Analysis of RMAC

Lars R. Knudsen[1] and Tadayoshi Kohno[2]

[1] Department of Mathematics, Technical University of Denmark, lars@ramkilde.com
[2] Department of Computer Science and Engineering
University of California at San Diego, tkohno@cs.ucsd.edu

Abstract. In this paper the newly proposed RMAC system is analysed. The scheme allows a (traditional MAC) attack some control over one of two keys of the underlying block cipher and makes it possible to mount several related-key attacks on RMAC. First, an efficient attack on RMAC when used with triple-DES is presented, which rely also on other findings in the proposed draft standard. Second, a generic attack on RMAC is presented which can be used to find one of the two keys in the system faster than by an exhaustive search. Third, related-key attacks on RMAC in a multi-user setting are presented. In addition to beating the claimed security bounds in NIST's RMAC proposal, this work suggests that, as a general principle, one may wish to avoid designing modes of operation that use related keys.

1 Introduction

RMAC [6, 2] is an authentication system based on a block cipher. The block cipher algorithms currently approved to be used in RMAC are the AES and triple-DES.

RMAC is based on a block cipher with b-bit blocks and k-bit keys. RMAC takes as inputs: a message D of an arbitrary number of bits, two keys $K1, K2$ each of k bits and a salt R of r bits, where $r \leq k$. It produces an m-bit MAC value, where $m \leq b$. The method is as follows (see also Figure 1). First pad D with a 1 bit followed by enough 0 bits to ensure that the length of the resulting string is a multiple of b. Encrypt the padded string using the block cipher in CBC mode using the key $K1$. The last ciphertext block is then encrypted with the key $K3 = K2 + R$ where '+' is addition modulo 2. The resulting ciphertext is then truncated to m bits to form the MAC. The two keys $K1, K2$ may be generated from one k-bit key in a standard way [6].

There are five parameter sets in [6] for each of two block sizes.

Parameter Set	$b = 128$ (r, m)	$b = 64$ (r, m)
I	$(0, 32)$	$(0, 32)$
II	$(0, 64)$	$(64, 64)$
III	$(16, 80)$	n/a
IV	$(64, 96)$	n/a
V	$(128, 128)$	n/a

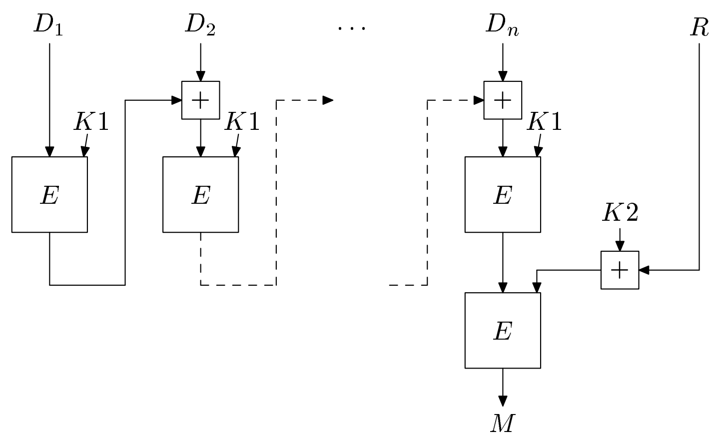

Fig. 1. The RMAC algorithm with keys $K1, K2$ on input a padded string $D_1 \| D_2 \| \cdots \| D_n$ and salt R. E is the underlying block cipher and '+' denotes addition modulo 2. The resulting MAC is M. We assume for illustrative purposes $m = b$ and $k = r$.

In Appendix A of [6] it is noted that for RMAC with two independent keys $K1$ and $K2$ an exhaustive search for the keys is expected to require the generation of 2^{2k-1} MACs, where k is the size of one key. However, for the cases with $m = b$ this can be done much faster under a chosen message attack with just one known message and one chosen message. Independently of how the two keys are generated, an exhaustive search for the key $K2$ requires only an expected number of 2^k decryptions of the block cipher [5]. Given a message D and the MAC using the salt R, request the MAC of D again. With a high probability this MAC is computed with a salt R', such that $R' \neq R$. For these two MACs, the values just before the final encryption will be equal and $K2$ can be found after about 2^k decryption operations. Subsequently, $K1$ can be found in roughly the same time.

