NIST Special Publication 800-38D
November, 2007

U.S. Department of Commerce

Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC

Morris Dworkin

C O M P U T E R S E C U R I T Y

NIST Special Publication 800-38D

Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC

Morris Dworkin

C O M P U T E R S E C U R I T Y

Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8930

November 2007

U.S. Department of Commerce
Carlos M. Gutierrez, Secretary

National Institute of Standards and Technology
James M. Turner, Acting Director

Reports on Information Security Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of technical, physical, administrative, and management standards and guidelines for the cost-effective security and privacy of sensitive unclassified information in Federal computer systems. This Special Publication 800-series reports on ITL's research, guidance, and outreach efforts in computer security, and its collaborative activities with industry, government, and academic organizations.

National Institute of Standards and Technology Special Publication 800-38D
Natl. Inst. Stand. Technol. Spec. Publ. 800-38D 37 pages (November 2007)
CODEN: NSPUE2

Acknowledgements

The author wishes to thank David McGrew, who co-invented GCM and submitted it to NIST, and also the author's colleagues who reviewed drafts of this document and contributed to its development, especially Elaine Barker, John Kelsey, Allen Roginsky, Donna Dodson, Tim Polk, and Bill Burr. The author also gratefully acknowledges the many comments from the public and private sectors to improve the quality and usefulness of this publication.

Abstract

This Recommendation specifies the Galois/Counter Mode (GCM), an algorithm for authenticated encryption with associated data, and its specialization, GMAC, for generating a message authentication code (MAC) on data that is not encrypted. GCM and GMAC are modes of operation for an underlying approved symmetric key block cipher.

KEY WORDS: authenticated encryption; authentication; block cipher; confidentiality, cryptography; encryption; information security; mode of operation.

Table of Contents

1 PURPOSE 1

2 AUTHORITY 1

3 INTRODUCTION 1

4 DEFINITIONS, ABBREVIATIONS, AND SYMBOLS 2

4.1 Definitions and Abbreviations 2
4.2 Symbols 5
4.2.1 Variables 5
4.2.2 Operations and Functions 6

5 ELEMENTS OF GCM 7

5.1 Block Cipher 7
5.2 Two GCM Functions 7
5.2.1 Authenticated Encryption Function 8
5.2.2 Authenticated Decryption Function 9
5.3 Primitives for Confidentiality and Authentication 9

6 MATHEMATICAL COMPONENTS OF GCM 10

6.1 Examples of Basic Operations and Functions on Strings 10
6.2 Incrementing Function 11
6.3 Multiplication Operation on Blocks 11
6.4 GHASH Function 12
6.5 GCTR Function 13

7 GCM SPECIFICATION 14

7.1 Algorithm for the Authenticated Encryption Function 14
7.2 Algorithm for the Authenticated Decryption Function 16

8 UNIQUENESS REQUIREMENT ON IVS AND KEYS 18

8.1 Key Establishment 19
8.2 IV Constructions 19
8.2.1 Deterministic Construction 20
8.2.2 RBG-based Construction 20
8.3 Constraints on the Number of Invocations 21

9 PRACTICAL CONSIDERATIONS FOR VALIDATING IMPLEMENTATIONS 22

9.1 Design Considerations 22
9.2 Operational Considerations 23

10 CONFORMANCE 24

APPENDIX A: IMPORTANCE OF THE UNIQUENESS REQUIREMENT ON IVS 25

APPENDIX B: AUTHENTICATION ASSURANCE 26

APPENDIX C: REQUIREMENTS AND GUIDELINES FOR USING SHORT TAGS 28

APPENDIX D: PROTECTION AGAINST REPLAY OF MESSAGES 30

APPENDIX E: BIBLIOGRAPHY 31

List of Figures

Figure 1: $\text{GHASH}_H (X_1 \| X_2 \| \ldots \| X_m) = Y_m$ 13
Figure 2: $\text{GCTR}_K (ICB, X_1 \| X_2 \| \ldots \| X_n^*) = Y_1 \| Y_2 \| \ldots \| Y_n^*$ 14
Figure 3: $\text{GCM-AE}_K (IV, P, A) = (C, T)$ 16
Figure 4: $\text{GCM-AD}_K (IV, C, A, T) = P$ or *FAIL* 18

List of Tables

Table 1: Constraints with 32-bit Tags 29
Table 2: Constraints with 64-bit Tags 29

1 Purpose

This publication is the fourth Part in a series of Recommendations regarding modes of operation of symmetric key block ciphers.

2 Authority

This document has been developed by the National Institute of Standards and Technology (NIST) in furtherance of its statutory responsibilities under the Federal Information Security Management Act (FISMA) of 2002, Public Law 107-347.

NIST is responsible for developing standards and guidelines, including minimum requirements, for providing adequate information security for all agency operations and assets, but such standards and guidelines shall not apply to national security systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), Securing Agency Information Systems, as analyzed in A-130, Appendix IV: Analysis of Key Sections. Supplemental information is provided in A-130, Appendix III.

This Recommendation has been prepared for use by federal agencies. It may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright. (Attribution would be appreciated by NIST.)

Nothing in this document should be taken to contradict standards and guidelines made mandatory and binding on federal agencies by the Secretary of Commerce under statutory authority. Nor should these guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other federal official.

Conformance testing for implementations of the mode of operation that is specified in this Part of the Recommendation will be conducted within the framework of the Cryptographic Module Validation Program (CMVP), a joint effort of NIST and the Communications Security Establishment of the Government of Canada. An implementation of a mode of operation must adhere to the requirements in this Recommendation in order to be validated under the CMVP. The requirements of this Recommendation are indicated by the word "shall."

3 Introduction

This Recommendation specifies an algorithm called Galois/Counter Mode (GCM) for authenticated encryption with associated data. GCM is constructed from an approved symmetric key block cipher with a block size of 128 bits, such as the Advanced Encryption Standard (AES) algorithm that is specified in Federal Information Processing Standard (FIPS) Pub. 197 [2]. Thus, GCM is a mode of operation of the AES algorithm.

GCM provides assurance of the confidentiality of data using a variation of the Counter mode of operation for encryption. GCM provides assurance of the authenticity of the confidential data (up to about 64 gigabytes per invocation) using a universal hash function that is defined over a

binary Galois (i.e., finite) field. GCM can also provide authentication assurance for additional data (of practically unlimited length per invocation) that is not encrypted.

If the GCM input is restricted to data that is not to be encrypted, the resulting specialization of GCM, called GMAC, is simply an authentication mode on the input data. In the rest of this document, statements about GCM also apply to GMAC.

GCM provides stronger authentication assurance than a (non-cryptographic) checksum or error detecting code; in particular, GCM can detect both 1) accidental modifications of the data and 2) intentional, unauthorized modifications.

The two functions of GCM are called authenticated encryption and authenticated decryption. Each of these functions is relatively efficient and parallelizable; consequently, high-throughput implementations are possible in both hardware and software. GCM has several other useful characteristics, including the following:

- The GCM functions are "online" in the sense that the lengths of the confidential data and the additional, non-confidential data are not required in advance; instead, the lengths can be calculated as the data arrives and is processed.

- The GCM functions require only the forward direction of the underlying block cipher (i.e., the inverse direction is not required).

- The authenticity of the protected data can be verified independently from the recovery of the confidential data from its encrypted form.

- If the unique initialization string is predictable, and the length of the confidential data is known, then the block cipher invocations within the GCM encryption mechanism can be pre-computed.

- If some or all of the additional, non-confidential data is fixed, then the corresponding elements of the GCM authentication mechanism can be pre-computed.

An important caution to the use of GCM is that a breach of the requirement in Sec. 8 for the uniqueness of the initialization strings may compromise the security assurance almost entirely, as detailed in Ref. [5] and summarized in Appendix A. Therefore, this mode of operation should not be deployed unless compliance with this uniqueness requirement is ensured. Some of the practical considerations are discussed further in Secs. 9.1 and 9.2.

The designers of GCM are McGrew and Viega. They submitted GCM to NIST in Ref. [6], and they discuss in detail its security and performance in Ref. [7].

