Article

Key Management Systems at the Cloud Scale

Matthew Campagna ^{1,†} and Shay Gueron ^{1,2,*,†}¹ Amazon Web Services Inc., Seattle, WA 98101, USA² Department of Mathematics, University of Haifa, Haifa 3498838, Israel

* Correspondence: shay@math.haifa.ac.il

† These authors contributed equally to this work.

Received: 14 July 2019; Accepted: 28 August 2019; Published: 5 September 2019

Abstract: This paper describes a cloud-scale encryption system. It discusses the constraints that shaped the design of Amazon Web Services' Key Management Service, and in particular, the challenges that arise from using a standard mode of operation such as AES-GCM while safely supporting huge amounts of encrypted data that is (simultaneously) generated and consumed by a huge number of users employing different keys. We describe a new derived-key mode that is designed for this multi-user-multi-key scenario typical at the cloud scale. Analyzing the resulting security bounds of this model illustrates its applicability for our setting. This mode is already deployed as the default mode of operation for the AWS key management service.

Keywords: AES-GCM; cloud computing; key management

1. Introduction

Key management for a public cloud is based on the promise for availability, durability, and absolute security and privacy. This requires the following: (a) adherence to best known cryptographic practices and standards, in a highly available low-latency service; (b) the ability to provide (semantic) security while handling extremely large volumes of requests that are placed by an extremely large number of users that utilize different keys (i.e., a multi-user–multi-key scenario). The adversary is assumed to have a user's capability (and credentials) and can eavesdrop and also make forgery attempts. We call such an encryption system that satisfies these requirements a “cloud-scale encryption system” (CES). This paper examines the requirements for a CES against some engineering decisions that were made in deploying the real-world solution called Amazon Web Services (AWS) Key Management Service [1]. It describes an enhancement of an encryption mode that is used by the implementation, and demonstrates security bounds that surpass the initially designed properties of the underlying model, under the cloud's multi-user–multi-key scenario.

An encryption service with high availability and durability requires redundancy, and therefore state management across redundant nodes. State replication introduces unwanted latency for many applications requiring consistency. To the extent feasible, state replication should be minimized across redundant nodes, especially for functions that are sensitive to latency. A CES should be able to handle simultaneous calls to encrypt and decrypt, without resource contention, noticing that encryption/decryption under a master key is a latency-sensitive functionality.

A fundamental decision of the AWS KMS design is to allocate a user-specific master key (CMK) as the root node for a user (customer). For operational and security reasons, the service provides only a limited set of data-plane operations using this master key, including encryption and decryption. Obviously, an adequate symmetric authenticated encryption with associated data (AEAD) mode of operation is a critical building block for the service, and AES-GCM [2] is a natural selection for such a primitive. It is a (NIST) standard model that enjoys overall acceptance, security proofs, and excellent

performance with side-channel resistance when running on modern platforms (that have AES-NI and PCLMULQDQ instructions) which are the target platforms in our context. But is AES-GCM indeed suitable at the cloud-scale? Here are a few challenges.

The small block size of AES, 128-bits, is in contention with PRP-PRF bounds. Encryption rates within a cloud environment can easily reach the birthday bounds for 128-bit block ciphers. Furthermore, to reduce state management in the deployed system, AES-GCM needs to be used with random *IV*s. The restriction on *IV* reuse with AES-GCM impinges further on the number of unsynchronized encryptions across distributed nodes. Conforming to standardized ciphers and modes induces requirements for avoiding *IV* reuse. This imposes a need for frequent change of master keys before reaching *IV* collision probability thresholds. This is too costly and cumbersome for real-world deployment.

These difficulties motivate the AWS KMS solution to use a new mode of operation, which we call CES-GCM. This mode uses a random nonce and *IV* and applies a nonce-based key derivation before every encryption.

1.1. Related Work

The sensitivity of AES-GCM to a nonce reuse and the imposed limits on the use of a key when the *IV*s are chosen randomly are known issues. The idea of re-keying cipher schemes to extend the lifetime of a key was proposed by Abdalla and Bellare [3] and standardization of its variants are being discussed by the IETF [4]. Multi-key scenarios for ciphers and MACs, with a one-of-many-keys key recovery goal, have also been studied (e.g., [5,6]). Analysis of AES-GCM in the multi-user scenario as used for TLS 1.3, is discussed by Bellare and Tackmann [7], and also in [8].

AES-GCM-SIV [9] builds on the synthetic *IV* mechanism proposed by Rogaway and Shrimpton [10] and is in the process of standardization by the IETF [11]. This mode takes a nonce-misuse-resistant underlying scheme (GCM-SIV+) as the basis, and builds a per-nonce key derivation on top. This extends the lifetime of a key. AES-GCM-SIV is not yet standardized, but is already in use (by Google; for QUIC [12]).

Gueron and Lindell [13] described a general approach for taking any nonce based encryption scheme and prepending a nonce-based key derivation for each encryption in a way that the nonce is used for both the key derivation and the scheme itself. They showed that this extends the lifetime of a key and improves the security bounds. Such schemes lead to a multi-key situation due to the per-message key derivation. AES-GCM-SIV is a special case of this construction. A very recent study by Bose, Hoang, and Tessaro [14] discusses the behavior of AES-GCM-SIV in a multi-user case, and establishes generalized and improved bounds. In fact, the composition of the results of [13,14] gives the analysis for AES-GCM-SIV in a multi-user-multi-key scenario.

This paper describes the CES-GCM mode currently used in AWS KMS. It does not claim that CES-GCM is necessarily superior to AES-GCM-SIV. Timeline wise, initial AWS KMS deployment started in 2014, using AES-GCM, and was limited only to the “Elastic Block Storage” service. Subsequently, the difficulties described above, that emerged from scaling it to other services, required a different mode. Authenticated encryption solutions that could address the cloud-scale challenges were not known (at least not publicly) or standardized at the time. Thus, the reported work predates the excellent works of [9,13] (on AES-GCM-SIV) and of [14] (that enhanced its analysis), which were published in 2017/8. CES-GCM differs from AES-GCM-SIV in several respects, including the separation of the nonce that is used for key derivation and the *IV* used for encryption. Such separation allows for the independent generation (at different steps of the encryption) of the random nonce and *IV*, thus insulating the implementation from a single system failure in the entropy injection. We call this property “nonce-misuse-independence”. Since CES-GCM builds directly over AES-GCM, its properties are achieved while straightforwardly adhering to FIPS certification requirements (AWS KMS is FIPS certified).

1.2. Our Contribution

- We describe the challenges and considerations that need to be addressed by any design of a CES. We explain why current modes (such as AES-GCM) are not a suitable solution to the problem and a tailored mode is required. Nevertheless, a CES for the public cloud should adhere to well established cryptographic standards in order to be trusted by cloud users. This restricts the flexibility of the choices that the tailored mode may have.
- We define and analyze a mode of operation (CES-GCM) that is suitable for multi-user-multi-key large scale usages, and has nonce-misuse-independence. It builds on top of a nonce respecting mode (AES-GCM). To the best of our knowledge, this is the first multi-user-multi-key mode that is being deployed in a real cloud system.

The remainder of this paper includes Section 2 with some preliminaries and notation. Section 3 provides a detailed description of the constraints and desired properties of a CES, design choices made while building the deployed AWS KMS, and the motivation for using the CES-GCM mode. Section 4 provides a security analysis of CES-GCM, and Section 5 summarizes the paper.

2. Preliminaries and Notation

In this paper, we refer to the well known AES-GCM scheme as defined in [15]. For context and notation, we provide a brief parameterized description.

Let E be a block cipher with block size n bits, and key length κ . E -GCM is the straightforward formulation of AES-GCM using E for the block cipher. We fix a parameter $0 < \delta < n$ and set the length of the initialization vector IV to $\ell_{IV} = n - \delta$. GHASH is the polynomial-evaluation universal hash function computed in $GF(2^n)$ using the hash key H , analogously to the definition of AES-GCM.

