Related-Key and Key-Collision Attacks Against RMAC

Tadayoshi Kohno

CSE Department, UC San Diego
9500 Gilman Drive, MC-0114
La Jolla, California 92093-0114, USA

IACR ePrint archive 2002/159, 21 October 2002, revised 2 December 2002.

Abstract. In [JJV02] Jaulmes, Joux, and Valette propose a new randomized message authentication scheme, called RMAC, which NIST is currently in the process of standardizing [NIS02]. In this work we present several attacks against RMAC. The attacks are based on a new protocol-level related-key attack against RMAC and can be considered variants of Biham's key-collision attack [Bih02]. These attacks provide insights into the RMAC design. We believe that the protocol-level related-key attack is of independent interest.

Keywords: RMAC, key-collision attacks, related-key attacks.

1 Introduction

Jaulmes, Joux, and Valette's RMAC construction [JJV02] is a new randomized message authentication scheme. Similar to Petrank and Rackoff's DMAC construction [PR97] and Black and Rogaway's ECBC construction [BR00], the RMAC construction is a CBC-MAC variant in which an input message is first MACed with standard CBC-MAC and then the resulting intermediate value is enciphered with one additional block cipher application. Rather than using a fixed key for the last block cipher application (as DMAC and ECBC do), RMAC enciphers the last block with a randomly chosen (but related) key. One immediate observation is that RMAC directly exposes the underlying block cipher to a weak form of related-key attacks [Bih93]. We are interested in attacks that do not exploit some related-key weakness of the underlying block cipher, but rather some property of the RMAC mode itself.

In this work we present observations about, and attacks against, RMAC. We begin with a protocol-level related-key attack against RMAC. By *protocol-level related-key attack*, we mean an attack that works when an

adversary knows or can control the difference between two users' RMAC keys.[1]

While our protocol-level related-key attack may have limited applicability, we feel that the existence of this attack raises concerns about the RMAC design. In particular, since it is widely believed that block ciphers should be designed to resist related-key attacks, it seems counter-intuitive and bad engineering practice to use a block cipher in a mode of operation that is vulnerable to protocol-level related-key attacks.

The other attacks presented in this paper are based on the above protocol-level related-key attack against RMAC, but work when an adversary does not (a priori) know the differences between multiple users' RMAC keys. The attacks can be formulated in multiple ways. In the first formulation, an adversary takes a message–tag pair from one user and modifies the tag in such a way that another user (with different keys) will accept that message. The attack can also be formulated as a variant of Biham's key-collision attack [Bih02]. Assuming a block cipher with 128-bit keys and blocks (for a total RMAC key-length of 256 bits), in one instantiation of these attacks an adversary forces 2^{64} users to each tag 2^{64} messages, and the adversary exerts on the order of 2^{129} work. In another instantiation, an adversary forces one user to tag 2^{64} messages, but the adversary must now execute approximately 2^{192} steps. This beats the level of security claimed in [NIS02].

Related work. The related-key and key-collision attacks in this paper were published on the IACR ePrint server in October of this year [Koh02]. In November Lloyd and Knudsen announced additional observations about RMAC [Llo02,Knu02]. And in December Rogaway submitted further comments to NIST about RMAC [Rog02].

Assume an instantiation of RMAC with a block cipher with 128-bit blocks and 128-bit keys (for a total RMAC key-length of 256 bits). In [Llo02] Lloyd describes a forgery attack against RMAC that uses approximately 2^{128} block cipher decryptions and several chosen-plaintexts.[2] In [Knu02] Knudsen describes the same attack as Lloyd, but also presents a more sophisticated forgery attack that requires less time, but more

[1] We use the term *protocol-level related-key attacks* to distinguish between related-key attacks that exploit some property of the mode in question and related-key attacks that exploit some related-key property of the underlying block cipher.

[2] The actual presentation in [Llo02] is that of a complete key-recovery attack using 2^{129} steps; the forgery attack is a simple modification. Knudsen [Knu02] attributes the same forgery attack to Chris Mitchell.

message–tag pairs. The more sophisticated attack requires approximately 2^{124} time and 2^{123} chosen-plaintexts.