4 Definitions, Abbreviations, and Symbols

4.1 Definitions and Abbreviations

AAD	Additional Authenticated Data
Additional Authenticated Data	The input data to the authenticated encryption function that is authenticated but not encrypted.
AES	Advanced Encryption Standard.
Approved	FIPS approved or NIST recommended: an algorithm or technique that is either 1) specified in a FIPS or a NIST Recommendation, or 2) adopted in a FIPS or a NIST Recommendation.
Authenticated Decryption	The function of GCM in which the ciphertext is decrypted into the plaintext, and the authenticity of the ciphertext and the AAD is verified.
Authenticated Encryption	The function of GCM in which the plaintext is encrypted into the ciphertext, and an authentication tag is generated on the AAD and the ciphertext.
Authentication Tag (Tag)	A cryptographic checksum on data that is designed to reveal both accidental errors and the intentional modification of the data.
Authenticity	The property that data originated from its purported source.
Bit	A binary digit: `0` or `1`.
Bit String	A finite, ordered sequence of bits.
Block	For a given block cipher, a bit string whose length is the block size of the block cipher.
Block Cipher	A parameterized family of permutations on bit strings of a fixed length; the parameter that determines the permutation is a bit string called the key.
Block Size	For a given block cipher and key, the fixed length of the input (or output) bit strings.
Byte	A sequence of 8 bits.
Byte String	A finite, ordered sequence of bytes.
Ciphertext	The encrypted form of the plaintext.

Direct Random String	In the RBG-based construction of IVs, an output string of an RBG that is used as the random field for an IV.
Exclusive-OR	The bitwise addition, modulo 2, of two bit strings of equal length.
FIPS	Federal Information Processing Standard.
Fixed Field	In the deterministic construction of IVs, the field that identifies the device or context for the instance of the authenticated encryption function.
Forward Cipher Function	A permutation on blocks that is determined by the choice of a key for a given block cipher.
Free Field	In the RBG-based construction of IVs, the field whose contents are not restricted.
Fresh	For a newly generated key, the property of being unequal to any previously used key.
GCM	Galois/Counter Mode
ICB	Initial Counter Block
Initialization Vector	A nonce that is associated with an invocation of authenticated encryption on a particular plaintext and AAD.
Inverse Cipher Function	The function that is the inverse of the forward cipher function for a given key.
Invocation Field	In the deterministic construction of IVs, the field that identifies the sets of inputs to the authenticated encryption function in a particular device or context.
IT	Information Technology
IV	Initialization Vector
Key	The parameter of the block cipher that determines the selection of the forward cipher function from the family of permutations.
Least Significant Bit(s)	The right-most bit(s) of a bit string.
Mode of Operation (Mode)	An algorithm for the cryptographic transformation of data that is based on a block cipher.

Most Significant Bit(s)	The left-most bit(s) of a bit string.
NIST	National Institute of Standards and Technology.
Nonce	A value that is used only once within a specified context.
Permutation	An invertible function.
Plaintext	The input data to the authenticated encryption function that is both authenticated and encrypted.
Random Field	In the RBG-based construction of IVs, either a direct random string or one of its successors.
RBG	Random Bit Generator
Successor	In the RBG-based construction of IVs, the result of one or more applications of the appropriate incrementing function to a direct random string.
XOR	Exclusive-OR.

4.2 Symbols

4.2.1 Variables

A The additional authenticated data

C The ciphertext.

H The hash subkey.

ICB The initial counter block

IV The initialization vector.

K The block cipher key.

P The plaintext.

R The constant within the algorithm for the block multiplication operation.

T The authentication tag.

t The bit length of the authentication tag.

4.2.2 Operations and Functions

$\mathtt{0}^s$	The bit string that consists of s '0' bits.
$\mathrm{CIPH}_K(X)$	The output of the forward cipher function of the block cipher under the key K applied to the block X.
$\mathrm{GCTR}_K(ICB, X)$	The output of the GCTR function for a given block cipher with key K applied to the bit string X with an initial counter block ICB.
$\mathrm{GHASH}_H(X)$	The output of the GHASH function under the hash subkey H applied to the bit string X.
$\mathrm{inc}_s(X)$	The output of incrementing the right-most s bits of the bit string X, regarded as the binary representation of an integer, by 1 modulo 2^s.
$\mathrm{int}(X)$	The integer for which the bit string X is a binary representation.
$\mathrm{len}(X)$	The bit length of the bit string X.
$\mathrm{LSB}_s(X)$	The bit string consisting of the s right-most bits of the bit string X.
$\mathrm{MSB}_s(X)$	The bit string consisting of the s left-most bits of the bit string X.
$\lceil x \rceil$	The least integer that is not less than the real number x.
$[x]_s$	The binary representation of the non-negative integer x as a string of s bits, where $x<2^s$.
$X >> 1$	The bit string that results from discarding the rightmost bit of the bit string X and prepending a '0' bit on the left.
$X \| Y$	The concatenation of two bit strings X and Y.
$X \oplus Y$	The bitwise exclusive-OR of two bit strings X and Y of the same length.
$X \bullet Y$	The product of two blocks, X and Y, regarded as elements of a certain binary Galois field.
X^i	For a positive integer i, the ith power of X under the product '•'.
$x \cdot y$	The product of two integers, x and y.

5 Elements of GCM

The elements of GCM and the associated notation and requirements are introduced in the three sections below. The underlying block cipher and key are discussed in Sec. 5.1. The data elements of the authenticated encryption and authenticated decryption functions of GCM are discussed in Sec. 5.2. The cryptographic primitives for confidentiality and authentication within these two functions are discussed in Sec. 5.3.

5.1 Block Cipher

The operations of GCM depend on the choice of an underlying symmetric key block cipher and thus can be considered a mode of operation (mode, for short) of the block cipher. The GCM key is the block cipher key (the key, for short).

For any given key, the underlying block cipher of the mode consists of two functions that are inverses of each other. The choice of the block cipher includes the designation of one of the two functions of the block cipher as the forward cipher function, as in the specification of the AES algorithm in Ref. [2]. GCM does not employ the inverse cipher function.

The forward cipher function is a permutation on bit strings of a fixed length; the strings are called blocks. The length of a block is called the block size. The key is denoted K, and the resulting forward cipher function of the block cipher is denoted CIPH_K.

The underlying block cipher shall be approved, the block size shall be 128 bits, and the key size shall be at least 128 bits. The key shall be generated uniformly at random, or close to uniformly at random, i.e., so that each possible key is (nearly) equally likely to be generated. Consequently, the key will be fresh, i.e., unequal to any previous key, with high probability. The key shall be secret and shall be used exclusively for GCM with the chosen block cipher. Additional requirements on the establishment and management of keys are discussed in Sec. 8.1.

5.2 Two GCM Functions

The two functions that comprise GCM are called authenticated encryption and authenticated decryption. The authenticated encryption function encrypts the confidential data and computes an authentication tag on both the confidential data and any additional, non-confidential data. The authenticated decryption function decrypts the confidential data, contingent on the verification of the tag.

An implementation may restrict the input to the non-confidential data, i.e., without any confidential data. The resulting variant of GCM is called GMAC. For GMAC, the authenticated encryption and decryption functions become the functions for generating and verifying an authentication tag on the non-confidential data.

The requirements and notation for the input and output data of these functions are discussed in Secs. 5.2.1 and 5.2.2. Algorithms for computing these functions are given in Sec. 7.

5.2.1 Authenticated Encryption Function

5.2.1.1 Input Data

Given the selection of an approved block cipher and key, there are three input strings to the authenticated encryption function:

- a plaintext, denoted P;
- additional authenticated data (AAD), denoted A; and
- an initialization vector (IV), denoted IV.

The plaintext and the AAD are the two categories of data that GCM protects. GCM protects the authenticity of the plaintext and the AAD; GCM also protects the confidentiality of the plaintext, while the AAD is left in the clear. For example, within a network protocol, the AAD might include addresses, ports, sequence numbers, protocol version numbers, and other fields that indicate how the plaintext should be treated.

The IV is essentially a nonce, i.e, a value that is unique within the specified context, which determines an invocation of the authenticated encryption function on the input data to be protected. The uniqueness requirement on the IVs (and keys) is stated precisely in Sec. 8, and two frameworks for constructing IVs are given in Sec. 8.2. Practical considerations in assuring the requirement are discussed in Secs. 9.1 and 9.2. The critical importance of the uniqueness of the IVs is detailed in Ref. [5] and summarized in Appendix A.

The bit lengths of the input strings to the authenticated encryption function shall meet the following requirements:

- $\text{len}(P) \le 2^{39}-256$;
- $\text{len}(A) \le 2^{64}-1$;
- $1 \le \text{len}(IV) \le 2^{64}-1$.

Although GCM is defined on bit strings, the bit lengths of the plaintext, the AAD, and the IV shall all be multiples of 8, so that these values are byte strings.

An implementation may further restrict the bit lengths of these inputs, consistent with the above requirements; for example, an implementation may establish smaller maximum values. The bit lengths that an implementation allows are called the supported bit lengths. A single set of supported bit lengths for each of the three inputs should be established for the entire implementation, independent of the key.

For IVs, it is recommended that implementations restrict support to the length of 96 bits, to promote interoperability, efficiency, and simplicity of design.

5.2.1.2 Output Data

The following two bit strings comprise the output data of the authenticated encryption function:

- A ciphertext, denoted *C*, whose bit length is the same as that of the plaintext.
- An authentication tag, or tag, for short, denoted *T*.

