Key commitment and Multi Key Collision Attacks

Uri Ariel Yahalom* Shay Gueron†

Abstract

In this report we examine key commitment in Authenticated encryption and vulnerabilities that arise when AEAD is misuded and robustness is assumed. Recent research shows new class of attacks such as partitioning oracles which arise when AE schemes that are not committing with respect to their keys are misused. As key commitment is not part of AE's design goal, AE schemes in general do not satisfy it.
We show an adaptive chosen ciphertext attack that utilizes key multi-collision against widely used AEAD schemes, including AES-GCM, XSalsa20/Poly1305, and ChaCha20/Poly1305, to build a partitioning oracle.
We also discuss an early implementation of the OPAQUE protocol (a protocol for password-based key exchange) that is vulnerable to partitioning oracle.

*Department of Computer Science, University of Haifa.
†Department of Management, University of Haifa.

1 Background

Authenticated Encryption

Authenticated Encryption (AE)[McGrew, 2008] is a well-studied primitive which avoids the pitfalls of unauthenticated Symmetric-key encryption with relatively small performance overhead AE schemes are used in widely adopted protocols and are the default Symmetric option in modern cryptographic libraries.
The design of encryption historically separated the goals of confidentiality and authenticity, which led to widespread deployment of encryption schemes vulnerable to chosen ciphertext attacks (CCA).
Subsequently, researchers showed how to exploit CCAs to recover plaintext data, most notably via padding and format oracle attacks As a result, cryptographers now advocate the use of authenticated encryption with associated data (AEAD) schemes and CCA-secure public key encryption.
There has since been a shift to adopt fast CCA-secure schemes, notably AES-GCM, and XSalsa20/Poly1305.

AES-GCM

AES-GCM is an AEAD scheme that composes AES in counter mode with a special MAC that uses an XOR-universal hash function called GHASH. Encryption takes in a nonce N, an AES key K, associated data AD, and plaintext $M_1, \ldots, M_n$ without loss of generality (AES-GCM specification handles various length). and it outputs a ciphertext $C_1, \ldots, C_n$ and an authentication tag T. The ciphertext blocks $C_1, \ldots, C_n$ are generated using counter mode with E, and the tag T is computed by applying GHASH to AD and $C_1, \ldots, C_n$. Decryption also computes the tag, compares it with T, and, if successful, outputs the counter-mode decryption of the ciphertext blocks. For further detailes please revise the original papers.[McGrew and Viega, 2004]

GHASH

We will omit associated data for simplicity.
For a key K, GHASH first derives a hash key $H = E_k(0^n)$. It then hashes by computing $h = C_1 \cdot H^{m+1} \oplus \cdots \oplus C_{m-1} \cdot H^3 \oplus C_m^* \cdot H^2 \oplus L \cdot H$ where C_m^* is C_m concatenated with enough zeros to get an n-bit string and L is an n-bit encoding of the length of the message. The maximum plaintext length is $2^{39} - 256$. The

multiplications are performed over the finite field GF(2^{128}) with a particular fixed irreducible polynomial.

2 Key Commitment

Key commitment guarantees that a ciphertext can only be decrypted under the same key it was encrypted with. If we are to find a ciphertext that decrypts into two valid plaintexts under two different keys do not commit with respect to the key.
Key commitment was initially studied by [Farshim et al., 2017] and while it might seem like an academic pursuit, [Dodis et al., 2018] and [Grubbs et al., 2017] show how to exploit AEADs that do not commit to their key. Nevertheless, AEAD key commitment has not received the required attention and can be overlooked in deployment.
A committing encryption scheme is a scheme for which it is computationally unfeasible to find two keys and a ciphertext that decrypts under both. Security standards for committing AEADs were first formalized by[Farshim et al., 2017].

For detailed and formalized explanation about key commitment please reffer to [Gueron, 2020].

3 Partitioning Oracles

Thus far, key commitment has not been considered an essential security goal for most cryptographic applications. This is perhaps because attacks exploiting key collision have arisen in relatively niche applications like message franking for encrypted messages.

We consider settings in which an attacker seeks to recover a secret $pw \in D$ from some set of possible values D. The attacker has access to an interface that takes as input a bit string V, and uses it and pw to output the result of some boolean function $f_{pw} : \{0,1\}^* \rightarrow \{0,1\}$. Here f_{pw} is an abstraction of some cryptographic operations that may succeed or fail depending on pw and V . We use $f_{pw}(V) = 1$ for success and $f_{pw}(V) = 0$ for failure.
Given oracle access to adaptively query fpw on chosen values, the question is: Can an attacker efficiently recover pw? This of course will depend on f. We refer to f as a partitioning oracle if it is computationally tractable for an adversary, given any set $S \subseteq D$, to compute a value $\hat{V}$ that partitions S into two sets S^* and $S \setminus S^*$ with $|S^*| \leq |S \setminus S^*|$, such that $f(pw, \hat{V}) = 1$ for all $pw \in S$ and $f(pw, \hat{V}) = 0$ for all $pw \in S \setminus S^*$. We call such a $\hat{V}$ a splitting value and refer to $k = |S^*|$ as the degree of a splitting value $\hat{V}$.