It seems reasonable to conclude that although Knudsen's attack requires less steps, the basic attack of Lloyd is more practical (albeit still theoretical) since it requires only a few chosen-plaintexts and significantly less memory. Nevertheless, the Knudsen attack is interesting because it highlights some unexpected properties with RMAC. The general (non-related-key) attacks presented in this paper require fewer message–tag pairs per user than Knudsen's attack, more message–tag pairs than Lloyd's attack, and more time than both Lloyd's and Knudsen's attacks. The attacks herein also illustrate some unexpected properties with the RMAC design.

OUTLINE. The remainder of this paper is organized as follows. In Section 2 we describe the RMAC message authentication scheme in more detail and in Section 3 we describe our notion of security (unforgeability) for MACs. We summarize related work, including Lloyd's and Knudsen's attacks, in Section 4. In Section 5 we describe a protocol-level related-key attack against RMAC and in Section 6 we modify the protocol-level related-key attack to work when the entire key difference is not known to the adversary. We present a generic attack against RMAC in Section 7, which is based on the attacks in Sections 5 and 6, and we present a Biham-style key-collision [Bih02] generalization of this generic attack in Section 8.

2 RMAC

TERMINOLOGY AND NOTATION. A message authentication scheme consists of three algorithms: a key generation algorithm, a tagging algorithm, and a verification algorithm. The key generation algorithm generates a random key. The tagging algorithm, on input a key and a message, outputs a tag (or MAC) for that message. The verification algorithm, on input a key, a message, and a candidate tag, returns accept if the candidate tag is a valid tag for the message and returns reject otherwise.

When presenting pseudocode, we use $\leftarrow$ to denote assignment from right to left, we use $\oplus$ to denote the XOR operation, and we use $\|$ to denote concatenation.

RMAC. We now describe the RMAC construction in more detail. The RMAC construction is parameterized by choice of an underlying block cipher F. Let k denote the block cipher's key length, let l denote the block cipher's block length, and let $F_K(B)$ denote the application of block cipher F on an l-bit block B with a k-bit key K. The RMAC algorithm

uses a total of $\alpha = 2k$ bits of key and produces a tag of length $l + k$. Before using the RMAC algorithm, a user first picks two random k-bit keys K_1 and K_2.

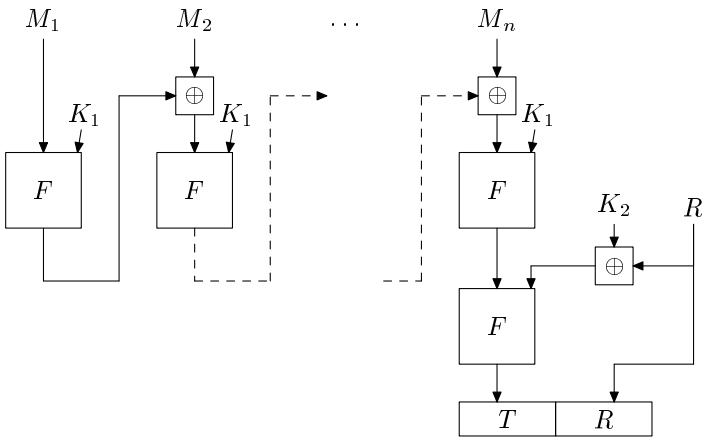

Fig. 1. The RMAC tagging algorithm with keys K_1, K_2 on input $M_1 \| M_2 \| \cdots \| M_n$. The underlying block cipher is denoted F. The randomness R is chosen anew on each invocation. The resulting tag is $T \| R$.