The rest of this paper is organized as follows. In §2 an attack on RMAC used with three-key triple-DES is presented. The attack finds all three DES keys in time roughly that of three times an exhaustive search for a DES key using only a few MACs. §3 presents an attack on RMAC used with any block cipher. The attack finds one of the two keys in the system faster than by an exhaustive search. In §4 we present a construction-level related-key attack against RMAC and in §5 we describe some ways to exploit the related-key attack of §4 when attacking multiple users.

2 Attack on RMAC with three-key triple DES

One of the block cipher algorithms approved to be used in RMAC is triple-DES with 168-bit keys. Consider RMAC with parameter set II, that is with

64-bit MACs and a 64-bit salt. The key for the final encryption is then $K3 = K2 + (R\|0^{104})$. However, it is not specified in [6] how the three DES keys are derived from $K3$. Assume that the first DES key is taken as the rightmost 56 bits of $K2 + (R\|0^{104})$, the second DES as the middle 56 bits, and the third DES as the leftmost 56 bits. Assume an attacker is given two MACs of the same message D but using two different values, R and R' of the salt. Assume that the rightmost eight bits of both R and R' are equal. Then the encryption of the last same block for the two MACs is done using triple-DES where for one MAC the key used is (a, b, c), and where for the other MAC the key used is $(a, b, c \oplus d)$. Since the attacker knows d, he can decrypt through a single DES operation, find c in 2^{56} operations and derive one of the three DES keys[3]. This attack has a probability of success of 2^{-8}. If the attack fails, it is repeated for other values of D, R, and/or R'. After the third DES key has been found, it is possible to find the second DES key with similar complexity. Note that eight bits of the salt affect the second DES key. Request the MAC of a message D_2 using two different values of the salt. Decrypt through the final DES component with the third DES key. With a probability of $1 - 2^{-8}$ the two second DES keys in the final encryption will be different as a result of different salt values. Since the salts are known by the attacker, one finds the second DES in about 2^{56} operations. Subsequently, the final DES key can be found using 2^{56} MAC verifications [4] as follows. Assume one is given the MACs, M_1 and M_2, of two different messages D_1 and D_2, each consisting of an arbitrary number of bits. Let P_1 and P_2 be the padding bits used in the respective MAC computations. Request the MAC, M_3, of the message $D_1\|P_1\|E$, where E is a one-block message. Let x_1, x_2 and x_3 be the values just before the final triple DES encryptions in the computations of M_1, M_2 and M_3. Given the value of the final single-DES key of $K2$ one can compute also the MAC of the message $D_2\|P_2\|(E \oplus x_1 \oplus x_2)$. Note that the value just before the final triple DES encryptions in this case is x_3. Also note that the attacker has full control over the key bits which are modified using the (random) salts. Therefore this last part of the attack works regardless of how the salts are chosen, as long as the attacker knows them. In total, with 2 known and 1 chosen MAC, one finds the third DES key of $K2$ using 2^{56} MAC verifications or alternatively using 2^{56} chosen messages.

3 A generic attack

In this section we present an attack on the RMAC system with parameter set II for $b = 64$ and RMAC with parameter set V for $b = 128$. The attack finds the value of $K2$ after which RMAC reduces to a simple CBC-MAC for which it is well-known that simple forgeries can be found. In the following, let $d_K(x)$ denote the decryption of x using the key K for the underlying block cipher.

The attack is based on multiple collisions.

Definition 1. *A t-collision for a MAC is a set of t messages all producing the same MAC value.*

We shall make use of the following lemma which is easily proved.

