GCM Update

David McGrew and John Viega
mcgrew@cisco.com, viega@securesoftware.com

May 31, 2005

1 Introduction

The original GCM specification submitted to NIST [1] was written with minimal guidance on parameter choices, and a minimal security section that referred to external work. In retrospect, it would have been better to have included strong guidance on parameters, especially the length of the authentication tag. A recently published revision [2] does so, without changing any other details.

Recently, Niels Ferguson submitted comments to NIST detailing an attack on GCM message authentication when authentication tags are truncated [3]. This work underscores the risks of using GCM with very short tags, and highlights the need for guidance on tag length. However, it does not violate the claims of GCM's security analysis [4], nor does it present any weakness that was not described in that analysis. GCM is able to provide high levels of security, a fact that Ferguson does not dispute.

In the following, we review the authentication weaknesses described in Ferguson's comments and the performance advantages that motivate the design. We also review some ongoing related work that analyzes similar weaknesses in other message authentication codes. We then address several other points made by Ferguson that are related to engineering issues.

2 Authentication strength

Ferguson's comments describe "two weaknesses in the the authentication functionality of GCM when it is used with a short authentication tag. The first weakness raises the probability of a successful forgery significantly. The second weakness reveals the authentication key if the attacker manages to create successful forgeries." However, these points are both raised in the GCM security analysis [4]. The second point is stated in the first paragraph of Section 5, which also explains why we chose a design with that property. The first point is implicit in Theorem 2 and is referred to in the text following that theorem, though it is not explicitly stated.

The fact that a cryptographer of Ferguson's caliber could read our peer-reviewed GCM analysis, yet be surprised when he derives the optimum attack against GCM, hints at a problem. As Ferguson suggests, the standard the definition-theorem-proof style that we used probably distracted from some of the details of the algorithm. Of course, we feel that the security proof is valuable, but some simpler statements of security are also needed. We have addressed this need in the updated specification.

In Ferguson's own words, his attack "does not provide a counterexample to GCM's proof of security." His contribution is to show a clever way to prosecute a forgery attack in the case that the authentication tags are truncated. He considers matrix representations of finite field operations that lead to a succinct expression for forgery attacks that will meet the upper bound on the probability of success. Ferguson's work dramatizes the relative weakness of GCM to multiple forgery attacks, relative to other MACs such as HMAC and CBC-MAC. GCM requires slightly longer authentication tags to achieve the same authentication strength. However, GCM is still able to provide high security. We provide more details below.

We expect that, for any new mode providing authentication that NIST approves, they will make a recommendation similar to the one they made for CMAC in Appendix A.2 in FIPS Special Publication SP 800-38B. One core idea of that guidance is that authentication tags of less than 64 bits in length "shall only be used in conjunction with a careful analysis of the risks of accepting an inauthentic message as authentic." Similar guidance may be useful for GCM.

2.1 Forgery probability

Ferguson criticizes GCM for not ensuring that the probability of a successful forgery is close to 2^{-t}, when used with a t-bit tag; that is, it does not approximate an ideal MAC. This criticism is unfair. The security of a MAC may strongly depend on other factors, such as the number of messages that are protected. AES CBC-MAC with a 128-bit tag is far from an ideal MAC when a number of messages anywhere close to 2^{64} have been processed [5]. This there are many message authentication codes in the literature which deviate significantly from an ideal MAC, including many of the MACs based on universal hashing [6].

We discuss the special case of short authentication tags in the next two subsections.

2.2 Multiple forgery attacks

Ferguson's "second weakness" relies on leveraging a single forgery into multiple forgeries. Security against multiple forgery attacks is a worthwhile goal, but is one that has received little attention so far. Typically, MAC security analyses consider the probability of a single forgery, which bounds the probability of multiple forgeries.

In related work, McGrew and Fluhrer explored the security of GCM, HMAC, and CBC-MAC against multiple forgery attacks [7]. They described multiple forgery attacks, and then analyzed the expected number of forgeries that the attacks produce. They show that HMAC and AES CBC-MAC are actually *worse* in this regard than GCM, for typical usage scenarios (e.g. with 96-bit tags). With an ideal MAC, the expected num-

ber of forgeries grows linearly with the number of forgery attempts. With GCM, the growth is quadratic. With HMAC and CBC-MAC, it is cubic.

When short authentication tags are used, HMAC and CBC-MAC have a smaller expected number of forgeries than GCM. These MACs better approximate an ideal MAC at short tag lengths, as Ferguson pointed out. However, we stress that the relative vulnerability of GCM to multiple forgery attacks at short tag lengths does *not* indicate some fundamental weakness that can be exploited at longer tag lengths.

2.3 Short tags

The only applications where the multiple-forgery weakness is a significant problem are those for which it is acceptable to have occasional forgeries, but it is unacceptable to have a long, continuous run of forgeries. This is the crux of the issue raised by Ferguson. However, very few applications will tolerate occasional forgeries.

