# Consideraciones

## GENERALES

* Buscar método de migración de C a Verilog.
* La llave expira con cada sesión (modelo OSI).
* Conseguir implementaciones de AES ya realizadas en Verilog.
* Armar un listado de “TO DO”.
* Armar un diagrama de Gantt.
* Documentar todo.
* El rendimiento de AES-GCM se puede mejorar logrando la multiplicación por el GF más óptima.
* La key viene de capas superiores.
* Security in GCM (fuente Wikipedia, fuente de Wikipedia, entre otros el nist sp de gcm)

GCM has been proven secure in the concrete security model.[15] It is secure when it is used with a block cipher that is indistinguishable from a random permutation; however, security depends on choosing a unique initialization vector for every encryption performed with the same key (see stream cipher attack). For any given key and initialization vector combination, GCM is limited to encrypting 239 − 256 bits of plain text (64 GiB). NIST Special Publication 800-38D[16] includes guidelines for initialization vector selection.

The authentication strength depends on the length of the authentication tag, as with all symmetric message authentication codes. The use of shorter authentication tags with GCM is discouraged. The bit-length of the tag, denoted t, is a security parameter. 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, but the use of these two tag lengths constrains the length of the input data and the lifetime of the key. Appendix C in NIST SP 800-38D provides guidance for these constraints (for example, if t = 32 and the maximal packet size is 210 bytes, the authentication decryption function should be invoked no more than 211 times; if t = 64 and the maximal packet size is 215 bytes, the authentication decryption function should be invoked no more than 232 times).

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

Ferguson and Saarinen independently described how an attacker can perform optimal attacks against GCM authentication, which meet the lower bound on its security. Ferguson showed that, if n denotes the total number of blocks in the encoding (the input to the GHASH function), then there is a method of constructing a targeted ciphertext forgery that is expected to succeed with a probability of approximately n⋅2−t. If the tag length t is shorter than 128, then each successful forgery in this attack increases the probability that subsequent targeted forgeries will succeed, and leaks information about the hash subkey, H. Eventually, H may be compromised entirely and the authentication assurance is completely lost.[17]

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.

Saarinen described GCM weak keys.[18] This work gives some valuable insights into how polynomial hash-based authentication works. More precisely, this work describes a particular way of forging a GCM message, given a valid GCM message, that works with probability of about n⋅2−128 for messages that are n × 128 bits long. However, this work does not show a more effective attack than was previously known; the success probability in observation 1 of this paper matches that of lemma 2 from the INDOCRYPT 2004 analysis (setting w = 128 and l = n × 128). Saarinen also described a GCM variant Sophie Germain Counter Mode (SGCM), continuing the GCM tradition of including a mathematician in the name of the mode.

## SOBRE AES (FIPS 197.- Pub. 2001)

### Implementaciones:

* Puede ser implementado en HW, SW o una combinación de ambas.
* Si es testeado por un laboratorio acreditado y validado, entonces se considerará que cumple con el estándar.  
  NIST Special Publication 800-21 “Guia para Implementar Criptografía en el Gobierno Federal”.

### Calificaciones:

* NIST reevalúa el estándar cada 5 años.

### Definiciones:

#### Glosario de Términos y Acrónimos

* Transformación Afín (Affine transformation): Es una transformación que consiste en una multiplicación por una matriz, seguido de la adición de un vector.
* Llave Criptográfica (Cipher Key): Llave secreta,criptográfica que es una por la rutina “Expansión de Llave (Key Expansion)” para generar un set de “Llaves de Ronda (Round Keys)”; puede visualizarse como un array rectangular de bytes, teniendo 4 filas y Nk columnas.
* Llaves de Ronda (Round Keys): Son valores derivado de la Cipher Key usando la rutina Key Expansion; ellas son aplicadas a “Estado (State)” en el Cifrado y en el Desifrado.
* Estado (State): Cifrado intermedio resultado que puede verse como una array rectangular de bytes, teniendo 4 filas y Nb columnas.
* S-box: Tabla de sustitución no lineal usada en varias transformaciones de sustitución de bytes y en la rutina Key Expansion para realizar una sustitución uno por uno del valor de un byte.

#### Parámetros algorítmicos, Símbolos y Funciones

* K : Llave de cifrado (Cipher Key)
* Nb: Numero de columnas (32-bit words) que conforman el State. Para este estándar Nb = 4.
* Nk: Numero de palabras de 32-bits que conformar la Cipher Key. Para este estándar Nk = 4, 6, u 8 (según la longitud de la llave).
* Nr: Numero de rondas, que es una función de Nk y Nb (el cual es fijo). Para este estándar Nr = 10, 12, o 14 (según la longitud de la llave).
* Rcon[]: El array de palabras contante de la ronda.
* : Multiplicación de dos polinomios (cada uno con grando < 4) modulo x^4 + 1.
* •: Multiplicación en un campo finito.

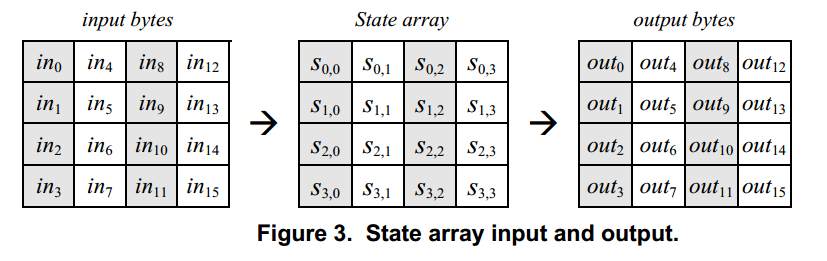
### Notación y Convenciones:

#### Inputs y Outputs

* Las inputs y outputs para el algoritmo AES consisten ambas en secuencias de 128 bits. A veces llamadas “bloques”.
* La Cipher Key es una secuencia de 128, 192 o 256 bits. Otras longitudes no son permitidas.