A block is an element of $\{0, 1\}^n$. A valid message M consists of (up to) ℓ_M plaintext blocks, and (up to) ℓ_{AAD} Additional Authenticated Data (AAD) blocks AAD , where $\ell_M \leq 2^\delta - 2$. For simplicity, we assume that plaintext and AAD lengths are integer multiples of n .

Consider the encryption of M under the key K and IV . The outputs are the ciphertext C (of the same length as the plaintext) and the authentication tag Tag (of n bits) that are computed as follows. First the GHASH key is set to $H = E_K(0)$. Then the $\ell_M + 1$ blocks $CTRBLOCK_{IV,j} = [IV \parallel \#(j+1)]$, $j = 0, \dots, \ell_M$, are fixed, where $\#j$ denotes the encoding of the integer j as a string of δ bits and $\parallel$ denotes concatenation. Note that for a given IV , $CTRBLOCK_{IV,j}$, $j = 0, \dots, \ell_M$, are distinct, and also different from the zero block, and that for every $0 \leq i, j \leq \ell_M$ and $IV' \neq IV$, we have $CTRBLOCK_{IV,i} \neq CTRBLOCK_{IV',j}$.

The ciphertext C consists of ℓ_M ciphertext blocks where the j -th ciphertext block is the XOR of the j -th plaintext block with $E_K(CTRBLOCK_{IV,j})$. The authentication tag is the XOR of the GHASH value (computed over the ciphertext blocks and the AAD) with $E_K(CTRBLOCK_{IV,0})$. The flows for decryption and tag verification are straightforward. Decryption takes C , Tag , AAD , and IV as input, and returns M if the authentication passes, and $\perp$ otherwise.

The list of $\ell_M + 1$ counter blocks that are encrypted with E during the encryption of M using IV is denoted

$$\mathcal{L}(IV, \ell_M) \quad (1)$$

The additional encryption of the zero block to form the GHASH key $H = E_K(0)$ is amortized over all encryptions under the same key K . For encryption, the IV can be chosen randomly, and in that case we call the scheme E -GCM with a random IV . However, it is critical to never encrypt two (or more) messages under the same key and with the same IV .

We use the following result of Suzuki et al. [16] for the probability of a multicollision.

Theorem 1 (Theorem 2 of [16]). *Let $2 \leq r \leq q \leq A$ be integers. Suppose that q balls are thrown, one by one (independently) at random, into A bins. An r -multicollision is the event where there exists at least one bin that contains at least r balls. Denote this event by $MultiColl(A, q, r)$. Then*

$$\Pr[\text{MultiColl}(A, q, r)] \leq \frac{q^r}{r! \cdot A^{r-1}}$$

3. A Cloud-Based Key Management Service

The AWS Key Management Service allows customers to create and manage keys and control the use of these keys across a wide range of AWS services and applications. AWS KMS protects these customer keys by ensuring they are never accessible outside of FIPS 140-2 validated hardware security modules. AWS KMS, while capable of being used as an encryption service, primary application is for generating, encrypting and decrypting data keys to be used in envelope encryption. Attempting to channel all data through a single service like AWS KMS would be bandwidth and computationally infeasible at acceptable latency times. Additionally, it makes little sense to transmit the data over an encrypted channel only to receive it back encrypted under another key. To prevent these unwanted use-cases AWS KMS limits the amount of data that can be encrypted to 4096 bytes.

We refer to the customer keys as Customer Master Keys (CMKs). When a CMK is created, a key identifier (keyId) and an access control policy is associated with the key. The policy restricts which users (principals) can call what APIs (actions) using this key. AWS provides a rich set of tools to specify access control policies [17].

AWS KMS is a distributed tier-service where users call into a front-end service, which in turn communicate with the HSMs. Distributed services deliver greater availability and partition tolerance at the expense of latency to ensure consistency. AWS KMS is designed to provide simultaneous data-plane calls (GenerateDataKey, Encrypt, and Decrypt) with minimal latency. This is accomplished by ensuring that data-plane calls do not require the distribution of state to ensure consistency. This introduces new requirements that impact how modern encryption modes, like AES-GCM, are used within a high-volume distributed service.

We depict the AWS KMS in Figure 1 through two basic APIs. CreateKey is a control plane API, that results in a CMK being created on behalf of a customer. This causes a state change within AWS KMS where a new CMK must be durably stored as an encrypted key token EKT, and synchronized across the data storage layer. Encrypt is a data plane operation which utilizes a CMK but does not alter the state of AWS KMS.

Figure 1. A description of AWS KMS from a user's perspective.

The service is integrated with many other AWS services to help users protect the data stored in these services. AWS KMS constitutes an instantiation of the CES model that is described in this paper.

3.1. Requirements

The design of a CES is naturally iterative, in which initial requirements are met with design decisions which impose new requirements. We start our description with constraints, and then identify new constraints imposed by design decisions.

Cloud computing provides on-demand compute power, database, storage, applications and other IT services. It provides these services through a web-based API, authenticated by using account credentials. Each API call is authenticated and authorized against policies associated with resources. Customers reason about their data differently, and a cloud-based key management system should

be flexible in the ways it allows customers to use authorization policies to compartmentalize access to data. The ability to define a policy based on a customer master key (CMK) object helps customers impose their logic onto the access control of data in cloud provider services.

A major benefit of cloud computing is the economies of scale using shared infrastructure. In cryptography, these benefits can only be realized if they are met with additional assurances that the cloud provider is providing comparable security measures. Thus, FIPS 140-2 certification of hardware components and clear SOC reports and audits that cover the system should be provided.

There are many ways to meet the challenges of key management in the cloud [18]. In addition to providing a secure interaction between the customer and the cloud services, and the integration of encryption into those services, the overall design must afford highly distributed encryption of data. In order to achieve this, a CES must support a distribution of roles of key generation and data encryption, a form of envelope encryption.

Additional requirements are derived from customer input and yielded the following:

- Copious: must support many CMKs. Customers want to reason about use cases.
- FIPS: CMKs can only be accessed within a FIPS 140-2 certified security module.
- Low-latency: generate, encrypt and decrypt of data keys under a CMK must have low-latency.
- Simultaneous use: A CMK can be used simultaneously within the system.
- Durability: CMKs must be at least as durable as the data that they protect.
- High volume: each CMK must be able to encrypt a large number of objects.
- Scalability: a CES should be able to support a large volume of calls across many customers.
- Distributive: a CES should be able to distribute the role of key generation and bulk data encryption.

3.2. Desired Properties of a CES

A CES is not meant to be a general-purpose cryptographic service provider because transmitting all data over the network for encryption would cause unnecessary bottlenecks. Instead, a CES design facilitates envelope encryption, where a data key is generated on an HSM, and returned to the user in plaintext and encrypted under the user's CMK. The user can encrypt their data locally, delete the data key in memory, store the encrypted data, and CMK-encrypted data key. Consequently, a CES design can limit the amount of data that can be encrypted in a single Encrypt call.

A CES should support IND-CPA and IND-CCA2 semantic security (i.e., be secure under non-adaptive and adaptive chosen plaintext and chosen ciphertext attacks). Ideally, it would use an encryption mode with 256-bit keys, in order to satisfy the strictest requirements that customers may require, and also accommodate multi-key and multi-user complications.

Message authenticity. A fully featured access control policy on a CMK allows authorization constraints that mimic aspects of public key cryptography. It is possible to write access control policies that separate a set of users who can call "Encrypt" or "GenerateDataKey" under a given CMK, from a set of users that can call "Decrypt". Further, a CES design should be able to enforce policies like "all users can encrypt before midnight", and "only a specific user can decrypt after midnight". This highlights a specific need for message authenticity for cases where the decrypting entity is different from the encrypting entity.