While Ferguson asserts that "there are many situations in which short authentication tags are used," we find the opposite to be the case. General purpose protocols strongly favor longer tags. The IPsec, TLS, and SSH standards all have mandatory-to-implement message authentication codes of length 96 bits or more. IPsec AH has a default length of 96 bits. IPsec ESP XCBC-MAC uses only 96 bit tags, and ESP CCM uses 64,96, or 128 bit tags. TLS uses HMAC-MD5 and HMAC-SHA1 without truncation, so the tags are at least 128 bits in length.

In the domains where GCM is already specified for use, there is no demand for short tags. In the case of the draft 802.1ae link-level security standard, GCM is specified with 128-bit tags mandatory, with no facilities for truncation beyond the definition of another cipher suite. Similarly, in the case of IPsec ESP GCM, the mandatory-to-implement tag size is 128 bits, and truncation is optional, with 64 bits being the smallest allowed tag.

The only valid motivation we have ever encountered for performing tag truncation is to conserve bandwidth, particularly in bandwidth-constrained packet networks[1]. The only type of application for which a short tag provides an adequate level of security, in our experience, is one in which the receiver is essentially stateless. Several audio decoders fit this description. Some conversational voice systems have this motivation and can tolerate low authentication strength. These systems are clearly the exception, rather than the rule, and we cannot find these conditions in any other systems.

The only general purpose protocol intended to be used with short authentication tags, of which we are aware, is the Secure Real-time Transport Protocol (SRTP) [8], which is used to provide security on voice over IP. Interestingly, it was designed so that it can be used with message authentication algorithms based on universal hashing, for which multiple forgeries may be a concern. In particular, SRTP supports frequent changes of the cipher and MAC keys ([8, Section 4.3]). This facility can be used to limit the number of multiple forgeries that are possible, even when the tags are short. However, the default MAC for SRTP is HMAC-SHA1, the default tag length is 80 bits, and the defaults are strongly encouraged.

[1]For some systems, such as disk-block encryption, it is accepted practice to not use authentication at all. We restrict ourselves to considering systems in which message authentication is used.

Certainly, in typical desktop environments, there is no clear motive for any sort of tag truncation. In the high-speed environments that drove the design of CWC and GCM, there is absolutely no need, due to the large available bandwidth and comparitively large message sizes.

2.4 Motivation

GCM was designed in response to feedback from hardware engineers who found the cost and scalability of other solutions wanting. One of the primary methods for achieving lower cost, higher scalability hardware implementations was the use of additive encryption of the hash output. This choice is in contrast to the ECB-mode encryption, as is used by CWC. This design allows maximum performance to be achieved with a single AES pipeline on chip, even on packetized data, by avoiding pipeline stalls. This property allows GCM to have a top speed about twice that of CWC [4, Section 3.1] on typical Internet data. We are well aware that this choice implies worse security for very short tags, but we continue to believe that this is a worthy tradeoff.

3 Other Comments

Ferguson also made several comments on GCM's design. We consider each below.

3.1 Plaintext size

GCM limits plaintext sizes to approximately 64GB. This enables it to work in any foreseeable packet network; it is able to handle any IPv6 jumbogram, even though we cannot anticipate a time where individual network packets will ever reach the maximum size of a jumbogram. Additionally, in a network protocol, GCM can be used when individual messages do exceed 64GB by fragmenting the messages prior to encryption.

One of the core elements of GCM is counter mode encryption. In the case of the 96-bit IV (which we expect to be the primary way in which GCM is used, as is bearing out in 802.1ae and IPsec), the plaintext has bits reserved to encode the block number uniquely for each message, and then bits that are left over become the nonce. If we were to allow for larger messages, we would do so at expense of the size of the nonce, thus reducing the number of messages that could be protected by a single key. The nonce length of 96 bits was barely enough to meet the needs of 802.1ae. Any mode built upon counter mode will have a similar tradeoff.

3.2 Bit alignment

Ferguson is critical of the fact that the GCM specification shows how GCM can be used on arbitrary-length bit strings. He states that implementations only allowing for byte-aligned inputs "cannot claim to implement full GCM". This restriction is certainly not what we intended when we wrote the specification. An implementation that only handles byte-aligned strings is still implementing GCM. We clarified this point in the updated specification.

It is interesting that the Counter and CBC-MAC Mode (CCM) of operation [9] was criticized for *not* handling arbitrary-length bit strings! The prominent point of the critique [10] was that "cryptographic algorithms reach beyond technological conventions like the prevalence [of] byte-orientation in computing systems. To put things in perspective: most cryptographers would have viewed it as a poor choice if MD4, MD5 and SHA1 had only been defined on octet strings. It is no less a defect if a general-purpose AEAD scheme is only defined on octet strings"

We do not know of any application domain that would ever adopt strings that are not byte-aligned. However, if there are domains where there is a good reason to do so, or if such domains emerge in the future, then GCM can meet their needs.

3.3 Software performance

Ferguson complains about the the large variety of implementation tradeoffs that can be made when implementing multiplication over the the finite field $GF(2^{128})$. However, it is clear that performance is not a serious issue. Even if a GCM implementation were so slow at to run at 100 cycles per byte, a typical low-end desktop CPU could keep up on a 100 Megabit link [4, Section 3.2].