Let $\mathsf{RMAC}_{K_1,K_2}(M)$ denote the application of the RMAC tagging algorithm on a message M using keys K_1 and K_2. We assume that the length of M is a multiple of the block size (since we are presenting attacks, we only describe the variant of RMAC that we attack; the attacks immediately extend to RMAC defined over arbitrary bit-length messages). Pseudocode for the RMAC tagging algorithm is presented below (see also Figure 1):

$\mathsf{RMAC}_{K_1,K_2}(M)$
  Parse M into l-bit blocks $M_1 \| M_2 \| \cdots \| M_n$
  $C_0 \leftarrow 0$
  For $i = 1$ to n do
    $C_i \leftarrow F_{K_1}(M_i \oplus C_{i-1})$
  $R \leftarrow$ random k-bit value
  $T \leftarrow F_{K_2 \oplus R}(C_n)$
  Return (T, R)

The RMAC verification algorithm, $\mathsf{RVER}_{K_1,K_2}(M,(T,R))$, returns accept if (T,R) is a valid tag for M and returns reject otherwise; the verification algorithm is defined in the natural way.

3 Modeling Attacks

The standard notion of security for message authentication schemes is that of unforgeability [BKR94]. Consider an adversary $\mathcal{A}$ attacking a message authentication scheme (such as RMAC). We represent the legitimate user of a message authentication scheme with a "tagging oracle" $\mathsf{RMAC}^{\text{oracle}}_{K_1,K_2}(\cdot)$ (where K_1, K_2 are randomly chosen keys). The tagging oracle is a "black box" that takes a message as input and uses the keys K_1, K_2 to compute and return the RMAC tag of the message. We give adversary $\mathcal{A}$ access to this tagging oracle so that it can obtain an RMAC tag for any message of its choice. (Note that $\mathcal{A}$ only has input-output access to the tagging oracle; it cannot access the keys K_1, K_2.) A query to the tagging oracle corresponds to $\mathcal{A}$ forcing a user to tag a message of $\mathcal{A}$'s choice. We also give $\mathcal{A}$ access to a verification oracle $\mathsf{RVER}^{\text{oracle}}_{K_1,K_2}(\cdot,\cdot)$. The verification oracle represents the original user's intended correspondent. Adversary $\mathcal{A}$ "wins" if it can find a message-tag pair $(M,(T,R))$ such that $\mathsf{RVER}^{\text{oracle}}_{K_1,K_2}(M,(T,R))$ returns accept and $\mathcal{A}$ never queried the oracle $\mathsf{RMAC}^{\text{oracle}}_{K_1,K_2}(\cdot)$ with input M.

The above standard notion of security models an adversary attacking a single user or session. In the following sections, however, we shall sometimes consider an adversary attacking multiple users or sessions. To model such a scenario, we give an adversary access to multiple tagging oracles and their corresponding verification oracles, and say that an adversary wins if it can force *any* of the verification oracles to accept a message that was not tagged by its corresponding tagging oracle. This model corresponds nicely to an observation about real-world security requirements for message authentication schemes: a message authentication scheme (or any cryptographic scheme) will often be used by many different users simultaneously, albeit each user will probably use different keys. Consider, for example, Internet users who use SSL. Clearly an adversary with access to *all* simultaneous SSL connections should not be able to efficiently break *any* user's SSL connection with high probability.

4 Other Attacks

In recent works Lloyd and Knudsen announce additional observations about RMAC. In [Llo02] Lloyd observes that, given several tags for some

fixed message M, one can mount an exhaustive search for the key K_2. In more detail, if (T_1, R_1) and (T_2, R_2) are both tags for M, then if the key guess G for K_2 is correct, $F^{-1}_{G \oplus R_1}(T_1) = F^{-1}_{G \oplus R_2}(T_2)$. If the guess G is incorrect, then $F^{-1}_{G \oplus R_1}(T_1) \neq F^{-1}_{G \oplus R_2}(T_2)$ with high probability. In [Knu02] Knudsen shows how to reduce the number of keys one needs to try in order to learn K_2. The improvement in [Knu02] comes at the expense of an increased number of required message–tag pairs — if the underlying block cipher uses 128-bit keys, then Lloyd's attack requires approximately 2^{128} steps (block cipher decryptions) and a few message–tag pairs for a fixed message M, and Knudsen's attack requires approximately 2^{124} steps and 2^{123} tags for a fixed message M. Knudsen's attack also requires significantly more memory than Lloyd's basic attack.

