Authentication Failures in NIST version of GCM

Antoine Joux

DGA
and
Université de Versailles St-Quentin-en-Yvelines
PRISM
45, avenue des Etats-Unis
78035 Versailles Cedex, France
Antoine.Joux@m4x.org

Abstract. In this note, we study the security of the Galois/Counter mode authenticated encryption recently published by NIST. We show how an adversary can recover the secret key of the keyed hash function underlying the authentication, using a chosen IV attack. Once this secret key is known, the encryption mode is no longer authenticated. As a consequence, all chosen ciphertext attacks against the confidentiality become feasible. Moreover, since the encryption mode is a counter mode, i.e. a stream cipher, the XOR malleability of the encrypted plaintext becomes a major security issue.

1 Introduction

Recently, the National Institute of Standard and Technology has published several recommend modes of operation for use with the AES block cipher. The fourth recommendation in this series [2] describes an authenticated encryption mode based on a proposal of McGrew and Viegra [4]. The goal of this recommendation is to provide a very efficient mode of operation, that performs a single block cipher invocation per message block and does not need access to block cipher decryption even when decrypting messages. The latter property is very useful, from a performance point of view, when using a substitution permutation algorithm such as the AES. Indeed, when using the AES implementing both encryption and decryption is costly and avoiding decryption altogether is a worthy goal. This issue has not attracted much attention in the past, since with Feistel schemes such as DES, decryption is just encryption with the order of the subkey reversed. Still, when performing encryption without authentication, a well-known solution exist: the counter mode.

However, the counter mode does not offer the level of security expected nowaday for encryption mode. Indeed, it is only designed to resist chosen

plaintext attack. The Galois/Counter mode (GCM) proposed by NIST in [2] follows the classical encrypt-then-MAC approach in order to turn the counter mode into a encryption scheme secure against chosen-ciphertext adversaries.

In order to cover a wide range of application, GCM also allows the authentication of some associated data send in the clear together with the encrypted message. The MAC used in GCM loosely follows the Wegman-Carter paradigm from [5], a keyed hash is computed and an encrypted copy of the hash value is added to the ciphertext to provide authentication. Since GCM is based on a counter mode, it requires an initialization vector denoted by IV. The default length of the IV is 96 bits, however, for added flexibility GCM allows the use of IVs of different length and for a given key the length of the IV does not even need to be fixed. It is widely known that replaying an IV in a counter mode voids the confidentiality of all messages encrypted with this IV. As a consequence, it is specified in GCM that IVs should never be repeated. Nevertheless, in order to introduce our attacks againsts NIST version of GCM, we first describe a "forbidden attack" involving a repeated IV. This forbidden attack is mainly proposed for illustration purposes.

The paper is organized as follows: in section 2 we recall the description of the Galois/Counter mode, in section 3 we present the forbidden attack with repeated IVs, in section 4 we study the security of the mechanism which allows the use of an IV of length different than the default (96 bits) and present two attacks against this mechanism as described in [2]. Finally, since our attacks allow the adversary to learn the authentication key, we recall in section 5 the security implications.

2 Description of GCM

The Galois/Counter mode (GCM) authenticated encryption can be used to encrypt a plaintext and authenticate the resulting ciphertext together with associated data. It relies on two basic primitives, a counter mode encryption, where the basic block cipher is used to generated a pseudo-random keystream, which is xored with the plaintext and a message authentication code which serves to authenticate the resulting ciphertext together with its associated data. Thus, GCM follows the encrypt-then-MAC paradigm which, since [1], is usually considered as the "right" way to compose a secret key encryption and a MAC into an authenticated encryption. The MAC used in GCM is a variation on the Wegman-Carter authentication, one first computes the value of a keyed universal hash

function on the ciphertext and associated data and then encrypts it. It differs from standard practice by the fact that this final encryption in the MAC computation also uses a counter mode encryption instead of simply applying the block cipher to the result of the hash function. Another deviation from standard practice is the fact that the plaintext encryption and the MAC encryption both use the same key, instead of independent keys as usually assumed in the encrypt-then-MAC paradigm.