Message authenticity of a successfully decrypted message requires verification that the correct CMK was used to decrypt the message. This can be enforced by verifying the correct CMK is used during the decrypt call. Otherwise, consider the scenario where all the ciphertexts are replaced with a new set of ciphertexts, encrypted to an adversary's CMK. The adversary could set a permissive policy that lets all users decrypt under their CMK. When the user calls Decrypt, the decryption will succeed, but the message will have been encrypted by the adversary. Default policies for a CES should prevent such a scenario, so by default users should not be allowed to call across accounts without modifications to default policies. In our design it is recommended that the input key identifier (keyId) is verified against the expected key identifier. Furthermore, it is important that the service would provide configurable alert mechanisms on decryption failures to alert on potential forgery attempts.

Durability. Cloud providers deliver durability through redundant storage. A CES should support at least the amount of durability the cloud provider supports for the integrated services that are using it. This requires secure redundant storage of CMKs. The requirement on redundancy and the number of independent CMKs a CES needs to support preclude the ability to store CMKs on cost-efficient commodity HSMs.

Auditability. Information systems require complete auditing. Especially since, unlike an on-premise system, customers cannot directly inspect cloud provider systems. Customers demand to have clear audit records of when attempts are made to access their data. Thus, a CES must also supply to its users all attempted access and use of their CMK.

3.3. Requirement Driven Design

This section shows how the general requirements are translated to the set of design choices made for AWS KMS. The requirements for a CES include the ability to secure all CMKs in a FIPS 140-2 certified HSM. The master keys themselves must be highly durable and accessible with low-latency. A service provider should strive to remove all limits on the customer's use. Users should be able to protect quadrillions of objects under a single master key, and the service should be able to support trillions of master keys on behalf of its customers.

The FIPS requirement restricts the selection of cryptographic algorithms to the use of FIPS-approved methods. The obvious (and practically the only) choice for the authenticated encryption, that ensures the system provides the broadest use case, is AES-GCM [15]. This mode (more specifically, AES256-GCM with a 96-bit initialization vector) is indeed chosen for all encryptions done by AWS KMS. However, we show here why the straightforward application of AES-GCM involves some serious limitations in the CES setting, thus illustrating why some enhancement is required.

In AES-GCM, the reuse of an *IV* results in major security issues. Repeating a derived counter value between two instantiations of AES-GCM under the same key leads to loss of confidentiality for these two messages. More critically, if an adversary observes two encryptions under the same *IV* (and key), it is likely that the hash key (*H*) used in the scheme would be discovered. Learning the hash key would allow an adversary to modify the *AAD* or ciphertext (or both), and use the exposed hash key to create a valid message authentication tag. The AES-GCM specification [15] addresses this concern and requires the following:

“The probability that the authenticated encryption function ever will be invoked with the same *IV* and the same key on two (or more) distinct sets of input data shall be no greater than 2^{-32} .”

This constraint is not limited to the single user scenario, but applies equally to the multi-user scenario of a CES. In other words, this creates a requirement for a CES to ensure that this probability is not exceeded globally, across all keys and *IV*s, and is not simply limited to a single key.

High availability for continuously operating services requires redundancy. Delivering a consistent service across a redundant fleet of HSMs requires some state management. To the extent possible, state replication should be minimized. A CES needs to ensure that the most time-sensitive operations (in particular, encrypt and decrypt under a customer master key) do not involve a synchronized state change across the HSM fleet. This allows for delivering a highly available service with low latency. To reduce managed state, AES-GCM is used with random *IV*s. We note that alternative counter-based methods require, at a minimum, process or thread communication of state across encryption calls locally on an HSM. Unfortunately, the use of a 96-bit random *IV* limits the amount of data that can be encrypted directly with a single CMK to 2^{32} (≈ 4 billion) messages (Note that using an *IV* with bit-length different from 96, translates (by the definition of AES-GCM) to processing a “randomized” 96-bit portion in the resulting counter blocks. This leads to exactly the same limitation on the number of usages, as the case of using 96-bit *IV*). To illustrate, if a commercial CES is designed to support encryption rates of 1200 requests per second, then master key rotations would be required every

41 days. This would result in more state management, and increased storage within the service. This serious drawback is an indication that AES-GCM is not suitable for a CES in its native form, and some enhancements are needed.

The top of an individual user's key hierarchy is a customer master key (CMK). It is important to understand that by the security design considerations, the owner of a CMK can use it only implicitly, via (authenticated) web-based APIs.

In our CES, AWS KMS, a user's CMK is generated on an HSM and is accessible only on the HSMs managed by the service. To meet our durability requirements, CMKs are stored encrypted outside of the HSM fleet in an online distributive database and a highly durable offline data store. CMKs are bound to a globally unique key identifier (keyId) assigned by the distributive database system. It is bound by the HSM's encryption of the CMK. The keyId is returned to the user on a successful request to create a CMK. AWS KMS only allows access to encrypt and decrypt calls under a CMK using the secure defaults of the system. An access control policy is associated with keyId, and enforced on every API call referencing keyId. The policy controls which users can encrypt or decrypt using a specific CMK.

For context, we provide some additional details on the AWS Key Management Service that utilizes FIPS-approved algorithms for the cryptographic operations. On an encryption call, a user who wishes to encrypt a message M under their CMK, passes the keyId, M , and AAD to an "Encrypt" API. On an HSM, a freshly generated nonce N is used for deriving a fresh encryption key K , using the NIST SP800-108 Key Derivation Function (KDF) in Counter Mode with PRF HMAC-SHA256 [19]. Subsequently, a fresh IV is used for encrypting M into ciphertext C and computing Tag for C and AAD, under the derived key K . In our concrete instantiation, the limit we analyze is 4096 bytes (256 blocks) for the plaintext message, and 8192 bytes (512 blocks) for AAD. A single data structure ciphertextBlob containing an internal key identifier u , N , IV, C , and Tag, is returned to the user. Figure 2 illustrates the process.

Figure 2. An outline of the AWS KMS encryption flow in the context of user u . $\Pi_K()$ symbolizes an IV-based authenticated encryption with associated data (AEAD) scheme, AES256-GCM in our case. The randomized nonce (N) and IV can come from separate entropy sources providing protection against correlated failures.

A good design of a CES should envision the possibility of separating the modules for generating per-message encryption keys from the modules handling the actual message encryption. This facilitates cryptographic isolation of the CMK-handling modules and allows the components to scale independently. A primary function of AWS KMS is to protect data keys for use in other applications through the "GenerateDataKey" API. This is the essential method for generating data keys to encrypt customer data within AWS. However, the design of AWS KMS leaves the ability to encrypt larger amounts of data within AWS KMS by moving the key K closer to the data to be encrypted, thus reducing bandwidth requirements. Within AWS KMS the role to generate the N and derive K from the CMK can be separated from the role to generate IV and encrypt the message M . This reduces correlated errors in generation of the N and IV, and allows for independent scaling of per-message key derivation and message encryption.

An authorized user can call a “Decrypt” API passing in AAD and ciphertextBlob. An HSM will parse ciphertextBlob, extract the key identifier u , N , IV , C , and Tag , and obtain K using the referenced CMK and N . The HSM will verify the message authenticity over C and AAD, and (conditionally) decrypt the message. Upon successful tag verification, the plaintext and CMK’s keyId are returned to the caller. A decryption error message is returned otherwise.

Part of the control privileges of a user in our design is that a user can configure key rotation for their CMK (see [20]). Key rotation would result in the generation of a new master key and an internal key identifier u , under the existing keyId. Obviously, the service must retain all key versions under keyId and enforce the associated key policy on subsequent calls using these keys. All new encryption calls will use the latest CMK version, and the version indicated by u in the ciphertextBlob will be used for all decryption calls. This way, key rotation at the CMK level is transparent to the user from the usability viewpoint. Since the designed AWS KMS does not store ciphertextBlob, only the user would be able to actively migrate already-encrypted data to a new CMK version. The design facilitates such migration through a “ReEncrypt” API.

