AES/GCM Cryptography

A layman's research into the matter

Author

Martin Andersson (webmaster@martinandersson.com)

Useful links

Example and bug related to AES/GCM:
http://bugs.java.com/bugdatabase/view_bug.do?bug_id=8016249

Also see:
http://blog.philippheckel.com/2014/03/01/cipherinputstream-for-aead-modes-is-broken-in-jdk7-gcm/

However, I cannot reproduce this bug using the SunJCE provider and Java 1.8.0_25. See for yourself by giving the proof-of-concept GitHub project a test run.

Problems with the default SunJCE's internal buffering:
http://stackoverflow.com/q/26920906/1268003

User can only reach 3 MB/s throughput using AES/GCM:
http://stackoverflow.com/questions/25992131/slow-aes-gcm-encryption-and-decryption-with-java-8u20

Example of IV creation:
http://stackoverflow.com/a/24718746/1268003

Checking tag and associated data (more on this topic next):
http://stackoverflow.com/q/23933582/1268003

Is the “associated data” (AD) required or optional?

Says optional (last paragraph says AD may be of zero length):

> Authenticated encryption [BN00] is a form of encryption that, in addition to providing confidentiality for the plaintext that is encrypted, provides a way to check its integrity and authenticity. Authenticated Encryption with Associated Data, or AEAD [R02], adds the ability to check the integrity and authenticity of some Associated Data (AD), also called "additional authenticated data", that is not encrypted.
>
> [..]
>
> The associated data A is used to protect information that needs to be authenticated, but does not need to be kept confidential. When using an AEAD to secure a network protocol, for example, this input could include addresses, ports, sequence numbers, protocol version numbers, and other fields that indicate how the plaintext or ciphertext should be handled, forwarded, or processed. In many situations, it is desirable to authenticate these fields, though they must be left in the clear to allow the network or system to function properly. When this data is included in the input A, authentication is provided without copying the data into the plaintext.
>
> Each AEAD algorithm MUST accept any associated data with a length between zero and A_MAX octets, inclusive, where the value A_MAX is specific to that algorithm.
>
> Source: http://www.ietf.org/rfc/rfc5116.txt

JavaDoc of Cipher says:

> Modes such as Authenticated Encryption with Associated Data (AEAD) provide authenticity assurances for both confidential data and Additional Associated Data (AAD) that is not encrypted. [..] Both confidential and AAD data can be used when calculating the authentication tag (similar to a Mac). This tag is appended to the ciphertext during encryption, and is verified on decryption.
>
> AEAD modes such as GCM/CCM perform all AAD authenticity calculations before starting the ciphertext authenticity calculations.
>
> Source: http://docs.oracle.com/javase/8/docs/api/javax/crypto/Cipher.html

A is "only authenticated" and "not encrypted":

> A is data which is only authenticated (not encrypted)
>
> Source: http://en.wikipedia.org/wiki/Galois/Counter_Mode

Verdict: AD is not required for message authentication of the encrypted cipher text. AD is optional if one need to also send unencrypted metadata and have that data be authenticated as well. This is also demonstrated by my proof-of-concept application that does not use AD.

Can AES/GCM be used as a stream cipher?

I think next quote talks specifically about CTR, but it still says that "counter mode":

> [..] turns a block cipher into a stream cipher. It generates the next keystream block by encrypting successive values of a "counter".
>
> Source: http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Counter_.28CTR.29

and..

> As the name suggests, GCM mode combines the well-known counter mode of encryption with the new Galois mode of authentication.
>
> Source: http://en.wikipedia.org/wiki/Galois/Counter_Mode

Says GCM "uses" counter mode:

> I would recommend using GCM mode encryption. It is included in the latest JDK (1.7) by default. It uses a counter mode encryption (a stream cipher, no padding required) and adds an authentication tag.
>
> Source: http://stackoverflow.com/a/9574553/1268003

Is a stream cipher:

> GCM is counter mode with authentication. All the rules for Counter Mode still apply. Counter mode effectively turns a block cipher into a stream cipher.
>
> Source: http://stackoverflow.com/a/5695328/1268003

JavaDoc of Cipher says:

> [..] block ciphers can be turned into byte-oriented stream ciphers by using an 8 bit mode such as CFB8 or OFB8.
>
> Source: http://docs.oracle.com/javase/8/docs/api/javax/crypto/Cipher.html

So if AES can be used with CFB and OFB to produce a cipher stream, should it not be the same for GCM?

GCM is "ideal" for "packetized data" (i.e, not a stream of bits):

> GCM is ideal for protecting packetized data, because it has minimum latency and minimum operation overhead.
>
> Source: http://en.wikipedia.org/wiki/Galois/Counter_Mode

Interesting answer:

> [..] stream ciphers such as GCM [..].
>
> Source: http://stackoverflow.com/a/15513133/1268003

Verdict: Yes it can, which my proof-of-concept application show that stream file bits.

Must the IV be unique?

JavaDoc of Cipher says:

> Note that GCM mode has a uniqueness requirement on IVs used in encryption with a given key. When IVs are repeated for GCM encryption, such usages are subject to forgery attacks. Thus, after each encryption operation using GCM mode, callers should re-initialize the cipher objects with GCM parameters which has a different IV value.
>
> Source: http://docs.oracle.com/javase/8/docs/api/javax/crypto/Cipher.html

Okay after "each encryption operation". Sure, Wikipedia says about the same:

> [..] security depends on choosing a unique initialization vector for every encryption performed with the same key (see stream cipher attack). For any given key and initialization vector combination, GCM is limited to encrypting 2^39 - 256 bits of plain text.
>
> Source: http://en.wikipedia.org/wiki/Galois/Counter_Mode

But the provided numbers is still equivalent to 68.72 gig plaintext:

> Source: http://www.wolframalpha.com/input/?i=%28%282%5E39%29-256%29+bits+to+gigabytes

This might shed some light:

> Thus if at any point you reuse PRNG stream then it maybe possible to obtain plain text with a simple XOR. Its important to note that the same Key+Nonce will always produce the same PRNG stream. In a protocol the same Key+Nonce can be used for the life of

> the session, so long as the mode's counter doesn't wrap (int overflow). An example protocol you could have two parties and they have a pre-shared secret key, then they could negotiate a new cryptographic Nonce for each session (Remember nonce means use ONLY ONCE).
>
> Source: http://stackoverflow.com/a/5695328/1268003

Verdict: The symmetric key generated by SRP (Secure Remote Protocol) is random and for my use case which brought me into this investigation, no data sent during one session will ever go past 68.72 gig. But, do the IV need to be unique? What if an IV has already been used with another key in another session, perhaps by another user? I'm not certain. Given that the key is random, and my IV will be a hash of that, I'm not going to bother about the uniqueness constraint. Also see the AesGcmCipher class used in the proof-of-concept application for some more details related to IV.