In order to describe GCM, we need to introduce a few notations. First, we let w (an even number ≥ 128) denote the block length in the available block cipher E, we let K denote the secret key used for encryption/decryption. GCM is used to encrypt a plaintext P together with its associated data A into a ciphertext C under an initial value IV. The plaintext, the ciphertext and the associated data are all seen as sequences of blocks of length w. For example, P is decomposed as:

$$P = P_1 \| P_2 \| \cdots \| P_n^*.$$

Note that the final block P_n^* may be an incomplete block. By convention, $\hat{C}$ and $\hat{A}$ respectively denote C and A right padded with enough zeroes to fit on the smallest possible number of full blocks. Similarly, when using IVs of varying length, we use the notation $\widehat{IV}$ to denote IV right padded with enough zeroes up to the next full block. We also use the notation $[x]_\ell$ to denote the ℓ-bit encoding of the number x.

Before describing GCM itself, we first need to define the universal hash function used in the Wegman-Carter MAC. This function called GHASH receives as input a key H and two strings M and M' constituted of full blocks. The core of GHASH consist in evaluating a polynomial defined from M at H, more precisely:

$$\begin{aligned}\mathrm{GHASHcore}_H(M, M') = &\sum_{i=1}^{\mathrm{len}(M)} M_i H^{\mathrm{len}(M)+\mathrm{len}(M')+1-i} \\ &+ \sum_{i=1}^{\mathrm{len}(M')} M'_i H^{\mathrm{len}(M')+1-i}.\end{aligned}$$

All the computations are performed by embedding H and the blocks of M and M' into some fixed representation of the Galois field $\mathbb{F}_{2^w}$. GHASH itself is defined from GHASHcore by adding the binary representation of the lengths of M and M' (denoted by $\mathrm{len}(M)$ and $\mathrm{len}(M')$) in the strings being processed:

$$\mathrm{GHASH}_H(M, M') = \mathrm{GHASHcore}_H(M, M' \| [\mathrm{len}(M)]_{w/2} \| [\mathrm{len}(M')]_{w/2}).$$

Another important auxillary function is counter encryption of a string. Given as input a counter value J and a string P, this function $\mathrm{GCTR}_K(J, P)$ outputs an encrypted string C. Each block of C is computed from the corresponding block of P by $C_i = P_i \oplus E_K(\mathrm{inc}^{i-1}(J))$, where inc denotes counter incrementation[1], with the convention that inc^{i-1} represents $i-1$ successive incrementations. When encrypting the final block, the output of E_K is truncated to match the length of this final block.

We can now define GCM authenticated encryption with the following steps:

1. Compute the hash key: $H = E_K(0^w)$
2. Compute the basic counter value J from IV:
   - If $\mathrm{len}(IV) = w - 32$ then $J = IV \| 0^{31}1$
   - If $\mathrm{len}(IV) \neq w - 32$ and NIST description then
   $$J = \mathrm{GHASHcore}_H(\{\}, \widehat{IV})$$
   - If $\mathrm{len}(IV) \neq w - 32$ and original description then
   $$J = \mathrm{GHASH}_H(\{\}, \widehat{IV})$$
3. Compute the ciphertext $C = \mathrm{GCTR}_K(\mathrm{inc}(J), P)$
4. Compute the keyed hash: $S = \mathrm{GHASH}_H(A, C)$
5. Encrypt the hash value: $T = \mathrm{GCTR}_K(J, S)$
6. Return C and T

First, we remark that the original description of GCM [4] and the NIST version of GCM [2] are subtly different when an IV of length other than $w - 32$ is used. The difference occurs when computing the internal counter initializing value J. In the original version, this is done using GHASH, which implies that IV is padded with its length. In the NIST version, the computation of J uses GHASHcore and the length of IV is not added. We also remark that in the above description, the MAC length is exactly w, whereas GCM allows MAC truncation to a shorter length in order to fit various needs. However, our attacks work better with non truncated MACs so we ignore this possible truncation in the sequel. Moreover, the NIST description of GCM specifies at page 7, that for IV of length different from 96 bits (i.e. $w - 32$ since the NIST specification assumes that $w = 128$) the MAC size shall be 128. Since our attacks

[1] In the definition of GCM, this only increments the number encoded by the last 32 bits of J modulo 2^{32}, however, the precise definition of inc is irrelevant for the attack presented here.

precisely work with this kind of IVs, we do not need to consider the truncated MAC case.