Lemma 1. *Let A, B, and C be boolean variables. Then*

$$A \Rightarrow B \quad \Leftrightarrow \quad not(B) \Rightarrow not(A), \quad and$$

$$A \Rightarrow (B \ AND \ C) \quad \Leftrightarrow \quad not(B) \ OR \ not(C) \Rightarrow not(A).$$

Let D be some message (with an arbitrary no. of blocks). Then the MAC of D, $\mathrm{MAC}_{K1,K2}(D,R)$, is the last block from the CBC-encryption using $K1$, encrypted once again using the key $K2+R$, where R is the salt. The attack goes as follows. Request the MACs of D for s different values of the salt R. Assume that the attacker finds a t-collision, where the salts are $R_0, \ldots, R_{t-1}$ and denote the common MAC value by M'. For simplicity denote $K2+R_0$ by K, and $K2+R_i$ by $K+a_{i-1}$ for $i = 1, \ldots, t-1$. The attacker guesses a key value L and computes the decryptions of the MAC value M' using the keys $L, L+a_0, \ldots, L+a_{t-1}$. Then it holds for $i = 0, \ldots, t-1$, that if $L = K$ or $L = K+a_i$ then $d_L(M') = d_{L+a_i}(M')$. Using Lemma 1 one gets that if $d_L(M') \neq d_{L+a_i}(M')$ then $L \neq K$ and $L \neq K+a_i$ for $0 \leq i < t$. Similarly, if $d_{L+a_i}(M') \neq d_{L+a_j}(M')$ then $L \neq K + a_i + a_j$ for $0 \leq i \neq j < t$. In this way an exhaustive search for $K2$ can be made faster than brute-force.

In some rare cases one gets equal values in the inequality tests. As an example, if $d_L(M') = d_{L+a_i}(M')$ for some i, then one needs to check if $d_L(M') = d_{L+a_0}(M') = d_{L+a_1}(M') = \ldots$ after which all false alarms are expected to be detected. The expected number of false alarms is $t + \binom{t-1}{2}$.

Let us show the case of a 3-collision in more details. Assume that the random numbers, the salts used, are R_0, R_1, and R_2 (which are known to the attacker). Since the messages are the same for all MACs and since the MACs are equal, say M', one knows that the keys $K2 + R_0, K2 + R_1$, and $K2 + R_2$ all decrypt M' to the same (unknown) message z, thus

$$d_K(M') = d_{K+a_0}(M') = d_{K+a_1}(M'),$$

where $K = K2 + R_0$, $a_0 = R_0 + R_1$ and $a_1 = R_0 + R_2$.

The following implications are immediate.

$$\begin{array}{lll}
L = K & \Rightarrow \ d_L(M') = d_{L+a_0}(M') & \text{AND} \\
 & \ d_{L+a_0}(M') = d_{L+a_1}(M') & \\
L = K + a_0 & \Rightarrow \ d_{L+a_0}(M') = d_L(M') & \text{AND} \\
 & \ d_L(M') = d_{L+a_0+a_1}(M') & \\
L = K + a_1 & \Rightarrow \ d_{L+a_1}(M') = d_{L+a_0+a_1}(M') & \text{AND} \\
 & \ d_{L+a_1}(M') = d_L(M') & \\
L = K + a_0 + a_1 & \Rightarrow \ d_{L+a_0+a_1}(M') = d_{L+a_1}(M') & \text{AND} \\
 & \ d_{L+a_1}(M') = d_{L+a_0}(M') &
\end{array}$$

Table 1.

t	$u = t + \binom{t-1}{2}$	u/t
3	4	1.3
4	7	1.8
5	11	2.2
6	16	2.7
7	22	3.1
8	29	3.6
9	37	4.1
10	46	4.6
17	136	8.0

Lemma 1 enables us to rewrite the above implications as follows.

$$\begin{aligned}
d_L(M') \neq d_{L+a_0}(M') &\Rightarrow L \neq K \\
d_{L+a_0}(M') \neq d_L(M') &\Rightarrow L \neq K + a_0 \\
d_{L+a_1}(M') \neq d_L(M') &\Rightarrow L \neq K + a_1 \\
d_{L+a_1}(M') \neq d_{L+a_0}(M') &\Rightarrow L \neq K + a_0 + a_1
\end{aligned}$$

Take (guess) a key value, L and compute $d_L(M'), d_{L+a_0}(M')$, and $d_{L+a_1}(M')$. If $d_L(M') \neq d_{L+a_0}(M')$, then $L \neq K$ and $L \neq K + a_0$, if $d_{L+a_0}(M') \neq d_{L+a_1}(M')$, then $L \neq K + a_0 + a_1$, and if $d_L(M') \neq d_{L+a_1}(M')$, then $L \neq K + a_1$.

