Multiple forgery attacks against Message Authentication Codes

David A. McGrew and Scott R. Fluhrer
Cisco Systems, Inc.
{mcgrew,sfluhrer}@cisco.com

May 31, 2005

Abstract

Some message authentication codes (MACs) are vulnerable to *multiple forgery attacks*, in which an attacker can gain information that allows her to succeed in forging multiple message/tag pairs. This property was first noted in MACs based on universal hashing, such as the Galois/Counter Mode (GCM) of operation for block ciphers. However, we show that CBC-MAC and HMAC also have this property, and for some parameters are more vulnerable than GCM. We present multiple-forgery attacks against these algorithms, then analyze the security against these attacks by using the expected number of forgeries. We compare the different MACs using this measure.

This document is a pre-publication draft manuscript.

1 Introduction

A Message Authentication Code (MAC) is a symmetric-key method that is used to protect a message from unauthorized alteration. A MAC consists of a tag generation algorithm and a tag verification algorithm. The tag generation algorithm takes as input a message M, a secret key K, and possibly a nonce IV, and returns a tag T. The tag verification algorithm takes as input M, T, K, and IV (if a nonce is used), and returns a binary value indicating whether or not the message/tag/nonce tuple is valid, that is, the tag provided as input matches the one produced by the tag generation algorithm applied to the other inputs. Some MACs use nonces, while others do not. For generality, we describe MACs as using nonces in this section.

In the standard model of MAC security, an adversary can send queries to a tag generation oracle and a tag verification oracle. These oracles model cryptomodules that implement the MAC and contain the secret key K. The tag generation oracle takes as input M and IV and returns T. The tag verification oracle takes as input M, T, and IV, and returns a binary output indicating whether or not the message/tag pair is valid. The adversary is allowed to make multiple queries to both oracles. For each oracle, the nonce values provided as inputs must be distinct. If the adversary is able to create a message/tag/nonce tuple that the tag verification oracle accepts as valid, and which does not correspond to any message/nonce pair provided to the tag generation oracle, we call this tuple a *forgery*.

Security analyses typically consider the probability that a computationally limited adversary can craft a forgery, and bound the number of queries that the adversary can make to either or both oracles. The probability of a single forgery has proven to be a useful measure of security, and many MACs have been shown to be secure with respect to it. However, some MACs are vulnerable to *multiple forgery attacks*, by which an adversary can create a large number of forgeries. The probability of a single forgery is not a good measure of security against these attacks. Consider the experiment in which we allow an adversary some number of queries to tag generation and tag verification oracles, then we count the number of forgeries that were made. That number is a random variable, which we denote as F. The value of $\mathbf{P}[F > f]$ for $f > 0$ is bounded by the conventional security measure of $\mathbf{P}[F > 0]$. However, the conventional measure tells us nothing about the probability of multiple forgeries. To better understand security against multiple forgery attacks, we compute the expected number of forgeries

$$E(F) = \sum_{f} f \, \mathbf{P}[F = f], \tag{1}$$

where E denotes the expectation operator.

The Galois/Counter Mode (GCM) of operation for block ciphers provides both authentication and confidentiality [1]. It essentially consists of counter mode encryption together with a MAC based on xor-universal hashing. In this work, we neglect the encryption aspect of GCM and focus on its message authentication component. Its authentication tag consists of a hash of the message exored with a pseudorandom value. As for all MACs that are based on the additive encryption of a hash value, GCM is vulnerable to multiple forgery attacks, as described in Section 3. Some commentators have pointed to this fact as a demerit of the mode. However, the most common MACs in practice, HMAC [2] and CBC-MAC [3], are also vulnerable to multiple forgery attacks, as we show in Section 2.1. Prior work has considered attacks similar to ours [4, 5], but has not focused on multiple forgeries as a measure of security. Other MACs are vulnerable to multiple forgery attacks; for example, Ferguson showed such an attack against OCB [6]. However, our

work focuses on GCM, CBC-MAC, and HMAC. We estimate the expected number of forgeries for these MACs, and show that the chained MACs are actually *worse* in this regard than GCM and MACs like it, for typical usage scenarios. We then compare the different MACs, and conclude by summarizing our results and observations.