4. Security Bounds for AWS KMS Mode Of Operation

4.1. Abstraction of an Idealized AWS KMS Mode CES-GCM⁽ⁱ⁾

This section outlines an abstraction of the AWS KMS mode of operation, CES-GCM, modeled with ideal primitives for the key derivation (h) and the block cipher (E). In particular, it leaves the AWS KMS authorization mechanism outside the scope of this discussion. We denote it CES-GCM⁽ⁱ⁾.

The CES-GCM⁽ⁱ⁾ mode is a variant of the derive key mode of [13], mounted on top of E -GCM. It operates in the context of a multi-users-multi-key system with the following parameters. The system supports U users, each one labeled by an identifier u , $1 \leq u \leq U$, associated with a master key (CMK_u), and has a budget of Q encryptions. The mode is nonce based, where the length of a nonce N is ℓ_N , satisfying $\ell_N \leq \kappa$. When a user u requests the encryption of a message M (plaintext and AAD), the encryption flow executes the following: a) chooses a random N and a random IV ; b) applies a key derivation function $h(CMK_u, N)$ (that is modeled here as ideal), to derive a key K ; c) E -GCM encrypts M with K and IV . The output is C , Tag , IV , and N . A decryption request takes C , Tag , IV , N , AAD, and a user identifier u as input. It triggers E -GCM decryption with a key K that is derived from CMK_u and N . The output is the plaintext of M if authentication passes, and $\perp$ otherwise.

Remark 1. In a concrete instantiation, AES256 is used for E , and PRF HMAC-SHA256 is used for h , $n = 128$, $\kappa = 256$, $\ell_{IV} = 96$, and $\ell_N = 128$. The maximum plaintext length is $\ell_M = 256$ and AAD length $\ell_{AAD} = 512$. The targeted limits are for $U = 2^{40}$ users and $Q = 2^{50}$ encryptions for each.

Remark 2. The CES-GCM⁽ⁱ⁾ mode can be viewed as a special case of the general derive key mode of [13], which can be applied over any IV based AEAD scheme, Π , but with the following difference. The derive key mode uses a single nonce N for the derivation of a per-message key, and for the encryption with Π . In contrast, CES-GCM⁽ⁱ⁾ mode uses N only for the derivation, and a separate (independent) random IV for Π . This decouples the per-request key derivation, from the actual message encryption.

Remark 3. To illustrate the value of deriving a per-message key from CMK and N , consider a trivialized instantiation where, for a user u , the derivation is $CMK_u \leftarrow (CMK_u, N)$, i.e., a direct use of CMK_u . With this, for every u , the probability that Q encryptions would lead to a collision in the randomized $n - \delta$ bits IV is at most $Q^2 / 2^{(n-\delta)}$. To ensure this probability remains below the target security margin of $2^{-\beta}$, the limit on Q is $2^{(n-\delta-\beta)/2}$. This imposes an undesired constraint on the users. For example, with $n = 128$, $\delta = 32$, $\beta = 32$, Q is limited to 2^{32} . The situation is even worse at the cloud scale, because if each one of U users encrypts Q messages, then the probability that (at least) one of them will repeat an IV (with their key) is $\approx UQ^2 / 2^{(n-\delta)}$, and bounding this probability limits $U \cdot Q^2$. The prepended nonce-based key derivation that is built into CES-GCM⁽ⁱ⁾, is intended to address these limitations.

4.2. Security Definitions for CES-GCM⁽ⁱ⁾

4.2.1. A CES-GCM⁽ⁱ⁾ Oracle

We define an oracle, $\mathcal{O}$, for CES-GCM⁽ⁱ⁾ encryption and decryption queries, which operates as follows.

Setup.

Select, uniformly at random,

- (a) a bit b .
- (b) U keys $\text{CMK}_1, \dots, \text{CMK}_U$, each one of κ bits.
- (c) a random function $h : \{0, 1\}^\kappa \times \{0, 1\}^{\ell_N} \rightarrow \{0, 1\}^\kappa$.

Response to an encryption query u, M, AAD .

- Select, uniformly at random,
 - (a) a string N of ℓ_N bits,
 - (b) a string IV of ℓ_{IV} bits,
 - (c) a string S of length $\ell_M + 1$ blocks.
- Compute $K = h(\text{CMK}_u, N)$.
- E-GCM encrypt M, AAD with IV , under the key K , obtaining the ciphertext C and the authentication tag Tag .
- Output: $N, IV, C \parallel \text{Tag}$, if $b = 0$, and u, N, IV, S , if $b = 1$.

Response to a decryption query $u, N, IV, C \parallel \text{Tag}, \text{AAD}$.

- Compute $K = h(\text{CMK}_u, N)$.
- E-GCM decrypt $C, \text{AAD}, \text{Tag}$, using IV , under the key K , obtaining the plaintext of M , and determining if the authentication passed or failed.
- Output: if $b = 0$ then: M (plaintext) if authentication passed and $\perp$ if it failed.
If $b = 1$ then: $\perp$.

4.2.2. Adversary against CES-GCM⁽ⁱ⁾

An adversary $\mathcal{A}$ against CES-GCM⁽ⁱ⁾ is an algorithm that submits encryption and decryption queries to $\mathcal{O}$, and then outputs a bit b' (as its guess for b). For simplicity (and with no loss of generality), assume that $\mathcal{A}$ exhausts the allowed number (Q) of encryptions for each user. Suppose also that all messages have the maximum allowed plaintext and AAD lengths. With this, the total number of encrypted messages during $\mathcal{A}$'s queries is $U \cdot Q$. Each query triggers $\ell_M + 2$ evaluations of E (with some key), over the list of counter blocks $\mathcal{L}$, all of which, except the invocation of $E_{(\cdot)}(0)$ (for the hash key) are revealed to $\mathcal{A}$ (whose encryption queries are with chosen plaintext). Since with nonzero probability, keys may repeat, the total number of different keys that are used with invocations of E , is at most $U \cdot Q$.

To model an active adversary, $\mathcal{A}$ can also submit Q_D decryption queries. We assume that $\mathcal{A}$ does not make superfluous decryption queries, i.e., it does not request to decrypt $(u1, N1, IV1, C1 \parallel \text{Tag1}, \text{AAD1})$ if it has already submitted an encryption query with $(u1, M1, \text{AAD1})$, and received the response $(N1, IV1, C1 \parallel \text{Tag1})$. Note that a decryption query of the form $(u2, N1, IV1, C1 \parallel \text{Tag1})$, where $u2 \neq u1$ is not superfluous. We call a non superfluous decryption query a forgery attempt.

We denote an event where $\mathcal{O}$ responds with a string (i.e., not $\perp$) to a forgery attempt “forge”. $\mathcal{A}$ can also make T_E (offline) evaluations of E (or E^{-1}), using its chosen keys, as an attempt to guess a secret key that $\mathcal{O}$ uses. The event where $\mathcal{A}$ found such a key is called “guess”. After the queries, $\mathcal{A}$ outputs a bit b' (as its guess for b).

Adversary advantage. The advantage of $\mathcal{A}$ against CES-GCM⁽ⁱ⁾ is $|\Pr(b' = 1 \mid b = 1) - \Pr(b' = 1 \mid b = 0)|$.