Summing up, with a 3-collision (provided a_0, a_1 are different) one can check the values of four keys from three decryption operations.

Let us next assume that there is a 4-collision. Let the four keys in the 4-collision be $K, K + a_0, K + a_1, K + a_2$. Then from the results of $d_L(M')$, $d_{L+a_0}(M')$, $d_{L+a_1}(M')$, and $d_{L+a_2}(M')$, one can check the validity of four keys. Moreover, by arguments similar to the case of a 3-collision, from the four decryptions, one can check the values of all keys of the form $K + a_i + a_j$, where $0 \leq i \neq j \leq 2$. Thus from four decryption operations one can check $4 + \binom{3}{2} = 7$ keys.

This generalizes to the following result. With a t-collision one can check the values of $u = t + \binom{t-1}{2}$ keys from t decryption operations. Table 1 lists values of t, u and u/t. It should be clear that t-collisions can be used to reduce a search for the key $K2$, one question is by how much. How many values of L need to be tested before the sets of keys $\{L, L + a_0, \ldots, L + a_{t-1}, L + a_0 + a_1, \ldots, L + a_{t-2} + a_{t-1}\}$ cover the entire key space?

Consider the case $t = 3$. One can assume $a_0 \neq a_1$ (otherwise there is no collision), and that with a high probability there are two bit positions where $a_0 \neq a_1$. Without loss of generality assume that these are the two most significant bits and that these bits are "01" for a_0 and "10" for a_1. Then a strategy is the following: Let L run through all keys where the most significant two bits are "00". Then clearly the sets

$$\{L, L + a_0, L + a_1, L + a_0 + a_1\}$$

cover the entire key space and an exhaustive search for $K2$ is reduced by a factor of $\frac{4}{3}$, since in the attack one can check the value of four keys at the cost of three decryptions.

Consider the case $t = 4$. With a high probability the b-bit vectors a_0, a_1, and a_2 are pairwise different. Also, with a high probability there are three bit positions where a_0, a_1, and a_2 are linearly independent (viewed as three-bit vectors). Without loss of generality assume that the bits are the three most significant bits and that these are "001" for a_0, "010" for a_1 and "100" for a_2. Then a strategy is the following: Let L run through all keys where the most significant three bits are "000". Then clearly the sets

$$\{L, L + a_0, L + a_1, L + a_2, L + a_0 + a_1, L + a_0 + a_2, L + a_1 + a_2\}$$

cover 7/8 of the key space. Next fix the most significant three bits of L to "111", find other bit positions where a_0, a_1, and a_2 are different and repeat the strategy. Thus, in the first phase of the attack one chooses 2^{b-3} values of L, does $4 \times 2^{b-3} = 2^{b-1}$ encryptions, and one can check $7 \times 2^{b-3}$ keys. In the next phase of the attack one chooses 2^{b-6} values of L, does $4 \times 2^{b-6} = 2^{b-4}$ encryptions, and one can check $7 \times 2^{b-6}$ keys. At this point, a total of $7 \times 2^{b-3} + 7 \times 2^{b-6} = 2^b - 2^{b-3} - 2^{b-6}$ keys have been checked at the cost of about $2^{b-1} + 2^{b-4}$ encryptions. In total, an exhaustive search for $K2$ is reduced by a factor of almost two.

For higher values of t the attacker's strategy becomes more complex. We claim that with a high probability ("good" values of a_i) the factor saved in an exhaustive search for the key is close to the value of u/t (see Table 1).