In all of our analyses, we model the underlying cryptographic primitives as being ideally random. However, we conjecture that it is possible to incorporate an alternate measure of MAC security such as $\mathbf{P}[F > f]$ into a concrete proof of security.

2 Chained MACs

Several message authentication codes are *chained MACs*; they essentially use a block chaining mode. Both HMAC and CBC-MAC fall into this category. A chained MAC can be described with the following equations:

$$\begin{aligned} X_i &= \begin{cases} f(K) & \text{for } i = 0 \\ g(K, X_{i-1}, M_i) & \text{for } 1 \le i \le m-1 \end{cases} \\ T &= h(K, X_{m-1}, M_m). \end{aligned} \tag{2}$$

Here $X_i \in \{0,1\}^b$ for all i, where b is the number of bits of internal state of the MAC. The functions f, g and h are determined by the details of the MAC. We let $M_1, M_2, \ldots, M_{m-1}, M_m$ denote the unique sequence of bit strings such that $M = M_1 \| M_2 \| \ldots \| M_{m-1} \| M_m$, where $M_1, M_2, \ldots, M_{m-1}$ are complete blocks, each of which is s bits in length. The final block M_m may be a partial block, that is, its length is between one and s bits.

CMAC is a CBC-MAC variant that has been recommended by NIST [7]. For this MAC, there are $b = 128$ bits of internal state, and the block length $s = 128$ bits. The functions f, g, and h are given by

$$\begin{aligned} f(K) &= 0 \\ g(K, X, M) &= E(K, X \oplus M) \\ h(K, X, M) &= \text{MSB}_t(E(K, X \oplus \text{Pad}(K, M))), \end{aligned} \tag{3}$$

where $\text{Pad}(K, M)$ is a key-dependent padding function, which takes a partial block M as input and returns a complete block, and the function $\text{MSB}_t(X)$ returns a bit string containing the initial t bits of X.

HMAC uses an unkeyed collision-resistant hash function, such as MD5 or SHA1, to implement a keyed MAC. HMAC-MD5 has $b = 128$ bits of internal state, while HMAC-SHA1 has $b = 160$ bits of internal state. Both MACs operate on input blocks of size $s = 512$ bits. For a given hash function Hash, HMAC can be described as

$$\begin{aligned} f(K) &= c(IV, K \oplus P_I) \\ g(K, X, M) &= c(X, M) \\ h(K, X, M) &= \text{MSB}_t(\text{Hash}((K \oplus P_O) \| c(X, \text{Pad}(M)))), \end{aligned} \tag{4}$$

where P_I and P_O are the constants IPAD and OPAD, $c(X, M)$ is the compression function of the underlying hash function, IV is the initial vector of the hash, and $\text{Pad}(M)$ is the padding function used by the hash[1].

2.1 Multiple forgery attack

The fundamental observation of our attack is that, if we apply the MAC to two messages A and B with a common suffix,

$$\begin{aligned} A &= P_1 \| P_2 \| \ldots \| P_i \| S_{i+1} \| \ldots \| S_n \\ B &= Q_1 \| Q_2 \| \ldots \| Q_i \| S_{i+1} \| \ldots \| S_n, \end{aligned} \tag{5}$$

and if the internal state variables X_i^A, X_i^B are equal, then the two messages will have the same authentication tag, independent of the value of the suffix $S_{i+1} \| \ldots \| S_n$. The attack attempts to find a pair of distinct prefixes P and Q that yield a common internal state. We call the messages $P \| S$ and $Q \| S$ a *colliding pair*. After finding such a pair, we generate a forgery by requesting the tag for the message $P \| S'$, which is also the tag for the message $Q \| S'$.