After an adversary learns key K_2, RMAC reduces to basic CBC-MAC and an adversary could exploit the standard forgery attacks against CBC-MAC. An adversary could also perform an exhaustive key search for key K_1.

We also point out that a "standard" output-collision attack against a single user exists. In this attack the adversary forces a user to tag approximately 2^{128} messages of the adversary's choice (the adversary looks for a total collision in the output of the RMAC oracle). A modified variant of this attack, when mounted against 2^{64} users, requires the adversary to force each user to tag 2^{96} messages. This adversary against multiple users looks for a collision in the tags of one session, and will succeed in forging a message for that one session. Later we shall compare our attacks in Section 7 and Section 8 to this "standard" attack.

5 Known-Difference Related-Key Attack

We begin with a known-difference related-key attack against two users using the RMAC message authentication scheme. Let $\mathcal{U}$ and $\mathcal{V}$ denote the two users and let $K_1^{\mathcal{U}}, K_2^{\mathcal{U}}$ be user $\mathcal{U}$'s keys and let $K_1^{\mathcal{V}}, K_2^{\mathcal{V}}$ be user $\mathcal{V}$'s keys. Assume for this attack that $K_1^{\mathcal{U}} = K_1^{\mathcal{V}}$. Assume also that the adversary knows the difference D between $K_2^{\mathcal{U}}$ and $K_2^{\mathcal{V}}$; i.e., the adversary knows $D = K_2^{\mathcal{U}} \oplus K_2^{\mathcal{V}}$.

Although some may consider this attack to be somewhat unrealistic (requiring that the two users share the same first key and that the attacker knows a priori the difference between the second keys), we present this attack here because it motivates the attacks in the following subsections. Another perspective is that, since block cipher designers design block ciphers to resist related-key attacks, block cipher modes should also be

designed to resist related-key attacks. To do otherwise would be counter-intuitive and would negate much of the effort put into block cipher design.

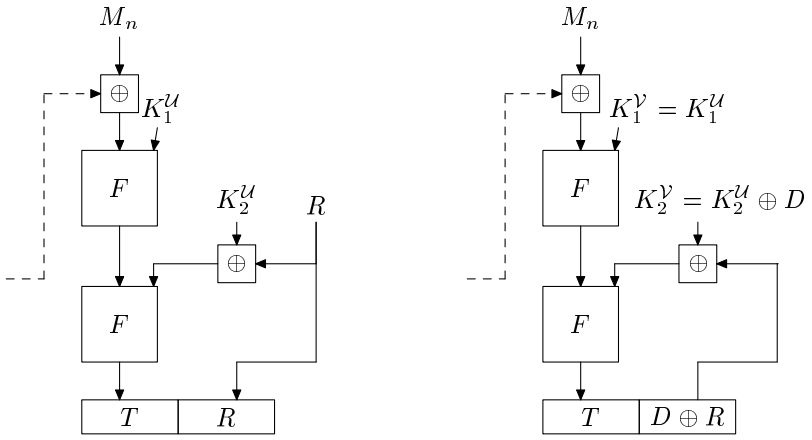

Fig. 2. The RMAC known-difference related-key attack (Section 5). The figure on the left shows how user $\mathcal{U}$ constructs a tag (T, R) for a message M in response to an adversary's query. The figure on the right suggests why user $\mathcal{V}$ will accept $(T, D \oplus R)$ as a tag for message M.