Since GCM is an authenticated encryption, it is essential when decrypting to check the MAC value before decrypting. If the MAC is invalid no plaintext is computed and a special symbol FAIL should be produced. Moreover, it is clear when reading the encryption procedure that decryption can be computed using the block cipher E in the forward direction only, thus hardware implementations can be made cheaper by omiting AES decryption.

2.1 Known weaknesses

As far as we know, the only published weaknesses of GCM are described in a paper of Ferguson [3]. The paper described two weaknesses, the first shows that truncating tags too much is dangerous and leads to a high probability for collision. The second weakness shows that in this context, one may learn information about the hash key H. In the rest of this paper, we consider this risk of learning H when a collision occurs in a slightly different light and show how NIST modification to GCM largely compound this risk.

3 A forbidden attack with repeated IV

While both the NIST and the original GCM specification make clear that an IV should never be repeated, in this section, we consider a forbidden attack with a repeated IV. This mainly serves illustration purposes. In fact, this forbidden is used in the sequel as a tool to construct real attacks in the NIST version of GCM.

In order to describe our attack with the weakest possible adversary, we assume that it is passive and only sees complete ciphertext data, including IV, associated data and MAC tag. On the other hand, we are considering the forbidden case where an IV value is repeated. Thus we assume that the adversary sees two different messages $M^{(1)}$ and $M^{(2)}$ encrypted with the same IV. Upon reception of these two messages, he proceeds as follows. From the two MAC tags, $T^{(1)}$ and $T^{(2)}$, he computes $T^{(1)} \oplus T^{(2)} = S^{(1)} \oplus S^{(2)}$. As explained in section 2, $S^{(1)}$ is the value of a polynomial at H, moreover the coefficient of the polynomial are derived from the ciphertext blocks and known by the adversary. Similarly, $S(2)$ is also the value of a known polynomial at H. Since $S^{(1)} \oplus S^{(2)}$ is known, the adversary learns that H is a root of a polynomial he knows. This is essentially the core of the second weakness described by Ferguson in [3].

One important problem for the adversary is that the degree of this polynomial can be high, since it is equal to the length in blocks of the longer of the two messages $M^{(1)}$ and $M^{(2)}$. Since the number of roots can potentially be as high as the degree, the adversary may hesitate between a large number of possible H. However, on average, we only expect a small number of roots. Moreover, if the adversary can obtain a second pair of messages with common IVs, he gets a second polynomial with root H. Then by computing the GCD of the two polynomials, he finds a polynomial of small degree with H as a root. Finally, he is left with a small number of candidates for H. If there is a single candidate, he is done and stops there. If there are a few, he can either collect additional pairs with common IVs or use a chosen ciphertext attack as in section 5 to test each possible value.

3.1 Discussing the IV assumption

Forbidding the repetition of IVs is a standard assumption in stream ciphers and counter modes. Indeed, it is well understood that repeating an IV for two different messages completely voids the confidentiality of these two messages. This comes from the simple fact that the two plaintexts are xored with the same pseudo-random string and such a "two-time pad" is of course insecure. However, if for some reason[2] a small number of IVs are repeated, it is usually expected that beyond this basic attack which reveal the corresponding messages, nothing too bad should happen. With an ordinary counter mode, this is indeed the case. However, with both versions of GCM, allowing a small number of repeated IVs leads to a serious leak where the authentication key is revealed. As usual, learning key material gives much more power to the adversary as shown in section 5. While the basic fact that information about the authentication key is leaked is already known from [3], the security risks induced by this leak have not been seriously considered. In some sense, the unavoidable assumption that IV are not repeated is even stronger with GCM than with ordinary counter modes.

4 Non default length in IV

In this section, we present two different attacks which only work with the NIST version on GCM described in [2] but do not affect the original version from [4]. These two attacks rely on a bad property of the

[2] Such as a disfunction in a counter or random generator.

keyed function GHASHcore used by NIST to compute the internal starting value of the counter J when using IVs of length different from 96 bits. Both attacks, assume that the adversary is granted the power to choose both the plaintext and the corresponding IV. However, according to the specifications, he is not allowed to use the same IV twice.