The PRF advantage of E in a multi-user-multi-key setting. Define a multi-user-multi-key oracle $\mathcal{O}'$ for E as follows. Let U and Q be given parameters. At setup, $\mathcal{O}'$ chooses a random bit c , $U \cdot Q$

random keys of κ bits, and $U \cdot Q$ random functions $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$, such that if two selected keys are equal the corresponding functions are also equal. Assume the keys and functions are organized in a table of U rows and Q columns indexed by u , and ind . A query to $\mathcal{O}'$ is a tripe $[u, ind, B]$ for some u , ind , and $B \in \{0, 1\}^n$. The response is either $E_{K_{u,ind}}(B)$ or $f_{u,ind}(B)$, depending on c . An adversary $\mathcal{A}'$ against E (in this setting) is an algorithm that submits queries to $\mathcal{O}'$ and outputs c' as its guess for c . The advantage of $\mathcal{A}'$ after exhausting a budget of $q' = U \cdot Q \cdot (\ell_M + 1)$ queries is $|\Pr(c' = 1 | c = 1) - \Pr(c' = 1 | c = 0)|$.

4.3. Security Bounds for CES-GCM⁽ⁱ⁾

4.3.1. Events That May Occur during Encryption Queries

Consider the list of Q encryption queries with a given identifier u , where the queried messages are $M_{u,1}, \dots, M_{u,Q}$, and the respective nonces, keys, and IV's, used with these queries are $N_{u,1}, \dots, N_{u,Q}$, $K_{u,1}, \dots, K_{u,Q}$, $IV_{u,1}, \dots, IV_{u,Q}$. Denote also the lists of nonces, keys and IV's, used during the queries, by $\mathcal{N}(u)$, $\mathcal{K}(u)$, $\mathcal{IV}(u)$, respectively, and the list of the U CMK's by $\mathcal{CMK}$. Note that these lists may contains repeated values. Finally, denote the combined (concatenated) list of $(U \cdot Q)$ IV's and of $U + U \cdot Q$ keys, respectively, by

$$\Sigma \mathcal{IV} = \mathcal{IV}(1) \parallel \mathcal{IV}(2) \parallel \dots \parallel \mathcal{IV}(U) \quad (2)$$

$$\Sigma \mathcal{K} = \mathcal{CMK} \parallel \mathcal{K}(1) \parallel \mathcal{K}(2) \parallel \dots \parallel \mathcal{K}(U) \quad (3)$$

We now account for several types of events, as follows. Let μ_0 be a parameter, (the value $\mu_0 \ll (U \cdot Q)$ is to be determined later). Define the following events that may occur during the oracle's setup and responses to queries.

1. (Λ_1) There are two identifiers $1 \leq u < v \leq U$, such that $\mathcal{CMK}_u = \mathcal{CMK}_v$.
2. (Λ_2) All the CMK's are distinct, and there are identifiers $1 \leq u < v \leq U$, and indexes $1 \leq i, j \leq Q$, such that $K_{u,i} = K_{v,j}$.
3. (Λ_3) There is an identifier $1 \leq u \leq U$, such that $\mathcal{K}(u)$ contains a value that is repeated 3 or more times.
4. (Λ_4) There is an identifier $1 \leq u \leq U$, and indexes i, j , $1 \leq i < j \leq Q$, such that $K_{u,i} = K_{u,j}$ and $IV_{u,i} = IV_{u,j}$.
5. (Λ_5) The combined list $\Sigma \mathcal{IV}$ includes a value that is repeated more than μ_0 times.

Interpretation. Events $\Lambda_1, \Lambda_2, \Lambda_4$ are "bad" events, that obviously compromise the security promise of CES-GCM⁽ⁱ⁾. Event Λ_1 is a collision on master keys: users u and v are completely not isolated with respect to encrypting and decrypting messages. Event Λ_2 is a cross-users key contamination. It allows user v to decrypt message $M_{u,i}$ that was encrypted by user u with $K_{u,i}$ (in the real AWS KMS context, this bypasses a decryption privilege policy imposed by u on $M_{u,i}$). Event Λ_4 implies that user u loses the privacy (of $M_{u,i}$ and $M_{u,j}$) and the authenticity (with $K_{u,i}$) due to an improper usage (IV reuse) of E-GCM. Event Λ_5 give a lower bound for the number of IV's (and hence counter blocks) that were repeated during the encryption queries. The interpretation motivates why we focus on these events. We are actually interested in the case where none of these events occur. The following lemma analyzes the positive properties of no such event occurring.

Lemma 1. *Let Λ be the event where at least one of $\Lambda_1, \Lambda_2, \Lambda_3, \Lambda_4, \Lambda_5$ happens during the encryption queries. Then, if the event Λ does not happen:*

- c1. *All the CMK's are distinct.*
- c2. *For every $1 \leq u \leq U$, the usage of E-GCM by u was proper. Furthermore, $\mathcal{K}(u)$ can be split to disjoint sub-lists: $s(u)$ keys that were used for encrypting a single message, and $d(u)$ keys that were used for encrypting two messages. These satisfy the relation $s(u) + 2d(u) = Q$, and $s(u), d(u) \geq 0$.*
- c3. *Across all the $U \cdot Q$ encryptions, every counter block is encrypted under at most μ_0 distinct keys.*

Proof. The proof follows directly from the definitions of the events $\Lambda_1, \Lambda_2, \Lambda_3, \Lambda_4, \Lambda_5$, and the definition of Λ . Specifically, note that the negation of Λ is the case where none of $\Lambda_1, \Lambda_2, \Lambda_3, \Lambda_4, \Lambda_5$ occurs. $\square$

Lemma 2.

$$\begin{aligned} \Pr(\Lambda_1) &\leq \frac{U^2}{2 \cdot 2^\kappa}, \quad \Pr(\Lambda_2) \leq \frac{(U \cdot Q)^2}{2 \cdot 2^\kappa}, \quad \Pr(\Lambda_3) \leq U \cdot \frac{Q^3}{2 \cdot 2^{2\ell_N}}, \\ \Pr(\Lambda_4) &\leq U \cdot \frac{Q^2}{2} \cdot \left(\frac{1}{2^\kappa} + \frac{1}{2^{\ell_{IV}}} \right) \cdot \frac{1}{2^{\ell_N}}, \quad \Pr(\Lambda_5) \leq \frac{(U \cdot Q)^{\mu_0}}{\mu_0! \cdot 2^{\ell_{IV} \cdot (\mu_0 - 1)}} \end{aligned} \quad (4)$$

Proof. The bound for $\Pr(\Lambda_1)$ is the standard bound for the collision probability among U randomly chosen values from a pool of 2^κ possibilities.

If no CMK values collide, then $U \cdot Q$ keys $K_{u,i} = h(\text{CMK}_u, i)$, $1 \leq u \leq U$, $1 \leq i \leq Q$ are uniform random samples from a pool of 2^κ possibilities. Therefore,

$$\Pr(\Lambda_2) \leq \frac{(U \cdot Q)^2}{2 \cdot 2^\kappa} \quad (5)$$

To bound $\Pr(\Lambda_3)$, fix some u , and consider the event $\Lambda_3^{(u)}$, where the list $\mathcal{K}(u)$ has a value that is repeated three times. For $\Lambda_3^{(u)}$ to happen, one of the conditions needs to be satisfied, for some distinct $1 \leq i, j, k \leq Q$: (a) $N_{u,i} = N_{u,j} = N_{u,k}$; (b) $N_{u,i} = N_{u,j} \neq N_{u,k}$ and $h(\text{CMK}_u, N_{u,i}) = h(\text{CMK}_u, N_{u,k})$; (c) $N_{u,i} \neq N_{u,j} \neq N_{u,k}$, and $h(\text{CMK}_u, N_{u,i}) = h(\text{CMK}_u, N_{u,j}) = h(\text{CMK}_u, N_{u,k})$. Obviously (recall that $\ell_N < \kappa$), the probabilities for #b and #c are smaller than the probability for #a. By Theorem 1, it follows that

$$\Pr(\Lambda_3^{(u)}) \leq 3 \cdot \frac{Q^3}{6 \cdot 2^{2\ell_N}} = \frac{Q^3}{2 \cdot 2^{2\ell_N}} \quad (6)$$

Finally, $\Pr(\Lambda_3) \leq U \cdot \Pr(\Lambda_3^{(u)})$.