In practice, it is probably best to use tables that will fit alongside the AES context comfortably in cache while encrypting a single message. 256-byte tables are more than acceptable for general-purpose use, and are not much larger than a single AES-128 expanded key, which occupies 160 bytes, and are smaller than a single AES-256 expanded key, which takes up 320 bytes. 4K tables also seem reasonable for any modern desktop architecture, and we expect there will be some implementations that use them instead of the 256-byte tables.

An optimized CWC-like strategy may outperform GCM in software on some CPUs. However, this software advantage will be slight, since GCM's hash is already significantly faster than AES in software, and both GCM and CWC use counter mode encryption. However, this slight software advantage would come at a significant expense in hardware performance, thus undermining the main method for achieving high-speed implementations. For example, [11] describes common hardware architecture issues for such modes, and identifies the CWC hash function as the primary performance bottleneck.

Our published GCM timing numbers were presented on a G4 due to my lack of direct access to a P4. Brian Gladman's numbers certainly show GCM performing in the same neighborhood on P3s and P4s as it does on a G4, and our tests on AMD Athlon hardware shows similar performance there, even for 256-bit tables.

As we look at desktops attempting to handle connections at gigabit speeds, it's clear that if we want to protect data on the wire but still realize the full performance of our network hardware, we will need to either increase clock speeds even faster than we have been, or push our symmetric crypto primititives into hardware. GCM makes it easier and more cost effective to do so than any other mode of operation, including CWC.

3.4 IV collisions

Ferguson notes that for IVs with lengths other than 96 bits, GCM uses the GHASH function to process the IV, to produce the initial counter used for the counter mode encryption. It is true that this IV processing step introduces a risk of collisions. However, Ferguson incorrectly states that collisions will occur after GCM has processed 2^{64} blocks of data. Rather, collisions are likely after that number of messages has been processed. At this point, any AES mode of operation is being driven past the point that its warranty has expired. At this point, CBC-MAC is vulnerable to multiple forgery attacks, as described in [7]. It is a valid point that the recovery of the GHASH key makes selective forgeries easier against GCM than they would be against CBC-MAC. However, users who want to avoid this increased exposure can do so by not using IVs with lengths other than 96 bits, or by limiting the number of times that they do so.

The ability to accommodate variable-length IVs helps GCM "adapt to the widely different circumstances in which ciphers are used," to borrow a phrase from Ferguson, by loosening the restrictions on the IV. It can be used in place of key derivation, obviating the need to derive additional keys. It may help security by providing users with a way to safely re-use an existing key to protect an additional channel. However, these benefits do come at a slight cost, in the increased exposure due to potential counter collisions. We feel that this tradeoff is worthwhile, especially considering that users can opt to avoid variable-length IVs altogether.

3.5 Tag length

Ferguson correctly points out that the authentication tag length must be fixed for each key, as described in [4] but not [1]. The updated specification makes this clear.

References

[1] D. McGrew and J. Viega. The Galois/Counter Mode of Operation (GCM), *Submission to NIST Modes of Operation Process*, January 15, 2004.

[2] D. McGrew and J. Viega. The Galois/Counter Mode of Operation (GCM), *Updated submission to NIST Modes of Operation Process*, May 31, 2005.

[3] N. Ferguson, Authentication weaknesses in GCM. *Comments submitted to NIST Modes of Operation Process*, May 20, 2005.

[4] D. McGrew and J. Viega. The Security and Performance of the Galois/Counter Mode (GCM) of operation. *Proceedings of Indocrypt '04*, 2004. Full version online at `http://eprint.iacr.org/2004/193`.

[5] 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).

[6] M. Wegman and L. Carter. *New hash functions and their use in authentication and set equality*. Journal of Computer and System Sciences, 22:265279, 1981.

[7] D. McGrew, S. Fluhrer, Multiple forgery attacks against Message Authentication Codes. *Prepublication manuscript*, May, 2005. Also submitted to NIST Modes of Operation Process.

[8] M. Baugher, D. McGrew, M. Naslund, E. Carrara, K. Norrman, The Secure Real-time Transport Protocol (SRTP), *IETF Request for Comments*, RFC 3711, March, 2004.

[9] D. Whiting, N. Ferguson, and R. Housley. *Counter with CBC-MAC (CCM)*. Submission to NIST, 2002. Available online at `http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/`.

[10] Mihir Bellare, Phillip Rogaway, and David Wagner. The EAX Mode of Operation (A Two-Pass Authenticated Encryption Scheme Optimized for Simplicity and Efficiency). Fast Software Encryption (FSE), Lecture Notes in Computer Science, vol. 3017, pp. 389-407, 2004.

[11] Bo Yang, Sambit Mishra, and Ramesh Karri. High Speed Architecture for Galois/Counter Mode of Operation (GCM). Cryptology ePrint Archive, Report 2005-156, May 2005. `http://eprint.iacr.org/2005/146`