The following result shows the complexity of finding t-collisions [7].

Lemma 2. *Consider a set of s randomly chosen b-bit values. With $s = c2^{(t-1)b/t}$ one expects to get one t-collision, where $c \approx (t!)^{1/t}$.*

If it is assumed for a fixed message D and a (randomly chosen) salt R that the resulting MAC is a random m-bit value, one can apply the Lemma to estimate the number of texts needed to find a t-collision.

Consider a few examples. With $s = 2^{(b+1)/2}$ one expects to get one pair of colliding MACs, that is, one (2-)collision. With $s = (1.8)2^{2b/3}$ one expects to get a 3-collision, that is, three MACs with equal values ($6^{1/3} \approx 1.8$). With $s = (2.2)2^{3b/4}$ one expects to get one 4-collision ($24^{1/4} \approx 2.2$).

From Stirling's formula $n! = \sqrt{2\pi n}(n/e)^n(1 + \Theta(\frac{1}{n}))$, one gets that $(t!)^{1/t} \approx t/e$ for large t. Thus, with $s = (t/e)2^{(t-1)b/t}$ one expects to get a t-collision. Table 2 lists the complexities of finding t-collisions depending on the block size b.

There are many variants of this attack depending on how many chosen texts the attacker has access to. Table 3 lists the complexities of some instantiations

Table 2. The estimated number of texts needed to find a t-collision.

t	#texts needed $b = 64$	$b = 128$
3	2^{44}	2^{86}
4	2^{49}	2^{97}
5	2^{53}	2^{104}
6	2^{55}	2^{108}
7	2^{57}	2^{112}
8	2^{58}	2^{114}
9	2^{59}	2^{116}
10	2^{60}	2^{118}
17	2^{63}	2^{123}

Table 3. Expected running times and chosen texts of attacks finding $K2$ of RMAC.

Algorithm	k	b	Parameter sets	t	Expected running time	# chosen texts
3-DES	112	64	II	12	2^{108}	2^{63}
AES	128	128	V	20	2^{124}	2^{123}

of the attacks, where for triple-DES the number of chosen texts has been chosen to be less than 2^{64} (since the salt can be a maximum of 64 bits) and for AES the time complexity and the number of chosen texts needed have been made comparable. In both cases an exhaustive search for the key has been reduced by a factor of eight, so the correct value of the key can be expected trying half of that number of values. As a final remark, note that the message D in the attack need not be chosen nor known by the attacker. Therefore one can argue that this attack is stronger than a traditional "chosen-text" attack.

4 Construction-level related-key attacks

Another consequence of adding the salt to $K2$ is that it exposes the RMAC system to a construction-level related-key attack. Consider the RMAC system with parameter set II for $b = 64$ and RMAC with parameter set III, IV, or V for $b = 128$. Let $K1, K2$ and $K1, K2'$ be two pairs of RMAC keys that are related by the difference $K2' + K2 = X \| 0^{k-r}$ for some r-bit string X.

If D is some message, then $\mathrm{MAC}_{K1,K2}(D, R) = \mathrm{MAC}_{K1,K2'}(D, R + (X \| 0^{k-r}))$ with probability 1. An attacker can use this property to, for example, take a message MACed by one user (with keys $K1, K2$), change the salt by adding $X \| 0^{k-r}$, and then trick the second user (with related keys $K1, K2'$) to accept the new MAC–salt pair as an authenticator for D.

5 Key-collision attacks

Even if an attacker cannot control or does not (*a priori*) know the difference between multiple users' keys, an attacker can still exploit the related-key attack in §4. Consider RMAC with parameter set II for $b = 64$ and parameter set V for $b = 128$. Assume $k = r$ (if $r < k$ then treat the bits of $K2$ not affected by the salt as part of $K1$).