To bound $\Pr(\Lambda_4)$, fix some u , and consider the event $\Lambda_4^{(u)}$, where $K_{u,i} = K_{u,j}$ and $IV_{u,i} = IV_{u,j}$ for some $1 \leq i \neq j \leq Q$. The collision $K_{u,i} = K_{u,j}$ occurs if $N_{u,i} = N_{u,j}$ or if $h(u, N_{u,i}) = h(u, N_{u,j})$. The probability that this and $IV_{u,i} = IV_{u,j}$ happen leads to

$$\Pr(\Lambda_4^{(u)}) \leq \frac{Q^2}{2} \cdot \left(\frac{1}{2^\kappa} + \frac{1}{2^{\ell_N}} \right) \cdot \frac{1}{2^{\ell_{IV}}} \quad (7)$$

Finally, $\Pr(\Lambda_4) \leq U \cdot \Pr(\Lambda_4^{(u)})$.

The event Λ_5 is a μ_0 -collision among $U \cdot Q$ random samples from a pool of $2^{\ell_{IV}}$ options, so the bound on $\Pr(\Lambda_5)$ follows from Theorem 1. $\square$

We now formulate the privacy bounds for the (idealized) CES-GCM⁽ⁱ⁾ mode, against a passive adversary that makes no forgery attempts.

Theorem 2 (CES-GCM⁽ⁱ⁾ privacy bound). *Let E be an ideal cipher. Let $\mathcal{A}$ make a total of $U \cdot Q$ encryption queries, each one with ℓ_M plaintext blocks and ℓ_{AAD} AAD blocks, where for each identifier $u = 1, \dots, U$, there are Q queries. Let $\mathcal{A}$ make also a total of T_E (offline) evaluations of E or its inverse, using its chosen keys. Let μ_0 be a (small) parameter. Then, the advantage of $\mathcal{A}$ against the (idealized) CES-GCM⁽ⁱ⁾ has the following upper bound*

$$\text{Adv}_{\text{CES-GCM}^{(i)}}(\mathcal{A}) \leq \frac{U^2}{2 \cdot 2^\kappa} + \frac{(U \cdot Q)^2}{2 \cdot 2^\kappa} + U \cdot \frac{Q^3}{2 \cdot 2^{2\ell_M}} + \frac{U \cdot Q^2}{2} \cdot \left(\frac{1}{2^\kappa} + \frac{1}{2^{\ell_M}} \right) \cdot \frac{1}{2^{\ell_{IV}}} + \frac{(U \cdot Q)^{\mu_0}}{\mu_0! \cdot 2^{\ell_{IV} \cdot (\mu_0 - 1)}} + 2 \cdot U \cdot Q \cdot \frac{(\ell_M + 1)^2}{2^{n+1}} + \mu_0 \cdot \frac{T_E}{2^\kappa} \quad (8)$$

Proof. Step 1. We define a random version of $\text{CES-GCM}^{(i)}$, as follows. Choose a random function $F : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^n$. Then for every encryption query $[u0, M0]$, for which the chosen nonce and IV are $N0$ and $IV0$, respectively, and the derived key is $K_{u0, N0}$, replace the invocations of $E_{K_{u0, N0}}(B)$, for $B \in \mathcal{L}(IV0, M0)$ with $F(K_{u0, N0}, B)$. We call this scheme $\text{CES-GCM}^{(i, \text{rand})}$.

Step 2. We build an adversary $\mathcal{A}'$ against the PRF security of E in the multi key setting, that has querying access to $\mathcal{O}'$. $\mathcal{A}'$ uses $\mathcal{A}$ as follows. $\mathcal{A}'$ chooses U random keys CMK' , and runs algorithm $\mathcal{A}$. For each query $[u0, M0]$ that $\mathcal{A}$ prescribes, $\mathcal{A}'$ generates a random nonce $N0'$ and a random IV , $IV0'$, and computes $X0' = h(\text{CMK}'_{u0}, N0')$. If $X0'$ is not a value that has appeared in previous queries, $\mathcal{A}'$ assigns the next index value $\text{ind}0'$ that has not been used (and uses the appropriate already-used index otherwise). Then, $\mathcal{A}'$ queries $\mathcal{O}'$ with queries of the form $[\text{ind}0', B]$ where $B \in \mathcal{L}(IV0', M0)$ and uses these values to compute the response to $\mathcal{A}$. We note that during the $U \cdot Q$ queries of $\mathcal{A}$, $\mathcal{A}'$ queries $\mathcal{O}'$ $U \cdot Q \cdot (\ell_M + 1)$ times. We have

$$\text{Adv}_{\text{CES-GCM}^{(i)}}(\mathcal{A}) \leq \text{Adv}_{(multi)E}^{\text{PRF}}(\mathcal{A}') + \text{Adv}_{\text{CES-GCM}^{(i, \text{rand})}}(\mathcal{A}) \quad (9)$$

Now, assume that Λ does not happen.

Under the negation of Λ , and by Lemma 1, we can see that $\text{Adv}_{\text{CES-GCM}^{(i, \text{rand})}}(\mathcal{A}) = 0$. This is because $\mathcal{A}$ gets to observe (at most) $U \cdot Q \cdot (\ell_M + 1)$ evaluations of the random functions over known blocks but distinct counter blocks.

To bound $\text{Adv}_{(multi)E}^{\text{PRF}}(\mathcal{A}')$ (under the negation of Λ), we use a standard hybrid argument (and the switching lemma). Fix some u , and consider the PRP-PRF advantage against the Q queries labeled under u . Using Lemma 1 (c2), we split the keys used for these messages to: $s(u)$ distinct keys, each one used for encrypting a single message, i.e., $(\ell_M + 1)$ blocks; $d(u)$ distinct keys, each one used for encrypting two messages, i.e., $2 \cdot (\ell_M + 1)$ blocks. We have $s + 2d = Q$, so, the cumulative PRP-PRF advantage for identifier u is upper bounded by

$$s \cdot \frac{(\ell_M + 1)^2}{2^{n+1}} + d \cdot \frac{(2 \cdot (\ell_M + 1))^2}{2^{n+1}} = (Q + 2d) \cdot \frac{(\ell_M + 1)^2}{2^{n+1}} \leq 2 \cdot Q \cdot \frac{(\ell_M + 1)^2}{2^{n+1}} \quad (10)$$

Note that we account for only $\ell_M + 1$ encryptions per message, ignoring the encryption of the zero block, which we assume that $\mathcal{A}'$ does not use for distinguishing (rather, only for computing GHASH in order to respond to $\mathcal{A}$). However, the time (in terms of calls to E) for $\mathcal{A}'$ should take into account $\ell_M + 1$ encryptions per message (just as the number of encryptions that the real usage of $\text{CES-GCM}^{(i)}$ requires from the system). After summing over all U identifiers, we get

$$\text{Adv}_{(multi)E}^{\text{PRF}}(\mathcal{A}') \leq 2 \cdot U \cdot Q \cdot \frac{(\ell_M + 1)^2}{2^{n+1}} \quad (11)$$

To account for the key recovery, we use Lemma 1 (c3) to establish

$$\text{Pr}(\text{guess}) \leq T_E \cdot \frac{\mu_0}{2^\kappa} \quad (12)$$

Using the bounds for the probabilities of $\Lambda_1 - \Lambda_5$, stated in Lemma 2, (12), and (11), gives the desired upper bound (8). $\square$

Remark 4 (The parameter μ_0). A selection of a value of μ_0 balances between the key recovery probability term in (8), $\mu_0 \cdot \frac{T_E}{2^\kappa}$ (which is linear in μ_0), and the upper bound on the probability for encountering an IV value that is repeated μ_0 times during the queries, namely $(U \cdot Q)^{\mu_0} / (\mu_0! \cdot 2^{\ell_{IV} \cdot (\mu_0 - 1)})$. For example, with $\ell_{IV} = 96$, selecting $\mu_0 = 20$ brings the probability term to

$$\frac{(U \cdot Q)^{20}}{20! \cdot 2^{1824}} \approx \frac{(U \cdot Q)^{20}}{2^{1924}}$$

which is smaller than 2^{-32} even when for $(U \cdot Q) = 2^{94}$. At the same time, the key recovery probability which is manifested through the term $20 \cdot \frac{T_E}{2^\kappa}$ remains negligible under any reasonable assumption, especially when $\kappa = 256$.