The adversary begins by having $\mathcal{U}$ tag a message M. Let (T, R) be the resulting tag; i.e., $(T, R) \leftarrow \mathsf{RMAC}^{\text{oracle}}_{K_1^{\mathcal{U}}, K_2^{\mathcal{U}}}(M)$. Since $K_2^{\mathcal{V}} = K_2^{\mathcal{U}} \oplus D$, user $\mathcal{V}$ will accept $(T, D \oplus R)$ as a tag for message M; i.e., $\mathsf{RVER}^{\text{oracle}}_{K_1^{\mathcal{V}}, K_2^{\mathcal{V}}}(M, (T, D \oplus R))$ will accept. See Figure 2. The query $(M, (T, D \oplus R))$ to the $\mathsf{RVER}^{\text{oracle}}_{K_1^{\mathcal{V}}, K_2^{\mathcal{V}}}(\cdot)$ oracle is considered a valid forgery because the adversary forces the user $\mathcal{V}$ to accept a message that was not tagged by one of the party's involved in $\mathcal{V}$'s session (i.e., not tagged by the $\mathsf{RMAC}^{\text{oracle}}_{K_1^{\mathcal{V}}, K_2^{\mathcal{V}}}(\cdot)$ oracle).

6 Partially-Known-Difference Related-Key Attack

Consider again the scenario in which an adversary is attacking the RMAC usage of two users $\mathcal{U}$ and $\mathcal{V}$. As before, let $K_1^{\mathcal{U}}, K_2^{\mathcal{U}}$ and $K_1^{\mathcal{V}}, K_2^{\mathcal{V}}$ respectively denote the two users' pairs of keys. Again assume that $K_1^{\mathcal{U}} = K_1^{\mathcal{V}}$. Unlike in Section 5, however, assume that the adversary does not know the difference D between the keys $K_2^{\mathcal{U}}$ and $K_2^{\mathcal{V}}$.

To mount a variant of the attack in Section 5, an adversary must first learn the difference between the keys $K_2^{\mathcal{U}}$ and $K_2^{\mathcal{V}}$. One way to learn this difference is shown in the following pseudocode. As discussed

in Section 3, the following adversary is given access to the tagging oracles $\mathsf{RMAC}^{\text{oracle}}_{K_1^{\mathcal{U}},K_2^{\mathcal{U}}}(\cdot)$ and $\mathsf{RMAC}^{\text{oracle}}_{K_1^{\mathcal{V}},K_2^{\mathcal{V}}}(\cdot)$ and the corresponding verification oracles $\mathsf{RVER}^{\text{oracle}}_{K_1^{\mathcal{U}},K_2^{\mathcal{U}}}(\cdot,\cdot)$ and $\mathsf{RVER}^{\text{oracle}}_{K_1^{\mathcal{V}},K_2^{\mathcal{V}}}(\cdot,\cdot)$.

Adversary
  $M, M' \leftarrow$ any two distinct messages
  For $i = 1$ to $2^{k/2}$ do // have $\mathcal{U}, \mathcal{V}$ tag M $2^{k/2}$ times
    $(T_i^{\mathcal{U}}, R_i^{\mathcal{U}}) \leftarrow \mathsf{RMAC}^{\text{oracle}}_{K_1^{\mathcal{U}},K_2^{\mathcal{U}}}(M)$
    $(T_i^{\mathcal{V}}, R_i^{\mathcal{V}}) \leftarrow \mathsf{RMAC}^{\text{oracle}}_{K_1^{\mathcal{V}},K_2^{\mathcal{V}}}(M)$
  For each pair of indices i, j such that $T_i^{\mathcal{U}} = T_j^{\mathcal{V}}$
    $(T, R) \leftarrow \mathsf{RMAC}^{\text{oracle}}_{K_1^{\mathcal{U}},K_2^{\mathcal{U}}}(M')$
    $\mathsf{RVER}^{\text{oracle}}_{K_1^{\mathcal{V}},K_2^{\mathcal{V}}}(M', (T, R_i^{\mathcal{U}} \oplus R_j^{\mathcal{V}} \oplus R))$ // forgery attempt