Let us start by assuming that we have two users who share the first key $K1$ but whose second keys $K2$ and $K2'$ have some unknown relationship. To mount the construction-level related-key attack from §4 the attacker must first learn the relationship between $K2$ and $K2'$. One way to learn this difference would be to first force each user to MAC some fixed message $2^{k/2}$ times. Let R_i be the i-th salt used by the first user and let M_i be the i-th MAC. Let R'_i be the i-th salt used by the second user and let M'_i be the i-th MAC.

If $K2 + R_i = K2' + R'_j$ for any indices i, j, then we have a key-collision for the key to the last block cipher application and $M_i = M'_j$ with probability 1. The attacker cannot observe the values $K2 + R_i$ directly, but if he sees a collision $M_i = M'_j$, then he guesses that the difference between $K2$ and $K2'$ is $R_i + R'_j$. Once this difference is known, the attacker can modify the MACs generated with $K1, K2$ to be valid MACs for $K1, K2'$. We expect to observe one collision $M_i = M'_j$ due to the key collision $K2 + R_i = K2' + R'_j$, and we expect 2^{k-m} collisions $M_i = M'_j$ at random, but recall that we are assuming that $k = m$. Note that if $M_i = M'_j$ occurs at random but $K2 + R_i \neq K2' + R'_j$, then with very high probability an attacker's subsequent forgery attempt will fail, and this is how we filter the signal from the noise.

Now consider a group of $2^{k/2}$ users, each with independently-selected random keys, and assume that the adversary forces each user to MAC some fixed message $2^{k/2}$ times. Note that, given a group of users this size, we expect two users to share the same first key $K1$ and, by the above discussion, we expect one collision $K2 + R_i = K2' + R'_j$ for this pair of users. By looking for collisions $M_i = M'_j$ across different users, an attacker can guess the relationship between two users' keys, and thereby force a user to accept a message that wasn't MACed with its keys.

Unfortunately, this attack against $2^{k/2}$ users has a much lower signal-to-noise ratio than the attack against two users who are known to share the first key $K1$. In particular, we expect approximately 2^{2k-m} collisions $M_i = M'_j$ at random. We filter the signal from the noise as before. The filtering step does not significantly slow down the attack since the attacker must already force $2^{k/2}$ users to each MAC $2^{k/2}$ messages and since we are assuming that $k = m$. As a concrete example, for AES with 128-bit keys, this attack works by forcing 2^{64} users to each MAC some message 2^{64} times. We expect 2^{128} collisions in the MAC outputs and one of those collisions will allow an adversary to take the messages MACed by one user and submit them as MACs to another user.

Another way to exploit the related-key property of §4 is based on the key-collision technique of [1]. For this attack let n denote the number of users an attacker is attacking, and let n', q, q' be additional parameters. The attack begins by the attacker picking keys $L1^u, L2^u$ for $u \in \{1, \ldots, n'\}$ (these keys correspond to "fake" users; these keys do not have to be random, but we assume that each $L1^u$ is distinct). Then, for each u, the attacker MACs some fixed message D q'

times; let $\overline{M}_i^u$ be the i-th MAC produced using keys $L1^u, L2^u$, and let $\overline{R}_i$ be the i-th salt value. We assume that each $\overline{R}_i$ is distinct, but not necessarily random.

Now assume that the attacker has each real user, indexed from 1 to n, MAC the message D q times, and let M_i^v be the i-th MAC produced by the v-th user, and let R_i^v be the i-th salt value for the v-th user (here we assume that all the salt values are chosen uniformly at random). Let $K1^v, K2^v$ denote the keys of the v-th real user. If $nn' \geq 2^k$ and $qq' \geq 2^k$, we expect at least one collision of the form $L1^u = K1^v$ and $L2^u + \overline{R}_i = K2^v + R_j^v$ to occur and, when this occurs, $\overline{M}_i^u = M_j^v$. If an adversary sees a collision of this form, it will learn both $K1^v$ and $K2^v$. We do, however, expect approximately $nn'qq'2^{-m}$ collisions $\overline{M}_i^u = M_j^v$ at random. This time, since we are guessing both RMAC keys, we can filter by recomputing the MAC of different messages using the key guess. (As an aside, note that the basic (total) key-collision attack approach of [1] would require $nn' \geq 2^{2k}$.)