The bit length of the tag, denoted *t*, is a security parameter, as discussed in Appendix B. In general, *t* may be any one of the following five values: 128, 120, 112, 104, or 96. For certain applications, *t* may be 64 or 32; guidance for the use of these two tag lengths, including requirements on the length of the input data and the lifetime of the key in these cases, is given in Appendix C.

An implementation shall not support values for *t* that are different from the seven choices in the preceding paragraph. An implementation may restrict its support to as few as one of these values. A single, fixed value for *t* from among the supported choices shall be associated with each key.

5.2.2 Authenticated Decryption Function

Given the selection of an approved block cipher, key, and an associated tag length, the inputs to the authenticated decryption function are values for *IV*, *A*, *C*, and *T*, as described in Sec. 5.2.1 above. The output is one of the following:

- the plaintext *P* that corresponds to the ciphertext *C*, or
- a special error code, denoted *FAIL* in this document.

The output *P* indicates that *T* is the correct authentication tag for *IV*, *A*, and *C*; otherwise, the output is *FAIL*. The authentication assurance that can be inferred in each case is discussed in Appendix B.

The values for len(*C*), len (*A*), and len(*IV*) that an implementation supports for the authenticated decryption function shall be the same as the values for len(*P*), len (*A*), and len(*IV*) that the implementation supports for the authenticated encryption function.

5.3 Primitives for Confidentiality and Authentication

The mechanism for the confidentiality of the plaintext within GCM is a variation of the Counter mode [10], with a particular incrementing function, denoted inc_{32}, for generating the necessary sequence of counter blocks. The first counter block for the plaintext encryption is generated by incrementing a block that is generated from the IV.

The authentication mechanism within GCM is based on a hash function, called GHASH[1], that features multiplication by a fixed parameter, called the hash subkey, within a binary Galois field.

[1] The designers of GCM define GHASH somewhat differently, appending encodings of the lengths of its two arguments. In this Recommendation, these encodings are incorporated instead into the definitions of the authenticated encryption/decryption functions. The specifications of the authenticated encryption/decryption functions are ultimately equivalent, but the simplified version of GHASH in this Recommendation does not obtain all of the properties that are shown for the designers' definition of GHASH in Ref. [7].

The hash subkey, denoted H, is generated by applying the block cipher to the "zero" block. The resulting instance of this hash function, denoted GHASH_H, is used to compress an encoding of the AAD and the ciphertext into a single block, which is then encrypted to produce the authentication tag.

GHASH is a keyed hash function but not, on its own, a cryptographic hash function. This Recommendation only approves GHASH for use within the context of GCM.

The intermediate values in the execution of the GCM functions shall be secret. In particular, this requirement precludes a system in which GCM is implemented using the hash subkey publicly for some other purpose, for example, as an unpredictable value or as an integrity check value on the key.

6 Mathematical Components of GCM

This section presents the mathematical components that appear in the specifications of the authenticated encryption and authenticated decryption functions in Sec. 7 below. Examples of the basic operations and functions on bit strings are given in Sec. 6.1. The incrementing function is defined in Sec. 6.2. Algorithm 1 for "multiplying" blocks is defined in Sec. 6.3. Algorithm 2 for the GHASH function that is constructed from this multiplication is defined in Sec 6.4. Algorithm 3 for the GCTR function is defined in Sec. 6.5.

The specifications of Algorithms 1-3 include the inputs, the outputs, the steps of the algorithm, diagrams, and summaries. Equivalent sets of steps that produce the correct output are permitted. The inputs that are typically fixed across many invocations of the function are called the prerequisites, although they may also be regarded as (varying) inputs.

6.1 Examples of Basic Operations and Functions on Strings

In this document, the '`0`' bit and the '`1`' bit are indicated in the `new courier` font to help distinguish them from the integers 0 and 1.

Given a real number x, the ceiling function, denoted $\lceil x \rceil$, is the least integer that is not less than x. For example, $\lceil 2.1 \rceil = 3$, and $\lceil 4 \rceil = 4$.

Given a positive integer s, $\mathtt{0}^s$ denotes the string that consists of s '`0`' bits. For example, $\mathtt{0}^8 =$ `00000000`.

The concatenation operation on bit strings is denoted ||; for example, `001` || `10111` = `00110111`.

Given bit strings of equal length, the exclusive-OR (XOR) operation, denoted $\oplus$, specifies the addition, modulo 2, of the bits in each bit position, i.e., without carries. For example, `10011` $\oplus$ `10101` = `00110`.

Given a bit string X, the bit length of X is denoted len(X). For example, len(00010)=5.

Given a bit string X and a non-negative integer s such that len(X)≥s, the functions $\text{LSB}_s(X)$ and $\text{MSB}_s(X)$ return the s least significant (i.e., right-most) bits and the s most significant (i.e., left-most) bits, respectively, of X. For example, $\text{LSB}_3(111011010) = 010$, and $\text{MSB}_4(111011010) = 1110$.

Given a bit string X, the (single) right-shift function, denoted $X >> 1$, is $\text{MSB}_{\text{len}(X)}(0 \| X)$. For example, $0110111 >> 1 = 0011011$.

Given a positive integer s and a non-negative integer x that is less than 2^s, the integer-to-string function, denoted $[x]_s$, is the binary representation of x as a string of bit length s with the least significant bit on the right. For example, for the (base 10) integer 39, the binary representation (base 2) is 100111, so $[39]_8 = 00100111$.

Given a (non-empty) bit string X, the string-to-integer function, denoted int(X), is the integer x such that $[x]_{\text{len}(X)} = X$. In other words, int(X) is the non-negative integer less than $2^{\text{len}(X)}$ whose binary representation is X. For example, int(00011010) = 26.

6.2 Incrementing Function

For a positive integer s and a bit string X such that len(X)≥s, let the s-bit incrementing function, denoted $\text{inc}_s(X)$, be defined as follows:

$$\text{inc}_s(X) = \text{MSB}_{\text{len}(X)-s}(X) \| [\text{int}(\text{LSB}_s(X))+1 \bmod 2^s]_s$$

In other words, the function increments the right-most s bits of the string, regarded as the binary representation of an integer, modulo 2^s; the remaining, left-most len(X)-s bits remain unchanged.

6.3 Multiplication Operation on Blocks

Let R be the bit string $11100001 \| 0^{120}$. Given two blocks X and Y, Algorithm 1 below computes a "product" block, denoted $X \bullet Y$:

Algorithm 1: $X \bullet Y$

Input:
blocks X, Y.

Output:
block $X \bullet Y$.

Steps:
1. Let $x_0x_1...x_{127}$ denote the sequence of bits in X.
2. Let $Z_0 = 0^{128}$ and $V_0 = Y$.
3. For $i = 0$ to 127, calculate blocks Z_{i+1} and V_{i+1} as follows:

$$Z_{i+1} = \begin{cases} Z_i & \text{if } x_i = 0; \\ Z_i \oplus V_i & \text{if } x_i = 1. \end{cases}$$

$$V_{i+1} = \begin{cases} V_i >> 1 & \text{if } \mathrm{LSB}_1(V_i) = 0; \\ (V_i >> 1) \oplus R & \text{if } \mathrm{LSB}_1(V_i) = 1. \end{cases}$$

4. Return Z_{128}.

The • operation on (pairs of) the 2^{128} possible blocks corresponds to the multiplication operation for the binary Galois (finite) field of 2^{128} elements. The fixed block, R, determines a representation of this field as the modular multiplication of binary polynomials of degree less than 128. The convention for interpreting strings as polynomials is "little endian": i.e., if u is the variable of the polynomial, then the block $x_0x_1...x_{127}$ corresponds to the polynomial $x_0 + x_1 u + x_2 u^2 + ... + x_{127} u^{127}$. The XOR operation is used to add coefficients of "like" terms during the multiplication. The reduction modulus is the polynomial of degree 128 that corresponds to $R \| 1$. Ref. [6] discusses this field in detail.

For a positive integer i, the ith power of a block X with this multiplication operation is denoted X^i. For example, $H^2=H•H$, $H^3=H•H•H$, etc.

6.4 GHASH Function

Algorithm 2 below specifies the GHASH function:

Algorithm 2: $\mathrm{GHASH}_H(X)$

Prerequisites:
block H, the hash subkey.

Input:
bit string X such that $\mathrm{len}(X) = 128m$ for some positive integer m.

Output:
block $\mathrm{GHASH}_H(X)$.

Steps:
1. Let $X_1, X_2, ..., X_{m-1}, X_m$ denote the unique sequence of blocks such that $X = X_1 \| X_2 \| ... \| X_{m-1} \| X_m$.
2. Let Y_0 be the "zero block," $\mathtt{0}^{128}$.
3. For $i = 1, ..., m$, let $Y_i = (Y_{i-1} \oplus X_i) • H$.
4. Return Y_m.