The attack consists of two stages. In the first stage, a colliding pair is found, and in the second, it is exploited to make multiple forgeries. The first stage proceeds by making queries to the tag generation oracle, each of which has the form $A \| B$, where the suffix B is identical for each query, but the prefix A varies. If the internal variable X_1 is equal for for two distinct queries i and j, then the tags for those queries will match. The converse is not true; colliding tags do not necessarily indicate colliding internal variables. For reasonable tag sizes, the effect of these false positives is negligible on the effort of the attack. A single query to the tag verification oracle enables us to filter out a false positive. The birthday paradox makes it likely that we will find a colliding pair after $2^{b/2}$ queries to the tag generation oracle. After we have found a colliding pair $P \| S$ and $Q \| S$, we can generate forgeries at will. The second stage proceeds by querying the tag generation oracle with the message $P \| S'$ for some arbitrary value of $S \neq S'$, then querying the tag verification oracle with the message $Q \| S'$. This process is then repeated for different values of S'.

With CMAC, there is a one-to-one correspondence between X_1 and M_1. Therefore, we need to have prefixes that are at least two blocks in length. For HMAC, we can use prefixes that are a single block in length.

3 Multiple forgery attacks on GCM

GCM's message authentication tag can be computed as

$$T = \text{MSB}_t(r(K, IV) \oplus s(A, C)) \tag{6}$$

where r is a pseudorandom function and s is a universal hash. Here C is the ciphertext resulting from the encryption of the plaintext provided to GCM, and A is data that is authenticated but not encrypted. For our purposes, A and C together constitute the message M.

[1] The padding function depends on the length of the message; we ignore this detail as it is irrelevant to our analysis.

A multiple forgery attack against GCM works as follows. An adversary who knows two valid four-tuples (IV_0, C_0, A_0, T_0) and (IV_0, C_1, A_1, T_1) such that $C_0 \neq C_1$ and/or $A_0 \neq A_1$ can produce a forgery by observing another valid four-tuple (IV_2, C_2, A_2, T_2) then setting $C_3 = C_2 \oplus (C_0 \oplus C_1)$, $A_3 = A_2 \oplus (A_0 \oplus A_1)$, and $T_3 = T_2 \oplus (T_0 \oplus T_1)$. The same trick can be repeated for each new valid four-tuple (IV_i, C_i, A_i, T_i).

4 Expected number of forgeries

To measure the effectiveness of a MAC against multiple forgery attacks, we use the *expected number of forgeries*, which we denote as $E(F)$, where F is the random variable denoting the number of forgeries resulting from a particular attack. We allow the attacker q queries to each oracle. For an ideal MAC, that is, a true random function, the best that an attacker can do is to guess a different tag with each query of the verification oracle. With a tag size of t, each forgery attempt has a probability of 2^{-t} of succeeding. The number of forgeries obeys Bernoulli's distribution, and the expected number of forgeries after q attempts is $q2^{-t}$.

In the following, we consider the security of several MACs in the ideal model, in which the underlying cryptographic components are assumed to be truly random.

4.1 Universal Hashing

We next consider the case in which the forgery probability for a single query to the verification oracle is ϕ whenever there have not yet been any successful forgeries, and the probability of a repeat forgery is one. The probability that the first successful forgery will occur on the f^{th} try equals the probability $(1-\phi)^{f-1}$ that the first $f-1$ attempts will fail, times the probability ϕ that the next attempt will succeed. If the f^{th} attempt is the first successful one, then the adversary can generate a total of $q-f+1$ successful forgeries. Thus the expected number of forgeries after q queries is

$$E(F) = \sum_{f=1}^{q}(q-f+1)(1-\phi)^{f-1}\phi = \phi(q+1)\sum_{f=1}^{q}(1-\phi)^{f-1} - \phi\sum_{f=1}^{q} f(1-\phi)^{f-1}. \tag{7}$$

The geometric series is $\sum_{k=1,}^{n} r^k = \frac{r(1-r^n)}{1-r}$. We use this fact to compute $E(F)$:

$$\begin{aligned} E(F) &= \frac{\phi(q+1)}{1-\phi}\sum_{f=1}^{q}(1-\phi)^f + \phi\frac{\partial}{\partial\phi}\sum_{f=1}^{q}(1-\phi)^f \\ &= q - \frac{1-\phi}{\phi}(1-(1-\phi)^q). \end{aligned} \tag{8}$$