We can instantiate this attack in different ways. If $n = n' = q = q' = 2^{k/2}$, then we get a key recovery attack (against one of the $2^{k/2}$ users) with resources similar to our previous attack against $2^{k/2}$ users. If $n = 1, n' = 2^k$, $q = q' = 2^{k/2}$, then we get an attack against a single user that uses $2^{k/2}$ chosen-plaintexts and approximately $2^{3k/2}$ steps. As a concrete example, if we consider AES with 128-bit keys, then the first instantiation attacks one of 2^{64} users using 2^{64} chosen-plaintexts per user, and 2^{128} offline RMAC computations (broken down into 2^{64} standard CBC-MAC computations and 2^{128} final RMAC block cipher applications). The attack also requires approximately 2^{128} additional block cipher applications as part of the filtering step. The latter instantiation attacks 1 user using 2^{64} chosen-plaintexts and approximately 2^{192} offline RMAC computations (broken down into 2^{128} standard CBC-MAC computations and 2^{192} final RMAC block cipher applications). The filtering phase requires an additional 2^{128} block cipher computations. As an additional note, we point out that the cost of the offline computations can be amortized across multiple attacks, thereby reducing the cost per attack.

6 Conclusions

There are several conclusions to draw from this work. The first and most obvious conclusion is that RMAC fails to satisfy the security claims in [6]. In particular, although NIST [6] claims that a key-recovery attack should require generating 2^{2k-1} MACs, we have presented a number of ways to extract RMAC keys using much less work.

We believe, however, that there are more important lessons to be learned from this research. First, our results suggest that one needs to be *extremely careful when using and interpreting "provable security" results*. What is being proven? And what assumptions are being made? In the case of RMAC we note that the proof of security is in the ideal cipher model. This is an extremely strong model and, unfortunately, not a good model for use with some popular block ciphers. For example, consider the attack against RMAC with triple-DES in §2. The attack in §2 worked because triple-DES is vulnerable to related-key attacks, whereas in the ideal cipher model there is no relationship between the permutations associated with different keys (each key corresponds to an independently selected random permutation). This suggests that the ideal cipher model is not a good model to use when designing a mode of operation.

Our results also show that, even when the underlying block cipher is secure against related-key attacks, interesting interactions can occur if a mode of operation uses related keys. For example, the attack in §3 reduces the search space of an exhaustive search attack by exploiting the fact that RMAC uses related keys. The construction-level related-key property in §4 also exists because RMAC uses related keys. And §5 shows the key-collision attacks become more serious when a mode of operation uses a large number of related keys. These attacks further support our recommendation that modes of operation should not use related keys.

Acknowledgments

We thank David Wagner for pointing out the relationship between the attacks in §5 and Biham's paper [1]. Tadayoshi Kohno was supported by a National Defense Science and Engineering Graduate Fellowship.

References

1. E. Biham. How to decrypt or even substitute DES-encrypted messages in 2^{28} steps. *Information Processing Letters*, 84, 2002.
2. E. Jaulmes, A. Joux, and F. Valette. On the security of randomized CBC-MAC beyond the birthday paradox limit: A new construction. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption 2002*. Springer-Verlag, 2002.
3. J. Kelsey, B. Schneier, and D. Wagner. Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and triple-DES. In Neal Koblitz, editor, *Advances in Cryptology: CRYPTO'96, LNCS 1109*, pages 237–251. Springer Verlag, 1996.
4. L.R. Knudsen and B. Preneel. MacDES: a new MAC algorithm based on DES. *Electronics Letters*, April 1998, Vol. 34, No. 9, pages 871–873.
5. Chris Mitchell. Private communication.
6. NIST. DRAFT Recommendation for Block Cipher Modes of Operation: the RMAC Authentication Mode. NIST Special Publication 800-38B. October 18, 2002.
7. R. Rivest and A. Shamir. Payword and Micromint: Two simple micropayment schemes. *Cryptobytes*, 2(1):7–11, 1996.