In effect, the GHASH function calculates $X_1•H^m \oplus X_2•H^{m-1} \oplus ... \oplus X_{m-1}•H^2 \oplus X_m•H$. Ref. [6] describes methods for optimizing implementations of GHASH in both hardware and software.

The GHASH function is illustrated in Figure 1 below, without the zero block, Y_0, whose exclusive-OR with X_1 does not change X_1.

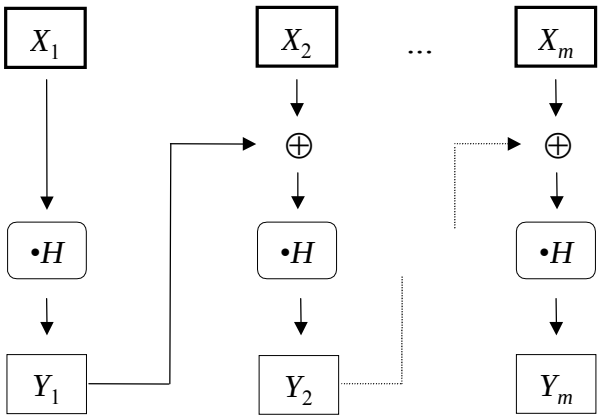

Figure 1: $\text{GHASH}_H (X_1 \| X_2 \| \ldots \| X_m) = Y_m$.

6.5 GCTR Function

Algorithm 3 below specifies the GCTR function. The suggested notation does not indicate the choice of the underlying block cipher.

Algorithm 3: $\text{GCTR}_K (ICB, X)$

Prerequisites:
approved block cipher CIPH with a 128-bit block size;
key K.

Input:
initial counter block ICB;
bit string X, of arbitrary length.

Output:
bit string Y of bit length $\text{len}(X)$.

Steps:
1. If X is the empty string, then return the empty string as Y.
2. Let $n = \lceil \text{len}(X)/128 \rceil$.
3. Let $X_1, X_2, \ldots, X_{n-1}, X_n^*$ denote the unique sequence of bit strings such that
   $X = X_1 \| X_2 \| \ldots \| X_{n-1} \| X_n^*$;
   $X_1, X_2, \ldots, X_{n-1}$ are complete blocks.[2]
4. Let $CB_1 = ICB$.
5. For $i = 2$ to n, let $CB_i = \text{inc}_{32}(CB_{i-1})$.
6. For $i = 1$ to $n-1$, let $Y_i = X_i \oplus \text{CIPH}_K(CB_i)$.
7. Let $Y_n^* = X_n^* \oplus \text{MSB}_{\text{len}(X_n^*)}(\text{CIPH}_K(CB_n))$.

[2] Consequently, X_n^* is either a complete block or a nonempty partial block, and if $1 \le \text{len}(X) \le 128$, then $X = X_1^*$.

8. Let $Y = Y_1 \| Y_2 \| ... \| Y_n^*$.
9. Return *Y*.

In Steps 1 and 2, the input string of arbitrary length is partitioned into a sequence of blocks to the greatest extent possible, so that only the rightmost string in the sequence may be a "partial" block. In Steps 3 and 4, the 32-bit incrementing function is iterated on the initial counter block input to generate a sequence of counter blocks; the input block is the first block of the sequence. In Steps 5 and 6, the block cipher is applied to the counter blocks and the results are XORed with the corresponding blocks (or partial block) of the partition of the input string. In Step 7, the sequence of results is concatenated to form the output.

Figure 2 below illustrates the GCTR function.

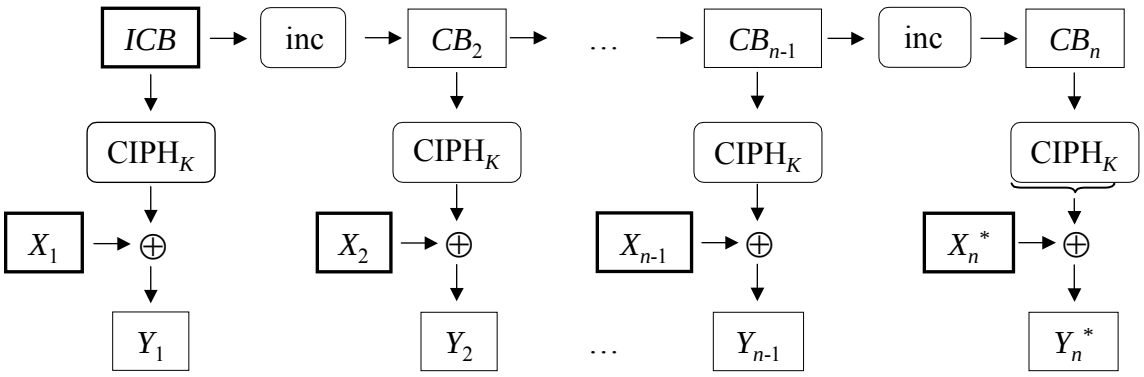

Figure 2: $\text{GCTR}_K (ICB, X_1 \| X_2 \| ... \| X_n^*) = Y_1 \| Y_2 \| ... \| Y_n^*$.

7 GCM Specification

Algorithms 4 and 5 for the authenticated encryption and authenticated decryption functions of GCM are specified in Secs. 7.1 and 7.2 below. The specifications include the inputs, the outputs, the steps of the algorithm, diagrams, and summaries. The suggested notation does not indicate the choice of the underlying block cipher. The inputs that are typically fixed across many invocations of the function are called the prerequisites; however, some of the prerequisites may also be regarded as (varying) input. The prerequisites and the other inputs shall meet the requirements in Sec. 5.

For both algorithms, equivalent sets of steps that produce the correct output are permitted. For example, in Algorithm 5, the verification of the tag may precede the computation of the plaintext.

7.1 Algorithm for the Authenticated Encryption Function

Algorithm 4 below specifies the authenticated encryption function:

Algorithm 4: GCM-AE$_K$ (IV, P, A)

Prerequisites:
approved block cipher CIPH with a 128-bit block size;
key K;
definitions of supported input-output lengths;
supported tag length t associated with the key.

Input:
initialization vector IV (whose length is supported);
plaintext P (whose length is supported);
additional authenticated data A (whose length is supported).

Output:
ciphertext C;
authentication tag T.

Steps:
1. Let $H = \text{CIPH}_K(0^{128})$.
2. Define a block, J_0, as follows:
   If len(IV)=96, then let $J_0 = IV \,||\, 0^{31} \,||\, 1$.
   If len(IV) $\neq$ 96, then let $s = 128\lceil \text{len}(IV)/128 \rceil - \text{len}(IV)$, and let
   $J_0 = \text{GHASH}_H(IV \,||\, 0^{s+64} \,||\, [\text{len}(IV)]_{64})$.
3. Let $C = \text{GCTR}_K(\text{inc}_{32}(J_0), P)$.
4. Let $u = 128 \cdot \lceil \text{len}(C)/128 \rceil - \text{len}(C)$ and let $v = 128 \cdot \lceil \text{len}(A)/128 \rceil - \text{len}(A)$.
5. Define a block, S, as follows:
   $$S = \text{GHASH}_H(A \,||\, 0^v \,||\, C \,||\, 0^u \,||\, [\text{len}(A)]_{64} \,||\, [\text{len}(C)]_{64}).$$
6. Let $T = \text{MSB}_t(\text{GCTR}_K(J_0, S))$.
7. Return (C, T).

In Step 1, the hash subkey for the GHASH function is generated by applying the block cipher to the "zero" block. In Step 2, the pre-counter block (J_0) is generated from the IV. In particular, when the length of the IV is 96 bits, then the padding string $0^{31}||1$ is appended to the IV to form the pre-counter block. Otherwise, the IV is padded with the minimum number of '0' bits, possibly none, so that the length of the resulting string is a multiple of 128 bits (the block size); this string in turn is appended with 64 additional '0' bits, followed by the 64-bit representation of the length of the IV, and the GHASH function is applied to the resulting string to form the pre-counter block. In Step 3, the 32-bit incrementing function is applied to the pre-counter block to produce the initial counter block for an invocation of the GCTR function on the plaintext. The output of this invocation of the GCTR function is the ciphertext.

In Steps 4 and 5, the AAD and the ciphertext are each appended with the minimum number of '0' bits, possibly none, so that the bit lengths of the resulting strings are multiples of the block size. The concatenation of these strings is appended with the 64-bit representations of the lengths of the AAD and the ciphertext, and the GHASH function is applied to the result to produce a single output block. In Step 6, this output block is encrypted using the GCTR function

with the pre-counter block that was generated in Step 2, and the result is truncated to the specified tag length to form the authentication tag. The ciphertext and the tag are returned as the output in Step 7.

