Key Recovery Attacks on AES-GCM-SIV – RESPONSE

Shay Gueron and Yehuda Lindell

We greatly thank you for your interest in the AES-GCM-SIV authenticated encryption scheme, and for your work on analyzing it.

We begin with the *first attack*. It is well known that in the model of multi-key security, if there are M different users with M different keys, and the key-space is of size N, then it is possible to find one of the keys in time M/N. We cast the attacks you describe in this light. Specifically, since our CFRG essentially changes the keys in every encryption, we obtain a multi-key setting even for a single user. Considering, for example, attack number 1: this attack works since it is possible to obtain encryptions of a single known block under many keys. Thus, the original multi-key attack of [1] is applicable.

Having said the above, we find that the tradeoff between the CFRG and original GCM-SIV paper to be favorable toward the CFRG. This is because as with all counter modes, the original GCM-SIV and GCM with a random nonce (counter) has the property that after 2^{32} different encryptions, the probability of a collision on the nonce (counter) is about 2^{-32}. Now, if less than 2^{32} encryptions are carried out, then neither the multi-key attacks nor the collision on the counter is a concern. However, if more than 2^{32} encryptions are computed, then the amount of work required for the multi-key attacks is still extremely large, whereas the probability of a collision on the nonce becomes a very real concern.

We do not dismiss the threat of multi-key attacks, and it has been recently noted that this may be a real problem in TLS sessions. This has prompted recent work, one example being [2]. Indeed, [2] explain that in TLS it is preferable to use a random nonce for GCM even though a fresh key is used in every session (and thus one could always start with a fixed nonce of 0). This is exactly to prevent/mitigate multi-key attacks.

We stand behind our proposal to change keys in every message, and were surprised that changing the nonce does not solve the problem (Ref. [1] in your document). We then noticed that in the original GCM-SIV paper appearing in ACM CCS, we XOR the nonce with the result of POLYHASH on the AAD and message, and then encrypt it (using AES). While our intention was to mount some nonce-based key derivation on top of GCM-SIV, we forgot (in one of the versions of the proposal) this XOR in the CFRG specification. By returning this XOR, we mitigate the risk of multi-key attacks since different nonces would now not enable an attacker to obtain many encryptions of the same (known) block under different keys.

Regarding the other attacks:

Attack 3 requires 2^{128} chosen plaintexts. We do not consider this to be realistic. Furthermore, when the block size is 128 bits then everything breaks well before that (with repeating inputs to AES).

Attack 2 is just a type of time-space tradeoff and these types of bounds are always possible on block ciphers. Note also that in order to reduce the work time by just two bits, the number of queries required is already 2^{96}. As stated regarding attack 3, with a 128-bit block size everything breaks well before this anyway.

Once again, we thank you for your interest in GCM-SIV, and we will make the change to mitigate the first attack as described. Although we think that users of GCM-SIV should not be concerned with the other attacks, as described above, we appreciate the feedback and we will document them for full transparency.

[1] E. Biham. How to Forge DES-Encrypted Messages in 2^{28} Steps. *Information Processing Letters*, 84(3):117-124, 2002.

[2] M. Bellare and B. Tackmann. The Multi-user Security of Authenticated Encryption: AES-GCM in TLS 1.3. In *CRYPTO 2016*, Springer (LNCS 9814), pages 247-276, 2016.