This value can be approximated by expanding the binomial $(1-\phi)^q$ and collecting leading terms for $q \ll 1/\phi$, the domain of interest, giving

$$E(F) = \frac{1}{2}q^2\phi + \mathcal{O}(q^2\phi^3) \tag{9}$$

Note that $E(F) \approx 1$ when the number of queries is about $\phi^{-1/2}$. For a MAC based on universal hashing with an ϵ-almost xor universal hash function, where the hash is exored with the output of a pseudorandom function, the probability of the first forgery is $\phi \leq \epsilon$.

For GCM using the AES block cipher [8], $\epsilon = (l+1)2^{-t}$, where l is the number of 128-bit blocks in the largest message and t is the number of bits in the authentication tag. An adversary guessing a tag has success probability of at most $\epsilon 2^{128}/(2^{128} - ql)$, using knowledge gained from the ql invocations of AES. Thus the expected number of forgeries for AES-GCM is

$$E(F_{\text{GCM}}) \approx \frac{l+1}{2} q^2 2^{-t}. \tag{10}$$

4.2 Chained MACs

For any chained MAC, the expected number of forgeries can be computed by considering the number of queries made during the first stage of the attack. That stage ends whenever there is a collision on the internal variable X_i. Consider the f^{th} query to the tag generation oracle. At this step, either we have found a colliding pair in a previous step, or we found the first colliding pair at this step, or we have not found a colliding pair. In either of the first two cases, we have found the information that we need in order to forge messages for the remaining queries. The probability that either of the first two cases holds is

$$1 - \frac{n!}{(n-f)!} \frac{1}{n^f}. \tag{11}$$

This probability considers collisions across internal states. Colliding tags which do not correspond to internal collisions are filtered out using a query to the tag verification oracle. The expected value of F is the sum of $\mathbf{P}[F > f]$ over all values of f:

$$E(F) = \sum_{f=1}^{q} 1 - \frac{n!}{(n-f)!} \frac{1}{n^f}. \tag{12}$$

Using Stirling's approximation and Taylor's expansion and then collecting terms gives

$$E(F) = \frac{1}{6} q^3 2^{-b} + \mathcal{O}(q^4 2^{-2b}) \tag{13}$$

The expected number of forgeries is cubic in the number of queries, with an error term that is small when $q \ll 2^{b/2}$. Thus this approximation is reasonable in the typical domain of usage of the MAC. In this domain, $E(F)$ can be nearly $1/6 \cdot 2^{b/2}$. In short, the expected number of forgeries is a significant fraction of the number of queries that are allowed.

For Merkle-Damgärd hash functions such as MD5 and SHA1, the collision probability increases with increasing message length. An attacker can use this fact to improve the attack on HMAC by using longer prefixes. In Equation 12, we neglect the fact that a collision may occur at any point in the chain $X_1, X_2, \ldots$. This collision is significant if $P_i = Q_i$, where i is the number of blocks in the prefix. An attacker can exploit this property to improve the expected number of collisions. Because of this, Equation 12 underestimates the value of $E(F)$ for HMAC.

5 Comparison

We summarize our results by denoting the expected number of forgeries for an ideal MAC, for AES-GCM, and for a chained MAC as $E(F_{\text{Ideal}}), E(F_{\text{GCM}})$, and $E(F_{\text{Chained}})$, respectively. We use the approximations derived earlier, which are all valid for $q \ll 2^{b/2}$ and $q \ll 2^t/l$:

$$\begin{aligned} E(F_{\text{Ideal}}) &\approx q\,2^{-t} \\ E(F_{\text{GCM}}) &\approx q^2\,\frac{l+1}{2}2^{-t} \\ E(F_{\text{Chained}}) &\approx q^3\,\frac{1}{6}2^{-b}. \end{aligned} \tag{14}$$

In terms of q, the expected number of forgeries is linear for an ideal MAC, quadratic for GCM, and cubic for a chained MAC.