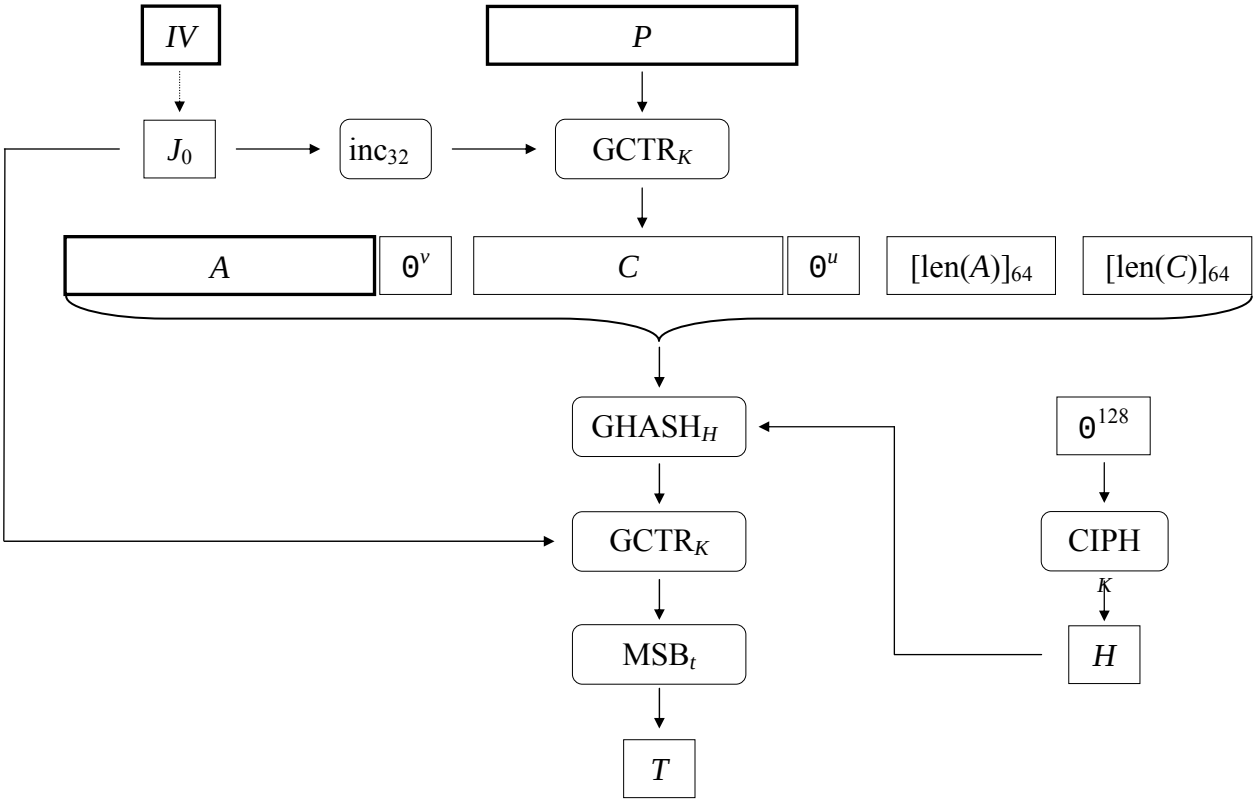

Figure 3: GCM-AE$_K$ (IV, P, A) = (C, T).

The authenticated encryption function is illustrated in Figure 3 above. The determination of J_0 from IV (Step 2) is not depicted.

7.2 Algorithm for the Authenticated Decryption Function

Algorithm 5 below specifies the authenticated decryption function:

Algorithm 5: GCM-AD$_K$ (IV, C, A, T)

Prerequisites:
approved block cipher CIPH with a 128-bit block size;
key K;
definitions of supported input-output lengths;
supported tag length t associated with the key.

Input:
initialization vector IV;
ciphertext C;

additional authenticated data *A*;
authentication tag *T*.

Output:
plaintext *P* or indication of inauthenticity *FAIL*.

Steps:

1. If the bit lengths of *IV*, *A* or *C* are not supported, or if len(*T*) ≠ *t*, then return FAIL.
2. Let $H = \text{CIPH}_K(0^{128})$.
3. Define a block, J_0, as follows:
   If len(*IV*)=96, then $J_0 = IV \,||\, 0^{31} \,||\, 1$.
   If len(*IV*) ≠ 96, then let $s = 128\lceil \text{len}(IV)/128 \rceil - \text{len}(IV)$, and
   $J_0 = \text{GHASH}_H(IV \,||\, 0^{s+64} \,||\, [\text{len}(IV)]_{64})$.
4. Let $P = \text{GCTR}_K(\text{inc}_{32}(J_0), C)$.
5. Let $u = 128 \cdot \lceil \text{len}(C)/128 \rceil - \text{len}(C)$ and let $v = 128 \cdot \lceil \text{len}(A)/128 \rceil - \text{len}(A)$.
6. Define a block, *S*, as follows:
   $S = \text{GHASH}_H(A \,||\, 0^v \,||\, C \,||\, 0^u \,||\, [\text{len}(A)]_{64} \,||\, [\text{len}(C)]_{64})$
7. Let $T' = \text{MSB}_t(\text{GCTR}_K(J_0, S))$.
8. If $T = T'$, then return *P*; else return *FAIL*.

In Step 1, the implementation's support for the lengths of the IV, the ciphertext, the AAD, and the authentication tag is verified. In Step 2, the hash subkey for the GHASH function is generated by applying the block cipher to the "zero" block. In Step 3, the pre-counter block (J_0) is formed as for the authenticated encryption function (Step 2 of Section 7.1). In Step 4, the 32-bit incrementing function is applied to the pre-counter block to produce the initial counter block for an invocation of the GCTR function on the ciphertext. The output of this invocation of the GCTR function is the plaintext that corresponds to the ciphertext for the given IV.

In Steps 5 and 6, the AAD and the ciphertext are each appended with the minimum number of '0' bits, possibly none, so that the bit lengths of the resulting strings are multiples of the block size. The concatenation of these strings is appended with 64-bit representations of the lengths of the AAD and the ciphertext, and the GHASH function is applied to the result to produce a single output block. In Step 7, this output block is encrypted using the GCTR function with the pre-counter block that was generated in Step 3, and the result is truncated to the specified tag length to form the authentication tag. In Step 8, the result of Step 7 is compared with the authentication tag that was received as an input: if they are identical, then the plaintext is returned; otherwise, *FAIL* is returned.

Equivalent sets of steps that produce the correct output are permitted. In particular, the verification of the tag may precede the computation of the plaintext.

The authenticated decryption function is illustrated in Figure 4 below. The determination of J_0 from *IV* (Step 3) is not depicted.