Interpreting Theorem 2

Let us first label the seven terms that contribute to the advantage bound in (8) as follows.

$$\begin{aligned} & \underbrace{\frac{U^2}{2 \cdot 2^\kappa}}_i, \quad \underbrace{\frac{(U \cdot Q)^2}{2 \cdot 2^\kappa}}_{ii}, \\ & \underbrace{U \cdot \frac{Q^3}{2 \cdot 2^{2\ell_N}}}_{iii}, \quad \underbrace{\frac{U \cdot Q^2}{2} \cdot \left(\frac{1}{2^\kappa} + \frac{1}{2^{\ell_N}} \right)}_{iv}, \quad \frac{1}{2^{\ell_{IV}}} \underbrace{\frac{(U \cdot Q)^{\mu_0}}{\mu_0! \cdot 2^{\ell_{IV} \cdot (\mu_0 - 1)}}}_v, \\ & \underbrace{2 \cdot U \cdot Q \cdot \frac{(\ell_M + 1)^2}{2^{n+1}}}_{vi}, \quad \underbrace{\mu_0 \cdot \frac{T_E}{2^\kappa}}_{vii} \end{aligned} \quad (13)$$

Consider the viewpoint of an individual user, say u . This can be deduced by taking $U = 1$ in (8) (the term i actually vanishes, because the exact numerator is $U \cdot (U - 1)$; the theorem uses the more loose U^2 just for convenience). The single user u is in a multi-key situation, where we take μ_0 to represent IV collisions on queries made by u . When κ is sufficiently large, we can assume even a “moderate” value of μ_0 (e.g., 20) and keep both v and vii terms small (note that the term vii which is linear in μ_0 remains extremely small for any reasonable T_E). To illustrate consider the case where $\ell_{IV} = 96$, $\kappa = 256$, and target a high number $Q = 2^{50}$ for the maximum allowed number of encryptions. Then, it suffices to set (exaggerated) $\mu_0 = 20$, in order to dwarf both v and vii .

We now focus on terms iv and vi . Term vi represents the PRP-PRF distinguishing for Q queries, but when the messages (ℓ_M) are sufficiently short, the quadratic numerator is still well behaved. The term iv represents the top concern of “repeated K and IV ” which breaks the usage of AES-GCM. The probability for this catastrophic collision is kept low if $\ell_{IV} + \ell_N$ is sufficiently large.

However, note that this does not cover all of u ’s concerns as a single user, in an environment that allows for $U > 1$. In such cases, cross-users “contamination” may occur through collisions on CMK’s or on derived keys (terms i and ii). This shows why the multi-user situation requires the use of a long encryption key, i.e., motivates the choice of $\kappa = 256$ over $\kappa = 128$ that could be acceptable with $U = 1$.

Consider the remaining terms under the general case $U > 1$. We deal with those that grow linearly or faster than linear with U . Term ii which is quadratic in $(U \cdot Q)$, is contained by the key length κ . In contrast, term vi , which is only linear in $(U \cdot Q)$ becomes the dominant term when both U and Q are large. Unfortunately, it is the direct consequence of PRP-PRF distinguishing over a block cipher with n bits, regardless of κ . This term represents a tight bound on distinguishing at the cloud provider’s level of all of the messages of all U users, but not at the user’s perspective (where the distinguishing

advantage of encryptions made with their CMK is linear in Q). However, it is reassuring to realize that with restricted-length messages, the cloud scale distinguishing advantage still remains below a reasonable bound. For example, even with $Q = 2^{50}$ and $U = 2^{40}$, an exaggerated extreme in real-world deployment, vi is bounded by 2^{-22} .

Accounting for Real Primitives

The bounds for $\text{Adv}_{\text{CES-GCM}^{(i)}}(\mathcal{A})$ use a model $\text{CES-GCM}^{(i)}$, where E and h are ideal primitives. A real CES-GCM scheme that uses FIPS-approved methods might substitute AES256 for E and the NIST SP800-108 Key Derivation Function (in Counter Mode with PRF HMAC-SHA256 [19]) for h . To account for these substitutions, the bounds in (8) need to be updated: the PRP security of AES256 (with $U \cdot Q \cdot (\ell_M + 1)$ invocations, distributed as prescribed in Theorem 2), and the PRF security of HMAC-SHA256 (with $U \cdot Q$ invocations) need to be added to the RHS of (8). Note that the AES256 invocations are distributed across multiple keys that are not controlled by the adversary. Thus, the situation is different from the case where $U \cdot Q \cdot (\ell_M + 1)$ samples of AES256 are harvested from the same key. We make the standard assumptions that the PRP advantage of AES256 and the PRF advantage of HMAC-SHA256 are very small compared to the terms in (8) that are associated with $\text{CES-GCM}^{(i)}$. With this, we conclude that Theorem 2 gives a good approximation for the bounds of a real instantiation of CES-GCM as well.

Accounting for an Active (Forging) Adversary

We explain the amendments needed in Theorem 2 and its proof in order to accommodate the case where $\mathcal{A}$ makes Q_D forgery attempts in addition to the encryption queries specified above. In this case,

$$\text{Adv}_{\text{CES-GCM}^{(i, \text{rand})}}(\mathcal{A}) \leq Q_D \cdot \frac{(\ell_M + \ell_{\text{AAD}} + 1)}{2^n} \quad (14)$$

instead of 0 when $Q_D = 0$ with the passive adversary. This follows directly from the properties of GHASH as a polynomial evaluation and the bound on the combined message and AAD lengths $\ell_M + \ell_{\text{AAD}} + 1$ (“+1” accounts for the length-encoding block in the GHASH computations).

We note that for a decryption query, $\mathcal{A}$ can choose to submit a query $(u^*, N^*, IV^*, C^* \parallel \text{Tag}^*)$ where the combination of u^* and N^* is not used for any of the encryption queries. This could (and most probably would) lead the key derivation procedure to generate a “new” key that was not generated (and used) in any of the encryption queries. In the proof of Theorem 2, this changes the distinguishing game of $\mathcal{A}'$ against E and the setup and operation of $\mathcal{O}'$ and $\mathcal{A}'$, as follows. At setup, $\mathcal{O}'$ will select Q_D additional random keys and random functions that will be used in the analogous way for every decryption query that generates a new key. When $\mathcal{A}$ prescribes such a decryption query, $\mathcal{A}'$ would point to an appropriate index (from the additional list) in order to get the encryption of the zero block from $\mathcal{O}'$ and use it to check the authentication Tag . Then, $\mathcal{A}'$ would pass the result of this check to $\mathcal{A}$. This means that $\mathcal{A}'$ submits Q_D more queries to $\mathcal{O}'$, compared to the case where $Q_D = 0$ (these are not used for the PRP-PRF distinguishing, but increase the time for $\mathcal{A}'$). Inspection of (14) shows that accounting for Q_D forgery attempts does not alter the security bounds of Theorem 2 in any meaningful way. In particular, since in addition to per-customer rate-limits, Q_D should be small because our recommended AWS KMS design includes mechanisms that alert on potential forgery attempts when multiple decryption failures are noticed. It should also be noted that a real adversary in the deployed system can only use valid values for u^* that exist in the system, and to which the adversary is authorized to call the Decrypt API. Hence, the adversary needs to be an authorized user of the CMK for the Decrypt API.