The most common MACs in practice are the chained MACs HMAC and CBC-MAC. The expected number of forgeries is significantly higher for these MACs than for an ideal MAC, even for values of q and t well within the domain of typical usage.

If a chained MAC is used with a tag of length less than about $b - 2\lg q$ bits, then a tag-guessing strategy will be more effective at producing multiple forgeries than the attack described in Section 2.1. Thus, in situations in which a relatively high forgery probability is acceptable but it is desirable to keep the expected number of forgeries low, chained MACs are suitable. In contrast, the expected number of forgeries for GCM is ql times higher than that for an ideal MAC, making it less attractive for use with short tags. The value of q can be kept low by limiting the amount of data processed under any given key, though the cost of key establishment restricts the usefulness of this strategy.

For larger tags, however, GCM does better than a chained MAC. GCM has fewer expected forgeries than a chained MAC whenever $l < q2^{t-b} < 2^{t-b/2}$. The second inequality follows from our limitation that q is below the birthday bound of $2^{b/2}$. When the chained MAC is CMAC using the AES block cipher, then $b = 128$. With a tag size $t = 96$ bits, GCM has fewer expected forgeries even when the number of 128-bit blocks l in each message is up to 2^{32}.

In our analysis, HMAC-MD5 has identical characteristics to CMAC because it also has $b = 128$. HMAC-SHA1 provides more security against multiple forgery attacks because it uses $b = 160$ bits of internal state. With a tag size $t = 96$ bits, GCM has fewer expected forgeries than HMAC-SHA1 even when the number of 128-bit blocks l in each message is up to 2^{16}. However, we have not yet optimized our attack to take advantage of the multiple internal collisions that can occur in HMAC. The actual security of HMAC with respect to these attacks will decrease as the message length increases.

6 Conclusions

Many MACs are vulnerable to multiple forgery attacks, including the commonly used chained MACs HMAC and CBC-MAC, and MACs using xor-universal hashing with additive encryption, like GCM. The

vulnerability of these MACs is captured by the expected number of forgeries $E(F)$. The vulnerability of the chained MACs grows faster than that of GCM as the number of messages that are protected grows. For typical parameters, GCM provides better resistance against repeat forgery attacks. However, when short authentication tags are used, chained MACs are less vulnerable.

Several commentators have noted GCM's vulnerability to multiple forgery attacks. However, we are aware of no similar complaints against HMAC, CMAC, or any other CBC-MAC variants. This disparity suggests that multiple forgery attacks are more of a theoretical concern than a practical one.

References

[1] D. McGrew and J. Viega, The Galois/Counter Mode of Operation (GCM), *Submission to NIST Modes of Operation Process*, January 2004.

[2] H. Krawczyk, M. Bellare, R. Canetti, HMAC: Keyed-Hashing for Message Authentication, RFC 2104, *IETF Request for Comments*, 1997.

[3] M. Bellare, J. Kilian, P. Rogaway. The Security of the Cipher Block Chaining Message Authentication Code. *J. Comput. Syst. Sci.* 61(3). pg. 362-399 (2000).

[4] B. Preneel, P. van Oorschot, On the security of iterated Message Authentication Codes, *IEEE Transactions on Information Theory*, 45, 1999.

[5] K. Brincat and C. J. Mitchell, New CBC-MAC forgery attacks, *Information Security and Privacy, ACISP 2001*, Springer-Verlag (LNCS 2119), Berlin (2001).

[6] N. Ferguson, Collision Attacks on OCB, *unpublished manuscript*, 2002. Available online at `http://www.cs.ucdavis.edu/ rogaway/ocb/links.htm`.

[7] M. Dworkin. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, *NIST Special Publication 800-38B*, May, 2005.

[8] U.S. National Institute of Standards and Technology, The Advanced Encryption Standard. *Federal Information Processing Standard* (FIPS) 197, 2002.