The above adversary begins by having both $\mathcal{U}$ and $\mathcal{V}$ repeatedly tag some message M. After each user generates approximately $2^{k/2}$ tags, we expect at least one collision between the last block cipher key used by $\mathcal{U}$ and the last block cipher key used by $\mathcal{V}$. That is, we expect to find two indices i, j such that $K_2^{\mathcal{U}} \oplus R_i^{\mathcal{U}} = K_2^{\mathcal{V}} \oplus R_j^{\mathcal{V}}$. The adversary detects this collision by looking for indices i, j such that $T_i^{\mathcal{U}} = T_j^{\mathcal{V}}$. When a collision $T_i^{\mathcal{U}} = T_j^{\mathcal{V}}$ occurs due to the above internal key collision, the adversary learns that the difference between the two keys $K_2^{\mathcal{U}}$ and $K_2^{\mathcal{V}}$ is $R_i^{\mathcal{U}} \oplus R_j^{\mathcal{V}}$. The adversary can use its knowledge of this difference to mount the attack in Section 5.

Assuming that the underlying block cipher is a family of independent random functions, after $2^{k/2}$ tagging requests per user we expect approximately 2^{k-l} additional collisions $T_i^{\mathcal{U}} = T_j^{\mathcal{V}}$ at random (not due to the above internal key collision). Thus we expect that an adversary may have to perform 2^{k-l} forgery attempts before it successfully forges a message. Provided that the underlying block cipher's key size k is not much larger than its block size l, an adversary will succeed after only a few forgery attempts.

7 Multi-User Attack

We shall now describe an attack against RMAC in the multi-user setting in which the attacker does not a priori know any information about the relationship between different users' keys. This attack extends the attacks of Section 5 and Section 6.

Consider the adversary shown in the following pseudocode. For this attack, we assume the adversary has tagging and verification oracle access

to $2^{k/2}$ different users (where each user's keys are chosen independently at random). Let $\mathsf{RMAC}^{\text{oracle}}_{K_1^u,K_2^u}(\cdot)$ and $\mathsf{RVER}^{\text{oracle}}_{K_1^u,K_2^u}(\cdot,\cdot)$ represent the uth user's tagging and verification oracles. As stated in Section 3, the adversary wins if it can force any of the users to accept a message which that user did not previously tag. To simplify the exposition, we present the attack in two phases; it should be clear that, if desired, the two phases can be interwoven.

Adversary
  Phase One:
    $M \leftarrow$ any message
    For $u = 1$ to $2^{k/2}$ do // for each of $2^{k/2}$ users
      For $i = 1$ to $2^{k/2}$ do // have user u tag M $2^{k/2}$ times
        $(T_i^u, R_i^u) \leftarrow \mathsf{RMAC}^{\text{oracle}}_{K_1^u,K_2^u}(M)$
  Phase Two:
    For each pair of distinct users u, v and for each pair of indices i, j such that $T_i^u = T_j^v$
      $M' \leftarrow$ a message not previously tagged by user v
      $(T, R) \leftarrow \mathsf{RMAC}^{\text{oracle}}_{K_1^u,K_2^u}(M')$
      $\mathsf{RVER}^{\text{oracle}}_{K_1^v,K_2^v}(M', (T, R_i^u \oplus R_j^v \oplus R))$ // forgery attempt

The intuition behind the above attack is that if two users u and v collide on their first keys K_1^u and K_1^v, then the adversary will be able to mount the attack in Section 6. In more detail: given $2^{k/2}$ users, we expect two users u and v to share the same first key $K_1^u = K_1^v$. After tagging $2^{k/2}$ messages each, we expect to find two indices i and j such that $K_2^u \oplus R_i^u = K_2^v \oplus R_j^v$. We detect this collision by looking for users u, v and indices i, j such that $T_i^u = T_j^v$. For the attack as presented in the above pseudocode, we expect to find approximately one such collision.

Unfortunately, the signal-to-noise ratio of this attack (as compared to the attack in Section 6) is greatly reduced; we expect up to approximately 2^{2k-l} collisions $T_i^u = T_j^v$ at random. Since a low signal-to-noise ratio is handled by brute forcing through the noise, a small signal-to-noise ratio only mean an inversely proportional large cost for the second phase of the attack. If we tolerate a cost of approximately 2^k oracle queries for the second phase of the attack (recalling that RMAC uses two keys for a total key length of $2k$), then the above attack works for k up to l. Larger k can be handled, but only at increased cost.

