 1 \| 0^{63})}_{512 \text{ bits}} \| \underbrace{(0^{448} \| l)}_{512 \text{ bits}}$ where l represents the 64-bit binary encoded format of the length of M. Similarly, if

$b' = 423$ bits then $k = 9$ bits and $M_2^* = m_2 \| 0^9 \| lpad$. So, if $b' \leq b - l - 24$ then H^{H_0} requires an extra block to pad and length encode M.

Alternatively, when $b' \leq b - l - 24$ bits, we could define $k = b - b' - 24 - l$ bits. Then the hash function H^{H_0} does not require an extra block to length encode the message M. In the above illustration, when $|m| = 936$ bits, $M_2^* = m_2 \| 0^0 \| lpad$ and $M_2 = M_2^* \oplus r_2$ where $|M_2^*| = 440$ bits and $M = M_0 \| M_1 \| M_2$. To process M using a hash function H^{H_0}, M is padded as follows: $M = M_0 \| M_1 \| \underbrace{(M_2 \| 1 \| 0^7 \| l)}_{440+72 \text{ bits}}$.

B Message Randomization Technique RMX_{SP}

Let m be the input message, r be a message independent random bit string of at least 128 bits and at most 1024 bits and M be the randomized message. Let $zpad$ be a string of zero bits, which is zero or more "0" bits. Let λ denotes zero "0" bits or an empty string. Let $pad = 1 \| zpad$. Let $rpad$ be the 16-bit binary format of $|r|$. The input message m is encoded to the form $m \| pad$ and this encoded message is then randomized (transformed to M) as specified below.

1. If $|m| + 1 \geq |r|$:
   (a) $pad = 1 \| \lambda = 1$.
   Else
   (a) $pad = 1 \| 0^{|r| - |m| - 1}$.
2. $m' = m \| pad$.
3. If $|r| > 1024$ then stop and output an error indicator.
4. $rem = |m'| \bmod |r|$
5. Concatenate $\lfloor |m'|/|r| \rfloor$ copies of the r to the rem left-most bits of r to get R, such that $|R| = |m'|$. Now let

$$R = \underbrace{r \| r \| \ldots \| r}_{\lfloor |m'|/|r| \rfloor \text{ times}} \| r^{[rem]}$$

6. The randomized output is given by $M = \text{RMX}_{\text{SP}}(r, m) = r \| (m' \oplus R) \| rpad$.

Illustration: Let $|r| = 128$ and $|m| = 927$ bits. Now $|m| + 1 \geq r$, therefore $zpad = \lambda$ and $pad = 1$. Now $m' = m \| pad = m \| 1$ and $|m'| = 928$ bits. The random value $R = \underbrace{r \| \ldots \| r}_{7 \text{ times}} \| r^{[32]}$.