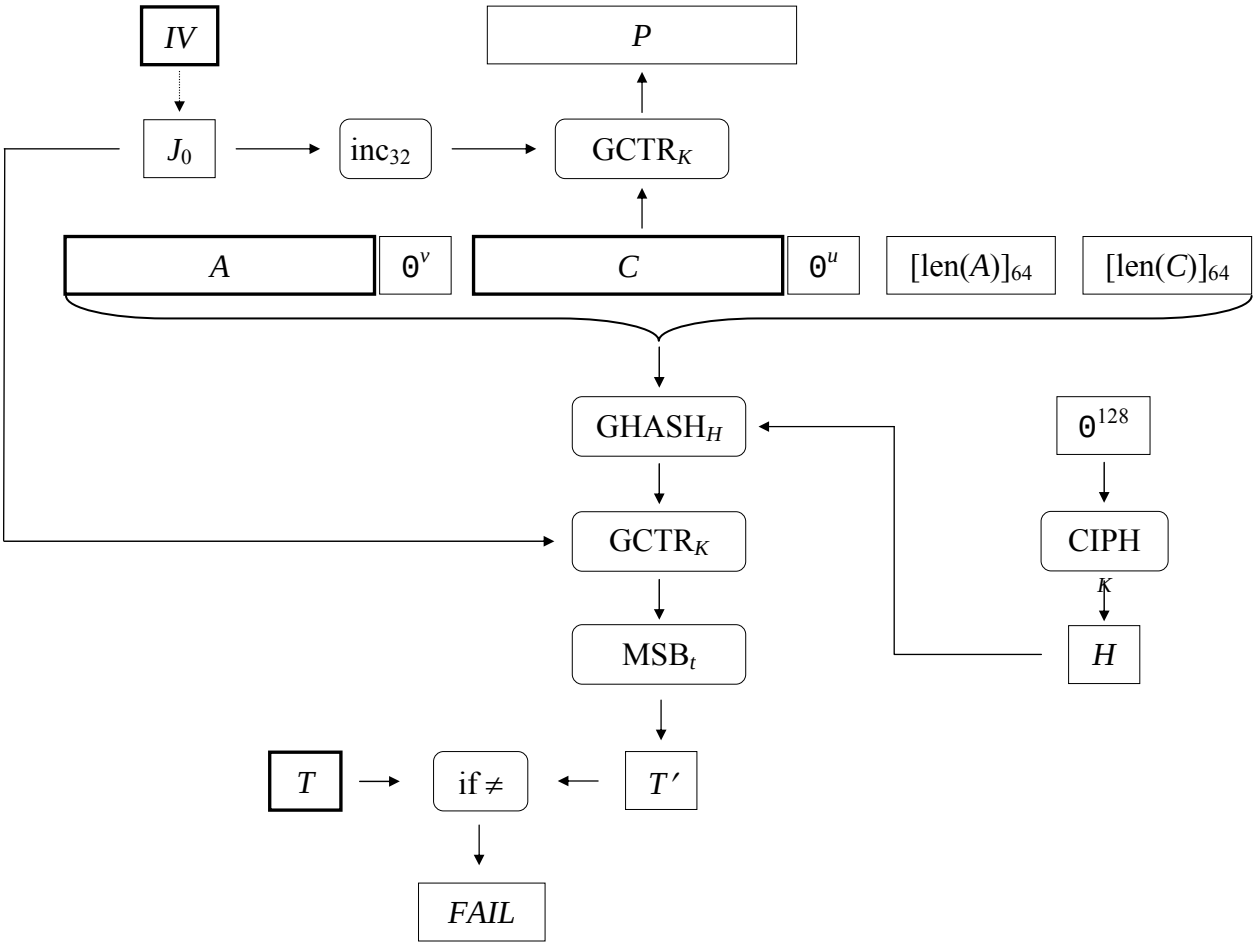

Figure 4: GCM-AD$_K$ (*IV*, *C*, *A*, *T*) = *P* or *FAIL*.

8 Uniqueness Requirement on IVs and Keys

The IVs in GCM must fulfill the following "uniqueness" requirement:

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}.

Compliance with this requirement is crucial to the security of GCM. Across all instances of the authenticated encryption function with a given key, if even one IV is ever repeated, then the implementation may be vulnerable to the forgery attacks that are described in Ref [5] and summarized in Appendix A. In practice, this requirement is almost as important as the secrecy of the key.

The role of key establishment in supporting this requirement is discussed in Sec. 8.1. The two allowed IV constructions for satisfying this requirement are given in Sec. 8.2. Constraints on the number of invocations of the authenticated encryption function are given in Sec. 8.3.

8.1 Key Establishment

The following requirement, which is the norm for secret key cryptographic algorithms in general, takes on explicit importance for GCM to support the uniqueness requirement in Sec. 8:

Any GCM key that is established among its intended users shall, with high probability, be fresh.

In practice, the requirements in Sec. 5.1 should ensure that a key is fresh when it is generated, if the generation mechanism is resistant to tampering. Achieving such resistance usually imposes requirements on the management of the key generation mechanism.

In particular, if the key generation mechanism is deterministic, then the management of the mechanism shall provide strong assurance that no outside entity can induce the repetition of a previous set of inputs to the mechanism, or otherwise cause the repetition of a previous output. For example, GCM keys may be established using the key derivation functions of the following protocols as allowed in [9]: Transport Layer Security, Internet Key Exchange v1 and v2, and Secure Shell.

Similarly, if a new key must be transported to its intended recipient(s), the method of transport/distribution shall provide strong assurance against "replay," so that no party can induce the substitution of a previous key for the intended key.

GCM keys should be established within the framework of an approved key management structure to assure their freshness, as well as their confidentiality and authenticity; the details of such structures are outside the scope of this Recommendation.

8.2 IV Constructions

This Recommendation provides two frameworks for constructing IVs. The first construction, described in Sec. 8.2.1, relies on deterministic elements to achieve the uniqueness requirement in Sec. 8; the second construction, described in Sec. 8.2.2, relies on a sufficiently long output string from an approved RBG with a sufficient security strength.

For any supported IV length that is strictly less than 96 bits, the construction in Sec. 8.2.1, shall be used, across all instances of the authenticated encryption function with the given key.

For any supported IV length that is 96 bits or greater, exactly one of the constructions, but not both, shall be used, across all instances of the authenticated encryption function with the given key.

For example, suppose that an implementation supports IV lengths of 64 bits, 96 bits, 128 bits, and 160 bits. For 64-bit IVs the only choice is the construction in Sec. 8.2.1. For the other three IV lengths, one possible combination of choices is the construction in Sec. 8.2.1 for 96-bit IVs and the construction in Sec. 8.2.2 for 128-bit and 160-bit IVs.

8.2.1 Deterministic Construction

In the deterministic construction, the IV is the concatenation of two fields, called the fixed field and the invocation field. The fixed field shall identify the device, or, more generally, the context for the instance of the authenticated encryption function. The invocation field shall identify the sets of inputs to the authenticated encryption function in that particular device.

For any given key, no two distinct devices shall share the same fixed field, and no two distinct sets of inputs to any single device shall share the same invocation field. Compliance with these two requirements implies compliance with the uniqueness requirement on IVs in Sec. 8.

If desired, the fixed field itself may be constructed from two or more smaller fields. Moreover, one of those smaller fields could consist of bits that are arbitrary (i.e., not necessarily deterministic nor unique to the device), as long as the remaining bits ensure that the fixed field is not repeated in its entirety for some other device with the same key.

Similarly, the entire fixed field may consist of arbitrary bits when there is only one context to identify, such as when a fresh key is limited to a single session of a communications protocol. In this case, if different participants in the session share a common fixed field, then the protocol shall ensure that the invocation fields are distinct for distinct data inputs.

The invocation field typically is either 1) an integer counter or 2) a linear feedback shift register that is driven by a primitive polynomial to ensure a maximal cycle length. In either case, the invocation field increments upon each invocation of the authenticated encryption function.

The lengths and positions of the fixed field and the invocation field shall be fixed for each supported IV length for the life of the key. In order to promote interoperability for the default IV length of 96 bits, this Recommendation suggests, but does not require, that the leading (i.e., leftmost) 32 bits of the IV hold the fixed field; and that the trailing (i.e., rightmost) 64 bits hold the invocation field.

8.2.2 RBG-based Construction

In the RBG-based construction, the IV is the concatenation of two fields, called the random field and the free field. For each IV length that is supported by the implementation and used with the RBG-based construction, the lengths of these fields shall be fixed for the life of the key. Moreover, the length of the random field shall be at least 96 bits; the free field may be empty.

If i is a supported IV length that is associated to the RBG-based construction, then let $r(i)$ denote the bit length of the random field. The random field shall either consist of 1) an output string of $r(i)$ bits from an approved RBG with a sufficient security strength, or 2) the result of applying the $r(i)$–bit incrementing function to the random field of the preceding IV for the given key. The $r(i)$–bit output string from the RBG is called a direct random string, and the random fields that result from applying the $r(i)$–bit incrementing function are called its successors.

There are no requirements on the bits in the free field. For example, they may identify the device, similar to the fixed field of the deterministic construction, except within the RBG-based construction these identifiers are not required to be distinct for each device. For any IV length that is associated to the RBG-construction, the free field is recommended to be empty, so that the random field is the entire IV.

The instantiations of the RBGs in any two distinct devices shall be independent, so that the distribution of direct random strings across all of the RBG instantiations is expected to be uniform. For example, if the initialization of the RBG instantiations depends only on a secret seed, then each instantiation shall be initialized with a distinct seed.

8.3 Constraints on the Number of Invocations

The following requirement applies to all implementations that use either 1) the deterministic construction with IVs whose length is *not* 96, or 2) the RBG-based construction, for IVs of any length. In other words, unless an implementation *only* uses 96-bit IVs that are generated by the deterministic construction:

The total number of invocations of the authenticated encryption function shall not exceed 2^{32}, including all IV lengths and all instances of the authenticated encryption function with the given key.

This is a "global" requirement that can be achieved by appropriate "local" limits on each instance of the authenticated encryption function with a given key. For example, suppose an implementation consists of 2^{10} devices that only support 64-bit, 96-bit, and 128-bit IVs. One way to satisfy the above requirement would be to limit each device to 2^{20} invocations with 64-bit IVs, 2^{21} invocations with 96-bit IVs, and 2^{20} invocations with 128-bit IVs.

For the RBG-based construction of IVs, the above requirement, in conjunction with the requirement that $r(i) \geq 96$, is sufficient to ensure the uniqueness requirement in Sec. 8, as follows from the discussion in Ref. [4].