For most f_{pw} of practical interest it will be trivial to compute splitting values with degree $k = 1$. In this case, a partitioning oracle attack coincides with a traditional online brute-force guessing strategy for recovering pw.
Partitioning oracles become more interesting when we can efficiently build splitting values of degree $k > 1$. In this case, we can perform a simple adaptive binary search for pw if we can compute splitting values of degree up to $k = \lceil d/2 \rceil$. Initially set $S = D$ and compute a value $\hat{V}_1$ that splits S into two halves of roughly the same size. Query $f_{pw}(\hat{V}_1)$ to learn which half of D the value pw lies within and recurse on that half. Like all binary searches this provides an exponential speed-up over the brute-force because we can recover pw in $\lceil \log d \rceil$ queries. We provide more details about this attack later in this report.

A partitioning oracle arises when an attacker can:

1. efficiently prepare ciphertexts that will decrypt under a large number of keys
2. submit those ciphertexts to an oracle that tells whether decryption succeeds or fails.

This enables gaining information about the password.
We will use the following algorithm to perform an attack on the encryption scheme E.

Data: oracle O
Result: The secret key K that is used by the encryption algorithm
initialization;
set N = the space of all possible keys.
set k = the partitioning of the set that we will use.
while $|N| > 1$ **do**
  divide N into two equal sets and choose one of them.
  set $K = choose(N/k)$.
  prepare a set of ciphertext C that will decrypt under all of the keys $K_1, \ldots, K_k$ in K, and the encryption scheme E.
  if $O(C) == 1$ *(the oracles successfully decrypts C)* **then**
    set $N = K$.
  else
    set $N = N/K$.
  end
end

Algorithm 1: Partitioning Oracle Attack

We will be limited by the size of the ciphertext C that we can input to the oracle since a ciphertext that we will find that will decrypt under k keys will be roughly of size k^2.
Hence this attack is mostly useful against misused AEAD in schemes that draw keys from a small key space e.g. password based key derivation schemes.

4 Multi Key Collision Attacks

Multi Key Collision Attacks is an attack where we find a ciphertext - C that decrypts under k different keys $K_1, \ldots, K_k$ to k different valid plaintexts $P_1, \ldots, P_k$.
In practice, to perform a multi key collision attack against AES-GCM we need to solve a system of linear equations. This is possible because of the algebraic properties of the universal hashing in AES-GCM. To successfully mount such an attack we will need an oracle that will be telling us whether decryption fails or succeeds.
We formalize our cryptanalytic goal as follows:
Let $AEAD = (AuthEnc, AuthDec)$ be an AEAD scheme, and let its key space be the set K. We write encryption $AuthEncK(N, AD, M)$ to denote running the encryption algorithm with secret key $K \in K$, nonce N (a bit string), associated data AD (a bit string), and message M (a bit string). Decryption is written analogously, as $AuthDecK(N, AD, C)$ where C is a ciphertext. Our attacks will require that decryption fails for $K \notin K$. We require of our AEAD scheme that $AuthDecK(N, AD, AuthEncK(N, AD, M)) = M$ for all N, AD, M not exceeding the scheme's length restrictions.
At a high level, our multi-collision attack against AES-GCM reduces the task of finding key multi-collisions to solving a system of linear equations. This is possible because of the algebraic properties of the universal hashing underlying integrity protection in AES-GCM[McGrew and Viega, 2004].
For a set $K = K_1, \ldots, K_k$ and nonce N, one can compute a single ciphertext $(C_1, \ldots, C_{k-1}, T)$ that decrypts correctly under every key in K. For each K_i, derive the associated GHASH key $H_i = E_{K_i}(0^n)$ and then construct the linear equation.

$$T = C_1 \cdot H_i^{k-1} \oplus \cdots \oplus C_{k-1} \cdot H_i^2 \oplus L \cdot H_i \oplus E_{K_i}(N \,\|\, 0^{31}1)$$

which one arrives at by assigning H_i to H and then substituting the result into the equation $T = h \oplus E_{K_i}(N||0^{31}1)$. Note that we have fixed the number of the ciphertext blocks to be k-1. The resulting system of k equations in k unknowns will be:

$$\begin{bmatrix} 1 & H_1^2 & H_1^3 & \cdots & H_1^{k+1} \\ 1 & H_2^2 & H_2^3 & \cdots & H_2^{k+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & H_k^2 & H_k^3 & \cdots & H_k^{k+1} \end{bmatrix} \cdot \begin{bmatrix} T \\ C_{k-1} \\ \vdots \\ C_1 \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_k \end{bmatrix}$$

where $B_i = (L \cdot H_i) \oplus E_{K_i}(N||0^{31}1)$. We can optimize this system of equations and transform it to a system which can be solved in time $O(k^2)$ and space $O(k)$.

Please refer to [Len et al., 2020] for details.