Concrete example. As a concrete example, if we consider RMAC instantiated with AES with 128-bit keys (and 128-bit blocks), then we expect this attack to succeed against 2^{64} users if an adversary can force each

user to tag 2^{64} messages and if the adversary can perform approximately 2^{129} steps (consisting largely of oracle queries). This compares favorably to the "standard" output-collision attack described in Section 4, but is certainly less efficient that Lloyd's attack [Llo02]. From a certain perspective, this attack can be considered more practical than Knudsen's attack [Knu02]. In particular, although this attack requires approximately 2^5 times as many operation as Knudsen's, it only requires each user to tag 2^{64} messages instead of 2^{123}. Although both these attacks are still infeasible in practice, it seems more reasonable to assume that an adversary can (undetectably) force a large set of users to tag 2^{64} messages than to force a single user to tag 2^{123} messages. Of course, it may be more easy for an adversary to monitor one communication channel than 2^{64} channels.

Additionally, Knudsen's attack has an advantage in that it works against a targeted user, whereas this attack works against one of 2^{64} users (and the attacker has no control over which user that will be). Still, as Biham points out in [Bih02], an attack against one of many users can still cause significant damage.

8 Key-Collision Attacks

Let us now consider modifying the attack in Section 7 to follow Biham's paradigm for key-collision attacks [Bih02]. Let n be the number of users an adversary is attacking and let n', q, q' be additional parameters for the attack. The adversary works as follows. As before, the attack is separated into phases for clarity, but the phases could be combined in an actual attack. The first phase can also be pre-computed and the cost of the first phase can be amortized over many attacks.

Adversary
  $M, M' \leftarrow$ any two distinct message
  Phase One:
    For $u = 1$ to n' do // "simulate" n' random users
      $L_1^u, L_2^u \leftarrow$ random RMAC key pair
      For $i = 1$ to q' do // have simulated user tag M q' times
        $(t_i^u, r_i^u) \leftarrow \mathsf{RMAC}_{L_1^u, L_2^u}(M)$
  Phase Two:
    For $v = 1$ to n do // for each of n users
      For $j = 1$ to q do // use oracle for user v to tag M q times
        $(T_j^v, R_j^v) \leftarrow \mathsf{RMAC}^{\text{oracle}}_{K_1^v, K_2^v}(M)$
  Phase Three:
    For each index u for the simulated users and v for the oracles

and for each pair i, j such that $t_i^u = T_j^v$
  // proceed assuming that $K_1^v = L_1^u$ and $K_2^v = L_2^u \oplus r_i^u \oplus R_j^v$
  $(T, R) \leftarrow \mathsf{RMAC}_{L_1^u, L_2^u}(M')$
  $\mathsf{RVER}^{\text{oracle}}_{K_1^v, K_2^v}(M', (T, r_i^u \oplus R_j^v \oplus R))$ // forgery attempt

If $nn' \geq 2^k$ and $qq' \geq 2^k$, then we essentially expect to find indices u, v, i, j such that $L_1^u = K_1^v$ and $L_2^u \oplus r_i^u = K_2^v \oplus R_j^v$. When this occurs, the forgery attempt in the third phase will succeed (and, moreover, the adversary will learn K_1^v and K_2^v). We expect approximately $nn'qq'2^{-l}$ collisions of the form $t_i^u = T_j^v$ at random. Assuming $nn'qq'2^{-l}$ is not significantly more than the sum of nq and $n'q'$, we tolerate this noise while mounting the above attack. Note that for a basic (total) key-collision attack we would require $nn' \geq 2^{2k}$.

By modifying the parameters n, n', q, q', we can instantiate the key-collision attack in different ways. If $n = n' = q = q' = 2^{k/2}$ and if $k = l$, then we get a key recovery and forgery attack with resources similar to that of Section 7. If $n = 1$, $n' = 2^k$, $q = q' = 2^{k/2}$, then we get an attack against a single user that uses $2^{k/2}$ oracle queries and approximately $2^{3k/2}$ steps. As a concrete example, if we consider AES with 128-bit blocks and 128-bit keys, then the first instantiation attacks 2^{64} users using 2^{64} oracle queries per user and approximately 2^{130} steps (broken down into 2^{129} computations of RMAC and 2^{129} tagging and verification oracle queries; the first 2^{128} RMAC computations are themselves broken down into 2^{64} standard CBC-MAC computations and 2^{128} final RMAC block cipher applications). The latter instantiation attacks 1 user using 2^{64} oracle queries and approximately 2^{192} steps (consisting mostly of RMAC computations). Another concrete example might be an adversary who forces each of 2^{32} users to tag 2^{64} messages and who executes approximately 2^{160} steps (consisting mostly of RMAC computations).

We again stress that the most interesting aspect of these attacks are not their resource requirements. Indeed, although these attacks beat the claimed security in [NIS02], they are still slower than the attack in [Llo02] and only arguably more practical than the attack in [Knu02] (depending on the definition of practical). Rather, we feel that these attacks are interesting because they expose additional, unusual properties of the RMAC construction and because they illustrate the use of Biham's key-collision attack on modes of operation that use a large number of keys.

Acknowledgments

The author is supported by a National Defense Science and Engineering Graduate Fellowship. Mihir Bellare, Alexandra Boldyreva, Chanathip Namprempre, Gregory Neven, Adriana Palacio, and David Wagner provided useful comments on an earlier version of this paper.

References

[Bih93] E. Biham. New types of cryptanalytic attacks using related keys. In T. Helleseth, editor, *Advances in Cryptology – Eurocrypt '93*. Springer-Verlag, 1993.

[Bih02] E. Biham. How to decrypt or even substitute DES-encrypted messages in 2^{28} steps. *Information Processing Letters*, 84, 2002.

[BKR94] M. Bellare, J. Kilian, and P. Rogaway. The security of cipher block chaining. In Y. Desmedt, editor, *Advances in Cryptology – Crypto '94*. Springer-Verlag, 1994.

[BR00] J. Black and P. Rogaway. CBC MACs for arbitrary-length messages: The three-key constructions. In M. Bellare, editor, *Advances in Cryptology – Crypto 2000*. Springer-Verlag, 2000.

[JJV02] É. 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.

[Knu02] L.R. Knudsen. Analysis of RMAC. NIST Modes of Operation for Symmetric Key Block Ciphers, Comments on Draft SP 800-38B, November 2002. `http://csrc.nist.gov/encryption/modes/comments/`.

[Koh02] T. Kohno. Related-key and key-collision attacks against RMAC. Cryptology ePrint Archive, Report 2002/159, October 2002. `http://55b...ug.95b2emmv...ollibeefood.rest/2002/159/`.

[Llo02] J. Lloyd. An analysis of RMAC. Cryptology ePrint Archive, Report 2002/170, November 2002. `http://55b...ug.95b2emmv...ollibeefood.rest/2002/170/`.

[NIS02] NIST. Draft recommendation for block cipher modes of operation: The RMAC authentication mode. NIST Special Publication 800-38B, 2002. `http://csrc.nist.gov/publications/drafts.html`.

[PR97] E. Petrank and C. Rackoff. CBC MAC for real-time data sources, 1997. DIMACS Technical Report 97-26. Available at `http://nb.nebag.tgm92egm.c0..ollibeefood.rest/TechnicalReports/abstracts/1997/`.

[Rog02] P. Rogaway. Comments on NIST's RMAC proposal. NIST Modes of Operation for Symmetric Key Block Ciphers, Comments on Draft SP 800-38B, December 2002. `http://..g.eeug.e0b.em5..9g..ollibeefood.rest/encryption/modes/comments/`.