For the deterministic construction, the lengths of the two fields imply two additional operational constraints. These constraints apply to any supported IV length, including 96 bits:

- The bit length of the invocation field limits the number of invocations of the authenticated encryption function with any given fixed field and key. In particular, if s denotes the number of bits in the invocation field, then the authenticated encryption function cannot be invoked on more than 2^s distinct input sets without violating the uniqueness requirement.

- Similarly, an s-bit fixed field implies a limit of 2^s on the number of distinct devices/contexts that can implement the authenticated encryption function for the given key, with IVs of the given length.

9 Practical Considerations for Validating Implementations

Both the designer of a GCM implementation and the information technology (IT) professional who deploys and maintains it within a particular system have important roles in meeting the uniqueness requirement in Sec. 8, as discussed in Secs. 9.1 and 9.2 below.

The additional requirements in these two sections are provided for the purpose of demonstrating compliance with the uniqueness requirement in Sec. 8 within a validation program. Specifically, analogous to the requirements in Ref. [9], the requirements in these two sections apply to implementations that are validated against the requirements of FIPS Pub. 140-2 (Ref. [3]), or any superseding version of FIPS 140.

Implementations that are not validated against the requirements of FIPS Pub. 140-2 may interpret the requirements in Secs. 9.1 and 9.2 as recommendations.

9.1 Design Considerations

In order to inhibit an unauthorized party from controlling or influencing the generation of IVs, GCM shall be implemented only within a cryptographic module that meets the requirements of FIPS Pub. 140-2. In particular, the cryptographic boundary of the module shall contain a "generation unit" that produces IVs according to one of the constructions in Sec. 8.2 above.

The documentation of the module for its validation against the requirements of FIPS 140-2 shall describe how the module complies with the uniqueness requirement on IVs. At a minimum, the documentation shall address the considerations in this section, and clearly document the responsibilities of the IT professional who configures, deploys, and maintains the GCM implementations within a larger system.

The following are three important design considerations for GCM modules:

1. The freshness of keys shall be assured, as discussed in Sec. 8.1.

2. The IV shall be a critical security parameter as defined in FIPS Pub. 140-2 until the authenticated encryption function is invoked with the IV. Prior to this invocation, the IV shall be provided the same protection as other critical security parameters in a module that is validated to the requirements in FIPS Pub. 140-2.

3. A loss of power to the module shall not cause the repetition of IVs. If the generation unit cannot recover from a loss of power, then the authenticated encryption function shall enter a failure state until a fresh key can be established.

The IV construction that is implemented from Sec. 8.2 above affects the options for recovery from a loss of power. For the deterministic construction, all of the deterministic elements that are necessary to construct the IV would have to be available when power is restored. For example, these elements could be stored in non-volatile memory.

When power is restored, neither the preceding IV nor any other previous IV shall immediately be repeated for the key. One way to avoid such a repetition would be to ensure that the invocation field value that is periodically stored in the non-volatile memory is always one or more values ahead of the operational value in the sequence.

One potential advantage of the RBG-based construction is that the RBG may be designed to recover from a loss of power in a straightforward manner, i.e., without requiring action from the IT professional that maintains the system. For example, the RBG may incorporate a non-deterministic source of bits that would automatically be available to the RBG when power is restored.

Alternatively, the entire state of the RBG may be stored periodically in non-volatile memory. In this case, similar to the deterministic construction, when power is restored, the design shall ensure that the RBG shall not output strings for use within new IVs until the state of the RBG is advanced beyond the state that generated the last IV for the key. In other words, the direct random string for the first new IV shall be, with high probability, different than the direct random string in any IV that was generated before the loss of power.

Even if the process is not automatic, the IT professional who maintains the system may simply reinitialize the RBG when power is restored, for example, with a fresh seed. In this case, either the design of the module or the IT professional that maintains the system shall ensure that compliance is maintained with the independence condition on the RBGs in Sec. 8.2.2.

9.2 Operational Considerations

Compliance with the uniqueness requirement on IVs, and hence the security of GCM, ultimately depends on the IT professional who configures, deploys, and maintains the GCM modules within a particular system. The documentation for a GCM module shall give the IT professional detailed instructions that are tailored to the particular design of the module.

The following are some typical operational considerations for the uniqueness requirement:

- Is the configuration of any GCM module, or any operational value, vulnerable to control or influence from any unauthorized party?

- For any given key, how are configuration choices enforced across all modules that ever implement the authenticated encryption function?

- How is the freshness of keys assured, as discussed in Sec. 8.1?

- If an implementation does not exclusively use 96-bit IVs that are generated by the deterministic construction, how is the requirement in Sec 8.3 on the number of invocations ensured?

- As discussed in Sec. 9.1, how does IV generation within the modules recover from a loss of power without violating the uniqueness requirement on IVs?

The following considerations are specific to the deterministic construction:

- How are the device identifiers installed into the fixed field so that compliance with the requirements on the fixed field in Sec. 8.2.1 is ensured, both for the initially deployed modules and for any subsequent deployed modules?

- Is the length of the invocation field sufficient to support all of the invocations that can occur in any module during the lifetime of any key, as discussed in the first bullet in Sec. 8.3?

- For any given key and IV length, is the length of the fixed field sufficient to support the number of modules that will implement authenticated encryption, as discussed in the second bullet in Sec. 8.3?

- If the default setting for the lengths of the fixed field and the invocation field can be altered, then how is the choice enforced across all modules with any given key?

The following consideration is specific to the RBG-based construction:

- How are the RBGs for IV generation initialized so that compliance is ensured with the independence requirement in Sec. 8.2.2, for the initially deployed modules and for any subsequent deployed modules?

10 Conformance

The two categories of implementations that may claim conformance with this Recommendation are GCM and GMAC.

Implementations may restrict the bit lengths of the plaintext, AAD, IVs, and authentication tags that it supports, as discussed in Secs. 5.2.1.1 and 5.2.1.2 above. Although such restrictions may affect interoperability, only the elimination of the plaintext altogether, i.e., GMAC, is considered as a distinct category of implementation.

For every algorithm that is specified in this Recommendation, a conforming implementation may replace the given set of steps with any mathematically equivalent sets of steps. In other words, different procedures that produce the correct output for any input are permitted.

Appendix A: Importance of the Uniqueness Requirement on IVs

Sec. 8 contains a uniqueness requirement for the IVs of GCM, across all implementations of the authenticated encryption function with any given key. This appendix summarizes why this requirement is crucial to the security of GCM, as first described in Ref. [5].

The secure operation of a cryptographic algorithm depends not only on the proper implementation of the steps of the algorithm, but also on the adherence to the associated requirements. An obvious example for any symmetric key algorithm is that the key needs to be kept secret. For GCM, the requirement in Sec. 5.2.1 for the uniqueness of the IVs is almost as important, although not as obvious.

Consider first the example of the Counter mode requirement in Ref. [10] for the counter blocks to be unique. If this requirement is breached, then the Counter mode encryption function may fail to provide the expected confidentiality for the data blocks that correspond to the repeating counter blocks, although the other data blocks are not affected.

The Counter mode decryption function has a vulnerability that is arguably more serious. Counter-mode ciphertext is "malleable" in the sense that flipping any of its bits will induce the corresponding bits of the plaintext to flip upon decryption. Therefore, if an adversary knows a plaintext-ciphertext pair and can induce the decryption function to use the counter blocks from the known pair, then, by choosing which bits to flip in the known ciphertext, the adversary can control the result of decryption, up to the length of the known plaintext.

This vulnerability motivates combining the Counter mode encryption mechanism with an authentication mechanism, such as occurs in GCM. The idea is to prevent the adversary's altered ciphertext from being accepted as valid. However, as detailed in Ref. [5], the authentication assurance in GCM crucially depends on the uniqueness of the IVs.

In particular, if IVs are ever repeated for the GCM authenticated encryption function for a given key, then it is likely that an adversary will be able to determine the hash subkey from the resulting ciphertexts. The adversary then could easily construct a ciphertext forgery. In particular, given the ciphertext and tag outputs for an invocation of the authenticated encryption function, along with the associated IV and AAD inputs, the adversary could substitute any ciphertext and AAD strings and use the subkey to generate the valid substitute tag. Although this forgery would use the same IV, in most cases it will not be feasible for the authenticated decryption function to detect the repetition. Thus, the authentication assurance essentially is lost.

Worse, the loss of authentication means that GCM inherits the problematic malleability of its Counter mode ciphertext. In the above scenario, knowing the original plaintext that corresponds to the given ciphertext, the adversary can construct the substitute ciphertext in such a way as to correspond to any desired plaintext of the same length. In other words, the adversary essentially could control the plaintext output of the authenticated decryption function.