5. Discussion

This paper discussed the challenges and considerations involved with building and deploying a CES. These give rise to extremely heavy use of a block cipher in a multi-user-multi-key setting,

over distributed systems that require randomized nonces and/or IVs. The challenges are aggravated at the cloud-scale where the system needs to support a huge number of users and allow each user to encrypt a virtually unlimited number of messages. After laying out some of the constraints, we showed one solution in the form of the CES-GCM mode, and provided analysis that illustrates its suitability for the problem. There are alternative CES designs that will lead to different trade-offs between cost, performance, complexity and security. This analysis is based on the design of the AWS Key Management Service and is intended to provide transparency into the design of the service. Without the same level of access to the design of alternative CES it is impossible to produce a comparative analysis.

We note that CES-GCM belongs to the family of derive key modes that have been recently proposed and analyzed in [13], in the multi-key context, where AES-GCM-SIV [9] is one instantiation. This work was very recently followed by [14] who provided improved bounds for the multi-user-multi-key scenario for AES-GCM-SIV. Since AES-GCM-SIV (the variant with random nonces) and CES-GCM addresses the same problems, it is interesting to see some of the differences between these modes.

As pointed out above, CES-GCM decouples the use of the nonce as a seed for the per-message key derivation, from the use of the IV that seeds the underlying AEAD scheme (in this case, AES-GCM). This allows for using independent sources of entropy during the different steps of the encryption. Implementations can, therefore, enjoy an extra layer of protection against accidental misuse due to a failure in the entropy injection, which is detrimental for AES-GCM. This so-called nonce-misuse-independence feature of CES-GCM does not build on top of an underlying nonce-misuse resistant mode, but rather on top of the standardized AES-GCM. By comparison, AES-GCM-SIV starts from the nonce-misuse-resistant mode GCM-SIV+ and extends the lifetime of the key prepending a nonce-based key derivation step. CES-GCM can be viewed as an online mode, i.e., encryption does not need to have the full message prior to initialization. Furthermore, unlike any SIV construction (AES-GCM-SIV included) which serializes the universal hashing (of the plaintext and AAD) and the actual encryption, CES-GCM simply uses the standard AES-GCM but with a freshly derived key and a fresh IV. This allows for parallelizing the encryption and the GHASH computations, and achieves improved encryption performance on modern platforms. Of course, AES-GCM-SIV would be a viable CES mode alternative after it becomes a standard.

One conclusion of this paper is that cloud-scale encryption is pushing up against the natural birthday bounds of the standardized modes of a 128-bit block cipher. New standardized wide-block ciphers would alleviate the specialized engineering required to reach desired security bounds using existing schemes. Alternative simple solutions could be based on using 128-bit block ciphers (AES) with truncation or sum-permutation methods.

Finally, we note that the investment in analysis, FIPS-certified HSMs, and the availability and durability of AWS KMS surpasses what most users of cloud computing can achieve on their own. It is not feasible for a cloud provider to integrate independent customer key management solutions into all their services. The CES system described in this paper is deployed and integrated into 34 AWS services.

Author Contributions: M.C. and S.G. contributed equally to this work.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Services, A.W. AWS Identity and Access Management, 2016. Available online: <https://aws.amazon.com/kms/> (accessed on 31 August 2017).
2. McGrew, D.A.; Viega, J. The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In *Progress in Cryptology—INDOCRYPT 2004*; Canteaut, A., Viswanathan, K., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 343–355.

3. Abdalla, M.; Bellare, M. Increasing the Lifetime of a Key: A Comparative Analysis of the Security of Re-keying Techniques. In Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, Kyoto, Japan, 3–7 December 2000; Springer-Verlag: London, UK, 2000; pp. 546–559.
4. Smyshlyaev, S.V. *Re-Keying Mechanisms for Symmetric Keys*; Internet-Draft draft-irtf-cfrg-re-keying-11; Internet Engineering Task Force: Fremont, CA, USA, 2019.
5. Chatterjee, S.; Menezes, A.; Sarkar, P. Another Look at Tightness. In *Selected Areas in Cryptography*; Miri, A., Vaudenay, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 293–319.
6. Mouha, N.; Luykx, A. Multi-key Security: The Even-Mansour Construction Revisited. In Proceedings of the Advances in Cryptology—CRYPTO 2015—35th Annual Cryptology Conference, Santa Barbara, CA, USA, 16–20 August 2015; pp. 209–223.
7. Bellare, M.; Tackmann, B. The Multi-user Security of Authenticated Encryption: AES-GCM in TLS 1.3. In *Advances in Cryptology—CRYPTO 2016*; Robshaw, M., Katz, J., Eds.; Springer Berlin/Heidelberg: Berlin/Heidelberg, Germany, 2016; pp. 247–276.
8. Luykx, A.; Mennink, B.; Paterson, K.G. Analyzing Multi-key Security Degradation. In Proceedings of the Advances in Cryptology—ASIACRYPT 2017—23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017; pp. 575–605.
9. Gueron, S.; Langley, A.; Lindell, Y. AES-GCM-SIV: Specification and Analysis. Cryptology ePrint Archive, Report 2017/168, 2017. Available online: <https://eprint.iacr.org/2017/168> (accessed on 31 July 2019).
10. Rogaway, P.; Shrimpton, T. A Provable-Security Treatment of the Key-Wrap Problem. In *Advances in Cryptology—EUROCRYPT 2006*; Vaudenay, S., Ed.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 373–390.
11. Gueron, S.; Langley, A.; Lindell, Y. AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption. *RFC* **2019**, 8452, 1–42. doi:10.17487/RFC8452. [[CrossRef](#)]
12. Iyengar, J.; Thomson, M. *QUIC: A UDP-Based Multiplexed and Secure Transport*; Internet-Draft Draft-Ietf-Quic-Transport-20; Internet Engineering Task Force: Fremont, CA, USA, 2019.
13. Gueron, S.; Lindell, Y. Better Bounds for Block Cipher Modes of Operation via Nonce-Based Key Derivation. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, 30 October–3 November 2017; pp. 1019–1036, doi:10.1145/3133956.3133992. [[CrossRef](#)]
14. Bose, P.; Hoang, V.T.; Tessaro, S. Revisiting AES-GCM-SIV: Multi-user Security, Faster Key Derivation, and Better Bounds. In Proceedings of the Advances in Cryptology—EUROCRYPT 2018—37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, 39 April–3 May 2018; pp. 468–499.
15. Dworkin, M. *SP 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2007.
16. Suzuki, K.; Tonien, D.; Kurosawa, K.; Toyota, K. Birthday Paradox for Multi-collisions. In Proceedings of the 9th International Conference on Information Security and Cryptology, Busan, Korea, 30 November–1 December 2006; Springer-Verlag: Berlin/Heidelberg, Germany, 2006; pp. 29–40, doi:10.1007/11927587_5. [[CrossRef](#)]
17. Services, A.W. AWS Key Management Service (KMS), 2019. Available online: https://docs.aws.amazon.com/IAM/latest/UserGuide/iam-ug.pdf#access_policies (accessed on 31 August 2017).
18. Ramaswamy Chandramouli, M.I.; Chokhani, S. *Cryptographic Key Management Issues & Challenges in Cloud Services*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2013.
19. Chen, L. *SP 800-108. Recommendation for Key Derivation Using Pseudorandom Functions (Revised)*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2009.
20. Campagna, M. AWS Key Management Service Cryptographic Details, 2016. Available online: <https://d0.awsstatic.com/whitepapers/KMS-Cryptographic-Details.pdf> (accessed on 31 August 2018).