For such a system to be solvable, there have to be rows that are linear combinations of other rows. Since each row is increasing powers of a random field element (i.e., the hash key) this dependence between rows should be rare as long as the block cipher acts like an ideal cipher. For explanations please refer to [Len et al., 2020]. an example of key multi collision algorithm on the AES-GCM scheme:

Multi-Collide-GCM$(\mathbb{K}, N, T)$:

$L \leftarrow \mathsf{encode}_{64}(0) \,\|\, \mathsf{encode}_{64}(|\mathbb{K}| \times 128)$
$\mathbf{pairs}[\cdot] \leftarrow \perp$; $C \leftarrow \varepsilon$
For $i = 1$ to $|\mathbb{K}|$:
    $H \leftarrow E_{\mathbb{K}[i]}(0^{128})$; $P \leftarrow E_{\mathbb{K}[i]}(N \| 0^{31}1)$
    $y \leftarrow ((L \cdot H) \oplus P \oplus T) \cdot H^{-2}$
    $\mathbf{pairs}[i] \leftarrow (H, y)$
$f \leftarrow \mathsf{Interpolate}(\mathbf{pairs})$; $\mathbf{x} \leftarrow \mathsf{Coeffs}(f)$
For $i = 1$ to $|\mathbb{K}|$:
    $C \leftarrow C \,\|\, \mathbf{x}[i]$
Return $N \,\|\, C \,\|\, T$

5 Our contribution

Multi Key Collision Attacks is an attack where we find a ciphertext - C that decrypts under k different keys.
In practice, to perform a multi key collision attack against AES-GCM we need to solve a system of linear equations. This is possible because of the algebraic properties of the universal hashing in AES-GCM.
We implemented a solver for such a system of linear equations for 2 keys, 3 keys, and an arbitrary constraints.

Key count	Time	Ciphertext size	Ciphertext size/Key count
2	0.04199	4	2
4	0.0934324	6	1.5
8	0.0628430	10	1.25
16	0.1656253	18	1.125
64	0.708905	66	1.03125
256	7.526772	258	1.0078125
1024	46.03521	1026	1.001953125
4096	996.350043	4098	1.00048828125

From the above table we can clearly see that the resulting ciphertext is linear in the size of the key space. and in fact are exactly the key count and two additional blocks.
The time to solve this linear equations without parallel optimizations is around polynomial third degree time. The main cryptanalytic step for our attacks is constructing (what we call) key multi-collisions, in which a single AEAD ciphertext can be built such that decryption succeeds under some number k of keys.
We build key multi-collision ciphertexts of length $O(k)$ in $O(k^2)$ time using polynomial interpolation from off-the-shelf libraries, making them reasonably scalable even to large k. We can further optimize to get $O(k \log^2 k)$ running time using a different polynomial interpolation technique[Borodin and Moenck, 1974].
Given access to an oracle that reveals whether decryption succeeds, our key multi-collisions for AES-GCM enables a partitioning oracle attack that recovers the secret key k in roughly $m + \log k$ queries in situations where possible keys fall in a set of size $d = m \cdot k$.

6 Future Work

In the future we plan to build a partitioning oracle on one the rust implementation of OPAQUE[Gustin, 2019] using the described multi key collision attack on AES-GCM.
Another interesting question that raises is to whether quantum computing will allow efficiently solving very large systems of equations thus maybe extending this attack to large key spaces, and rendering AEADs fully vulnerable to partitioning oracles.

References

[Borodin and Moenck, 1974] Borodin, A. and Moenck, R. (1974). Fast modular transforms. *Journal of Computer and System Sciences*, 8(3):366–386.

[Dodis et al., 2018] Dodis, Y., Grubbs, P., Ristenpart, T., and Woodage, J. (2018). *Fast Message Franking: From Invisible Salamanders to Encryptment*, pages 155–186. Lecture Notes in Computer Science. Springer.

[Farshim et al., 2017] Farshim, P., Orlandi, C., and Rosie, R. (2017). Security of symmetric primitives under incorrect usage of keys. *IACR Cryptol. ePrint Arch.*, 2017:288.

[Grubbs et al., 2017] Grubbs, P., Lu, J., and Ristenpart, T. (2017). Message franking via committing authenticated encryption. Cryptology ePrint Archive, Report 2017/664. https://eprint.iacr.org/2017/664.

[Gueron, 2020] Gueron, S. (2020). Key committing aeads. Cryptology ePrint Archive, Report 2020/1153. https://eprint.iacr.org/2020/1153.

[Gustin, 2019] Gustin (2019). opaque. https://github.com/gustin/opaque.

[Len et al., 2020] Len, J., Grubbs, P., and Ristenpart, T. (2020). Partitioning oracle attacks. Cryptology ePrint Archive, Report 2020/1491. https://eprint.iacr.org/2020/1491.

[McGrew, 2008] McGrew, D. (2008). An Interface and Algorithms for Authenticated Encryption. RFC 5116.

[McGrew and Viega, 2004] McGrew, D. A. and Viega, J. (2004). The security and performance of the galois/counter mode of operation (full version). Cryptology ePrint Archive, Report 2004/193. https://eprint.iacr.org/2004/193.