Varying length IVs In the NIST version of GCM, the key fact for our first attack is that we can collide the internal initialization value J with external IVs which are formally different. First, remark that for an initial value IV not a multiple of w bits, we have the property that:

$$\text{GHASHcore}_H(\{\}, \widehat{IV}) = \text{GHASHcore}_H(\{\}, \widehat{IV\|0}).$$

This is straightforward, since both $\widehat{IV}$ and $\widehat{IV\|0}$ denote IV right padded with zeroes up to the next full block. As a consequence, the adversary can easily feed two different initial values into GCM and get an internal collision on the initial counter values. This allows him to use the forbidden attack of section 3 in a context where the attack is no longer forbidden.

All zero IV Our second attack is based on a similar idea, but instead of having a collision between initialization value, we force a collision between some internal value and the authentication key. The attack makes use of an IV string of the form 0^ℓ with $\ell \neq w - 32$. In the NIST version of GCM, the algorithm is initialized with $J_0 = 0 \cdot H = 0$. Moreover, the resulting tag value T is equal to the encryption of J_0 xored with S. Since J_0 is the all zero block, it encrypts to the authentication key H under the key K. Also, as in section 3, S is the value of a known polynomial at H. Moreover, since the adversary is allowed to choose the plaintext, he can use a short plaintext. Since the degree of the polynomial grows with this length, S is the value of a known low degree polynomial at H. Moreover $T = S \oplus T$ also is the value of a low degree polynomial at H. Thus the value of H is learned.

It is worth noting that when following the original specification and using GHASH instead of GHASHcore, none of the two above attacks apply. It is clear from the footnote on page 7 of [2] that this change in the specification is not a typo but was done on purpose. However, no rationale is given for this ill advised modification.

5 Further considerations

All the three attacks, including the forbidden attack, described in sections 3 and 4 let the adversary learn the key H of the universal hash

function underlying the MAC scheme. Of course, once this key is known, the adversary can compute the keyed hash of any ciphertext he wish to fake. Moreover, since the MACs are encrypted hash values using the counter mode encryption, it is extremely easy using a pair of XORs to replace a hash value by another as long as the internal counter J is the same. Thus, once H is known, any arbitrary string can be substituted in place of a valid ciphertext with associated data. The only restriction is that the IV cannot be changed or more precisely that an IV can only be substituted by an equivalent IV. Since computing equivalent IVs (i.e. IVs giving the same values of J) is easily done when H is known, at least for IVs of non default length, the adversary has a wide range of options. First, he can freely replace any associated data by any string of his choice, he can truncate or expand ciphertext, he can use the XOR malleability of stream cipher to produce a valid encryption of any string of his choice given a known plaintext/ciphertext pair. To sum it up, the only thing the adversary cannot do is recover the main key K. However, the main goal of cryptography is to protect the security of data not keys. Thus the GCM version described in [2] does not fill its security goals.

An important remaining question is how to repair the GCM scheme. The basic answer is of course to remove the modification introduced between [4] and [2]. However, taking into account the risk that disfunctions might induce IV collisions, it might be reasonable to perform a few changes to GCM. In this context, replacing the counter encryption for MACs by the classical encryption [3] with the block cipher usually used with Wegman-Carter MACs seems a safe option. Moreover, in order to further mitigates the security risks, we suggest to use a strong key derivation at the beginning of the algorithm and computing a different key for each different purpose (one for encryption, one for intiializing J, one for the keyed hash and one for the MAC encryption).

6 Conclusion

In this paper, we have shown an important attack of the NIST version GCM mode. This stems from the fact that GCM excessively relies on the hypothesis that IVs are never repeating. Moreover, the modification introduced by NIST turns this fact into a effective attack when variable length IVs are used.

[3] Note that this does not require to add access to E^{-1}, since the MAC equality testing can be done on encrypted blocks.

References

1. M. Bellare and C. Namprempre. Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In Springer-Verlag, editor, *Proceedings of Asiacrypt 2000*, 2000.
2. M. Dworkin. Recommendation for block cipher modes of operation: Gcm for confidentiality and authentication. NIST special publication 800-38D, April 2006.
3. N. Ferguson. Authentication weaknesses in gcm. NIST web page http://csrc.nist.gov/CryptoToolKit/modes/comments, May 2005.
4. D. McGrew and J. Viegra. The security and performance of the Galois/Counter mode. In Springer-Verlag, editor, *Proceedings of INDOCRYPT'04*, 2004. Full paper available on eprint, report 2004/193.
5. M. Wegman and L. Carter. New hash functions and their use in authentication and set equality. *J. of Comp. and System Sciences*, 22:265–279, 1981.