Appendix B: Authentication Assurance

The creation of an authentication tag by the authenticated encryption function provides the mechanism whereby assurance of the authenticity of the plaintext and AAD (and IV) can be obtained upon the execution of the authenticated decryption function. The nature of this assurance depends on the output of the authenticated decryption function:

- If the output is the plaintext, then the design of the mode provides strong, but not absolute, assurance that the purported source of the data created the tag, i.e., that the plaintext and the AAD (and the IV and the tag) are authentic. Consequently, the mode also provides strong assurance that this information was not subsequently altered, either intentionally or unintentionally.

- If the output is *FAIL*, then it is certain that at least one of the given inputs (i.e., the ciphertext, the AAD, the IV, or the tag) is not authentic.

In the first case, the assurance is not absolute because forgeries are possible, in principle. In other words, an adversary, i.e., a party without access to the key or to the authenticated encryption function, may be able to produce the correct tag for some triple of ciphertext, AAD, and IV.

As with any tag-based authentication mechanism, if the adversary chooses a t-bit tag at random, it is expected to be correct for given data with probability $1/2^t$. With GCM, however, an adversary can choose tags that increase this probability, proportional to the total length of the ciphertext and AAD. Consequently, GCM is not well-suited for use with short tag lengths or very long messages.

In particular, if n denotes the total number of blocks in the encoding (i.e., the input to the GHASH function in the definition of S in Secs. 7.1 and 7.2 above) of the ciphertext and AAD, then there is a method of constructing a "targeted" ciphertext forgery that is expected to succeed with a probability of approximately $n/2^t$. Moreover, each successful forgery in this attack 1) increases the probability that subsequent targeted forgeries will succeed, and 2) leaks information about the hash subkey, H. Eventually, H may be compromised entirely, with consequences as described at the end of Appendix A: the authentication assurance is completely lost.

Ref. [1] describes the attack, and Ref. [7] gives a detailed treatment of the security properties of GCM. Appendix C gives guidance and requirements for the use of the two shortest tags sizes, 32 and 64 bits.

Independent of this attack, an adversary may attempt to systematically guess many different tags for a given input to authenticated decryption, and thereby increase the probability that one (or more) of them, eventually, will be accepted as valid. For this reason, the system or protocol that implements GCM should monitor and, if necessary, limit the number of unsuccessful verification attempts for each key.

Moreover, as with most block cipher modes of operation, the security assurance of GCM degrades as more data is processed with a single key. Therefore, the total number of blocks of plaintext and AAD that are protected by invocations of the authenticated encryption function during the lifetime of the key should be limited. A reasonable limit for most applications would be 2^{64}, consistent with the requirement on the number of invocations in Sec. 8.3.

Appendix C: Requirements and Guidelines for Using Short Tags

For some voice or video applications, short authentication tags can be appropriate. The forgery of some fraction of the individual authenticated "packets" may be tolerable, because each packet of data in a large stream may carry very little of the overall meaning.

However, even for voice and video applications, short tags can be problematic for GCM, due to the targeted forgery attack that is summarized in Appendix B and detailed in Ref. [1]. Absent the requirements and guidelines in this appendix, it may be practical for the attack to produce the hash subkey, H, after which the authentication assurance is completely lost. Nevertheless, this Recommendation does not preclude short tags entirely, because knowledgeable security professionals should be able to manage the risks in connection with this attack, and its potential improvement.

The following guidelines implicitly describe the special circumstances that may be appropriate for short tags. A packet in this context is a set of inputs to the authenticated decryption function.

1. Packets that fail the integrity check within the authenticated decryption function should be silently discarded. In other words, the controlling protocol/system over which packets are received should not provide an ACK/NACK response regarding the integrity of individual packets. However, the receiver should log authentication errors internally—in a way which is undetectable from side-channel information—and terminate the connection or notify the user if the percentage of errors exceeds what would be considered normal. This is standard security practice with any protocol/system and any algorithm choice.

2. The AAD within packets should be limited to the necessary header information and should not contain messages to be authenticated along with the encrypted data.

3. The substance or meaning of the overall message that is comprised of a large number of packets should not be lost or compromised by the forgery of a single, arbitrary packet. For example, packets could carry a sequence of snippets of voice or visual data, but an individual packet should not carry a .txt or .doc file. Ideally, the plaintext data underlying the encryption should not be so stereotypical as to be guessable.

4. The controlling protocol/system should establish a new GCM key—and thus a new hash subkey, H—frequently, depending on the maximum combined length of the ciphertext and AAD that can occur within a single packet. Moreover, for 32-bit tags, this combined length should be very small—on the order of tens or hundreds of bytes; 64-bit tags extend the maximum combined length into the millions of bytes.

An example of a protocol that meets these guidelines is Secure Real-time Transport Protocol carrying Voice over Internet Protocol, running over User Datagram Protocol.

Tables 1 and 2 below quantify the recommendations in Item 4 above, for 32-bit and 64-bit tags, respectively. Each row has two entries: 1) a maximum combined length for the ciphertext and the AAD in a single packet, in bytes, and 2) a corresponding maximum number of invocations of the authenticated decryption function, across all instances of the GCM with the given key.

For any implementation that supports 32-bit or 64-bit tags, one of the rows in Table 1 or Table 2, respectively, shall be enforced. In particular, the supported lengths for the plaintext/ciphertext and the AAD shall ensure that every valid packet satisfies the length restriction in the row, and the controlling protocol/system shall ensure that the key is changed before the authenticated decryption function is invoked more than the maximum that is given in the row. A smaller maximum may also be enforced.

Table 1: Constraints with 32-bit Tags

Maximum Combined Length of the Ciphertext and AAD In a Single Packet (bytes)	**Maximum Invocations of the Authenticated Decryption Function**
2^5	2^{22}
2^6	2^{20}
2^7	2^{18}
2^8	2^{15}
2^9	2^{13}
2^{10}	2^{11}

Table 2: Constraints with 64-bit Tags

Maximum Combined Length of the Ciphertext and AAD In a Single Packet (bytes)	**Maximum Invocations of the Authenticated Decryption Function**
2^{15}	2^{32}
2^{17}	2^{29}
2^{19}	2^{26}
2^{21}	2^{23}
2^{23}	2^{20}
2^{25}	2^{17}

Appendix D: Protection Against Replay of Messages

As described in Appendix B, the successful verification of the tag within the authenticated decryption function gives assurance of the authenticity of the data; however, the party that presents the input data to the authenticated decryption function may not be the *original* source of the plaintext and AAD. In other words, GCM, like many other authentication mechanisms, does not inherently prevent an adversary from intercepting the output of an invocation of authenticated encryption and "replaying" it for authenticated decryption at a later time, for example, in an attempt to impersonate a party that has access to the key. In some protocols an adversary may even be able to use data that the verifier itself generated earlier in the protocol.

The controlling system or protocol may protect against such an event by monitoring for any duplication of the IVs that are presented for authenticated decryption. Alternatively, certain identifying information can be incorporated into the AAD. Examples of such information include a sequential message number or a timestamp. Upon successful verification of authenticity, this information may provide a means for the detection of replayed messages, out-of-sequence messages, or missing messages.

Appendix E: Bibliography

[1] Ferguson, N., Authentication Weaknesses in GCM, Natl. Inst. Stand. Technol. [Web page], http://www.csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/Ferguson2.pdf, May 20, 2005.

[2] FIPS Publication 197, The Advanced Encryption Standard (AES), U.S. DoC/NIST, November 26, 2001.

[3] FIPS Publication 140-2, Security Requirements for Cryptographic Modules, U.S. DoC/NIST, May 25, 2001.

[4] IEEE P1619.1™/D23, Draft Standard for Authenticated Encryption with Length Expansion for Storage Devices.

[5] A. Joux, Authentication Failures in NIST version of GCM, Natl. Inst. Stand. Technol. [Web page], http://www.csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/800-38_Series-Drafts/GCM/Joux_comments.pdf.

[6] D. McGrew, J. Viega, The Galois/Counter Mode of Operation (GCM), Natl. Inst. Stand. Technol. [Web page], http://www.csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-revised-spec.pdf, May 31, 2005.

[7] D. McGrew and J. Viega. The Security and Performance of the Galois/Counter Mode (GCM) of Operation. Proceedings of INDOCRYPT '04, Springer-Verlag, 2004. Full paper available from the IACR Cryptology ePrint Archive: Report 2004/193, [Web page], http://eprint.iacr.org/2004/193/, October 7, 2004.

[9] National Institute of Standards and Technology and Communications Security Establishment, Implementation Guidance for FIPS Pub. 140-2 and the Cryptographic Module Validation Program, Natl. Inst. Stand. Technol. [Web page], http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf.

[10] NIST Special Publication 800-38A, 2001 ED, Version 1, Recommendation for Block Cipher Modes of Operation—Methods and Techniques, December 2001, Natl. Inst. Stand. Technol. [Web page], http://www.csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf.