An Analysis of RMAC

Jack Lloyd*

November 18, 2002

Abstract

A recent trend in message authentication is the use of a randomizing parameter, such that the authentication tag is based not only on the message and the key, but a public nonce which is changed for every authenticated message. This generally affords a better security proof. However, several new classes of attacks are made available by these techniques. We examine these attacks, and apply some of them to RMAC, a recently published MAC mechanism.

1 Introduction

Traditional techniques of confidentiality (such as a block cipher in CBC mode) do not provide authentication. While more recent methods (such as [9]) do provide both confidentiality and encryption in a single primitive, most confidentiality techniques require the use of an additional authentication mechanism. In addition, there is a wide variety of applications that have no need for confidentiality, but need authentication of messages.

Most common message authentication codes (MACs) are deterministic; they produce an authentication tag which is based only on the message and the secret key (for example [7]). A recent trend in MAC design is the use of a randomizing parameter or nonce, so that the tag is determined by the nonce as well as the key and message. These nonces often allow better security proofs as compared to standard MAC algorithms. However, it seems that very little thought has been given to the full ramifications of non-deterministic MAC functions.

2 General Attacks

There are some general classes of attacks applicable to randomized MACs precisely due to their non-deterministic nature. These are not general attacks in the sense that they necessarily apply to all non-deterministic MACs; whether they apply or not depends on the individual algorithms.

2.1 Repeated Messages

In the case of a deterministic MAC function (with a fixed key), a particular message will only have a single authentication tag. However, with a non-deterministic MAC, an attacker can collect a series of nonce/tag pairs (denoted by $(R_i, T_i) : 0 \leq i \leq n$) for a single message M. The most common attacks that come from this seem to be faster methods of exhaustive key search, and creating a new tag T with a new nonce $R \neq R_i : \forall i \in 0 \ldots n$. This second attack, at first glance, seems pointless, in that it is simply a replay attack. However, an application designer way well assume (particularly in the absence of statements to the contrary) that ensuring that nonces are never repeated is enough to prevent a replay attack when using a non-deterministic MAC.

*Email: `lloyd@srl.cs.jhu.edu` WWW: `http://www.randombit.net`

2.2 Repeated Nonce

Most, if not all, of the specifications for non-deterministic MACs require that a sender never authenticate two messages with a single nonce, and that receivers verify that nonces are non-repeating. However, as has been discussed elsewhere [5], application designers often misuse the cryptographic primitives they have available, sometimes to an astounding degree. The probability that at least some of them will be careless in ensuring that nonces are non-repeating is unity. So, while it unreasonable to expect that the MAC will retain it's full security when misused in such a way, they should be designed such that catastrophic failure is not the result of repeating nonces.

2.3 Other Possible Attacks

Other possible attacks, specific to non-deterministic MAC functions, include related nonce and chosen nonce attacks. Neither of these attacks are examined in detail in this paper.

3 RMAC

RMAC [6] is a recently published randomized MAC function, based on an underlying block cipher. It is essentially a simple CBC-MAC [2], with an extra operation at the end to randomize the authentication tag. For simplicity of discussion, we reuse the notations of [4], including that the block size of the underlying cipher is b bits.

To produce a MAC, RMAC uses two keys, K_1 and K_2, and a randomizing parameter, R. The length of R (denoted as r) can range from 0 to b bits. For simplicity, we will assume that $|K_1| = |K_2| = k$. RMAC is defined as:

RMAC(K_1,K_2,R,M) :

1. Divide M into b bit blocks $M_1, \ldots M_n$ (padding as needed)
2. $O_1 = E_{K_1}(M_1)$
3. $O_i = E_{K_1}(M_i \oplus O_{i-1}), 2 \leq i \leq n$
4. $K_3 = K_2 \oplus (R||0^{k-r})$
5. $T = E_{K_3}(O_n)$
6. Output (R, T) as an authenticator for M

In later discussion we will often refer to O_n as Z.

3.1 A key recovery attack on RMAC

If K_1 and K_2 are specified independently (as allowed by [4]), then RMAC is keyed with a total of $2k$ bits of key. From this, one might naively expect that it would take an average of 2^{2k-1} MAC operations in order to recover the secret keys (via key search). We will now show that this is not the case, and that, even if K_1 and K_2 are independent, the complete RMAC key can be recovered with an estimated 2^{k+1} work, and very little memory. In addition, no interaction with any legitimate user of the keys is necessary, beyond collection of a small number of message/nonce/tag triplets.

This attack only works in certain situations. In particular, we require that the complete tag, not a truncated version, be available to the attacker, and that $r > 0$. This requirement can be met within the limits set out in [4], and in fact is recommended for general use by that document.

Consider a single message, M, authenticated using several nonce/tag pairs, $(R_i, T_i), 0 \leq i \leq n$. We now describe how an attacker can recover the complete RMAC keys K_1 and K_2 using a simple divide and conquer attack.

For each key K in the space of K_2, compute $Z_i = D_{K \oplus R_i}(T_i)$ for $0 \leq i \leq n$. If $Z_i = Z_j, \forall i, j$, then K is a probable K_2. For each key we have to compute at least 2 decryptions (in order to have some basis for comparison). There is no need to compute more values of Z_i unless the first two match, and with very high probability this will not happen unless the K in question actually is K_2. Thus it should take about $2 \cdot 2^{k-1} = 2^k$ decryptions to find K_2. Once K_2 is determined, we also know $Z = Z_1 = ... = Z_n$, which is simply the CBC-MAC of M using K_1. This key can be recovered using the obvious key search attack now that we know Z, which we can use to verify a guess.

3.1.1 Work Estimates

For a message consisting of n blocks, the number of block cipher executions is $n+1$ for computing RMAC, and 2, 4, or 6 more for deriving K_1 and K_2 from the master key. Assuming (optimistically) that only 2 block cipher executions are required for deriving the keys, the average cost of a brute force search will be $2 \cdot 2^{k-1} \cdot (n+1) = n \cdot 2^k + 2^k$. Note this is the "easiest" case for a brute force attack: if K_1 and K_2 are independent, or if 4 or 6 applications of the block cipher are required to derive K_1 and K_2 from the master key, the costs will be even higher.

With our attack, the cost of searching for K_2 is about 2^k, and the cost of computing the CBC-MAC (searching for K_1) is n block cipher encryptions. This leads to a work estimate for our attack of $n \cdot 2^{k-1} + 2^k$ applications of the block cipher, which is significantly less than even the most optimistic estimate for brute force attacks on RMAC.

Finally, we note that these attacks are not at all inconsistent with the bounds of the security proof given by the designers of RMAC in [6]. In fact, further analysis may reveal them to be quite complimentary to the security proofs by providing a strong upper bound to RMAC's strength.

3.2 A forgery attack

We can use a variation on the previous attack to create arbitrary nonce/tag pairs for a single message with very little effort, given an upfront computation. Starting with a situation similar to the previous attack, we recover K_2 and Z with an estimated 2^k work. Knowing these two values, we can generate a tag for M using any R of our choice with very little effort by computing $T = E_{K_2 \oplus R}(Z)$.

It might seem obvious that K_2 can be recovered with 2^{k-1} effort, but a brute force attack on K_2 is actually harder than it seems, because at no point do we have a plaintext/ciphertext pair to base such an attack on.

3.3 Interactions between RMAC and DES

NIST's RMAC draft specification recommends RMAC be used with either AES [3] or DES [1] in EDE mode with 2 or 3 keys. Recall that some of the bits of a DES key do not have any effect on the cryptographic operation of the cipher, so when using RMAC with DES, 8 bits of R have absolutely no effect on the output. Thus, even if the users of RMAC are conscientious about ensure that R is unique, it's easy to generate a large series of "distinct" nonces that produce the same authentication tag for a given message.

The nature of DES-EDE's key schedule also causes some concerns. In particular, for a 64-bit block cipher, such as DES-EDE, r is limited to 64, but k can be much larger (192 bits in the case of 3 key DES-EDE). Thus, much of the latter portion of K_3 (in particular, the last 128 bits) are not affected by R at all. This means that only one of the three internal DES operations is dependent upon the value of R. The author knows of no attack which can make use of this property, but it is somewhat worrisome.

3.4 Workarounds for RMAC

While the attacks shown do not present serious concerns to the security of RMAC, considering the key lengths commonly supported by modern block ciphers, they do show that RMAC can be attacked faster than a brute force attack on the key. In addition, we have shown that, at least in some situations, little or no security benefit is gained from using two independent keys versus deriving the two keys from a single secret key. However, relatively simple countermeasures enable RMAC to be used safely, such always truncating the output of the MAC. This makes it much harder to take advantage of the fact that E is an invertible permutation (which our attacks rely on). The cautious may wish to avoid triple DES and use another cipher (such as AES).

If preventing replay attacks is a concern, relying solely on the RMAC nonce can be insufficient: include a serial number within the message body itself.

4 Conclusions

We first describe some general classes of attacks that can be applied to message authentication codes which make use of a nonce or randomization parameter in the generation of the authentication tag. We then applied some of these attacks to RMAC, showing several ways in which a simple divide and conquer attack can be used to break RMAC faster than brute force.

The properties of randomized MACs has, up to know, received very little attention, and it is hoped that this paper will spark further research in the area.

5 Acknowledgments

The author would like to thank Morris Dworkin for several helpful comments on an earlier draft of this paper.

References

[1] FIPS Publication 46-2, "Data Encryption Standard (DES)", U.S. DoC/NIST, October 25, 1999

[2] FIPS Publication 113, "Computer Data Authentication", U.S. DoC/NIST, 1994

[3] FIPS Publication 197, "Advanced Encryption Standard (AES)", U.S. DoC/NIST, November 26, 2001

[4] NIST Special Publication 800-38B (Draft) "Recommendation for Block Cipher Modes of Operation: the RMAC Authentication Mode", U.S. DoC/NIST, October 18, 2002

[5] P. Guttman, "Lessons Learned in Implementing and Deploying Crypto Software", Usenix '02

[6] É. Jaulmes, A. Joux, and F. Valette, "RMAC: A randomized MAC beyond the birthday paradox limit"

[7] H. Krawczyk, M. Bellare, and R. Canetti, "The Keyed-Hash Message Authentication Code (HMAC)", RFC 2104, February 1997

[8] P. Rogaway, "The Security of DESX" RSA Labs *Cryptobytes*, Vol. 2, No. 2, Summer 1996

[9] P. Rogaway, "OCB Mode: Parallelizable Authentication Encryption"