#### El State

Internamente las operaciones del algoritmo AES son realizadas en un array de bytes bidimensional llamado State. Este consiste de 4 filas de bytes, cada una conteniendo Nb bytes, donde Nb es el largo del bloque dividido por 32.



### Preliminares Matemáticas

* Todos los bytes en el algoritmo AES son interpretados como elementos de un campo finito.

#### Adición

* La adición es realizada mediante la operación XOR módulo 2.

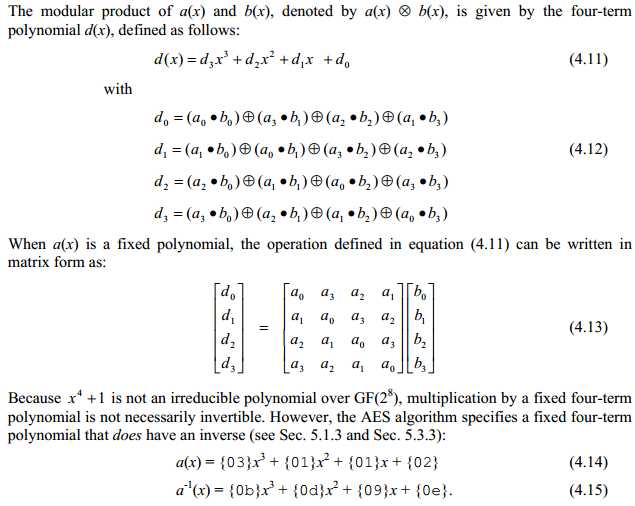
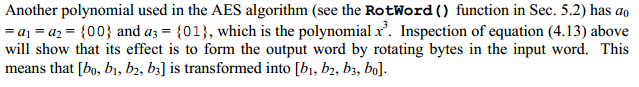
#### Multiplicación

* La Multiplicación en el GF(2^8), denotada como “•” corresponde a la multiplicación de polinomios módulo de un polinomio irreducible de grado 8.
* Para AES, dicho polinomio es:  
   
* La reducción modular por m(x) asegura que el resultado será un polinomio binario de grado menor a 8, y por lo tanto puede ser representado por un byte.

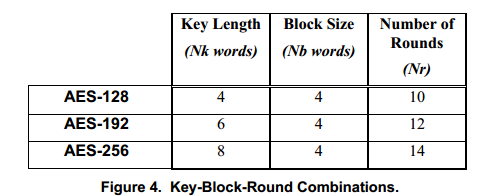
#### Multiplicación por x

* Multiplicar un polinomio binario por el polinomio x resulta en:  
    
  El resultado final es obtenido tras reducción el resultado de arriba módulo m(x).
  + Si b7 = 0, entontes el resultado ya está reducido.
  + Si b7 = 1, entonces hay que restar (o sea hacer XOR) el polinomio m(x).
* La multiplicación por x (en hex, {02}) puede ser implementada a nivel de byte como un shift a la izquierda, y un subsecuente XOR con {1b}.
* La operación anterior se denota “xtime()”.
* Multiplicación de potencias mayores de x puede ser implementada como la aplicación repetida de xtime().

### Polinomios con Coeficientes en GF(2^8)

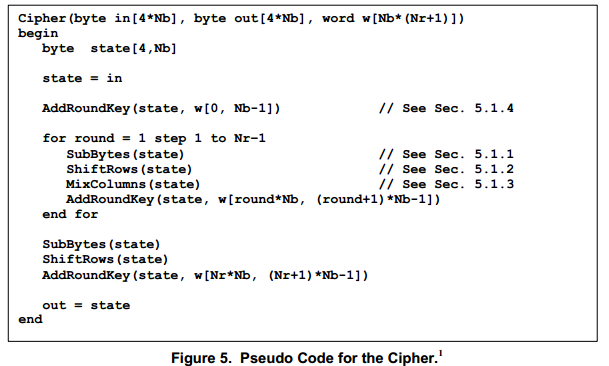
* Los coeficientes en esta sección son en si elementos de un campo finito, bytes, en vez de bits también, la multiplicación de polinomios de 4 términos usa un polinomio de reducción diferente.
* Suma: Corresponde a un XOR entre los correspondientes bytes de cada palabra:  
  
* Multiplicación: Se realiza en 2 pasos:
  + Producto:  
      
      
      
      
    El resultado c(x) no es una palabra de 4 bytes. Entonces el segundo paso…
  + Reducción c(x) módulo un polinomio de grado 4. Para AES ese polinomio es x^4 + 1.
* 
* 

### Especificación Algorítmica



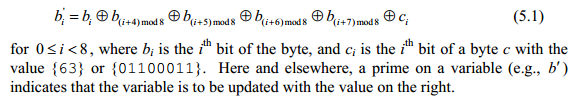
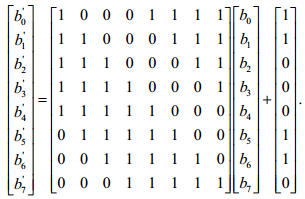
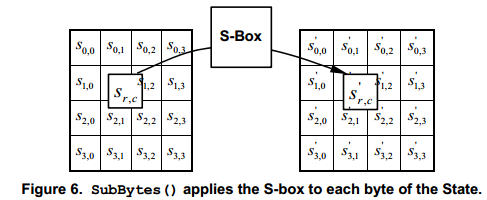
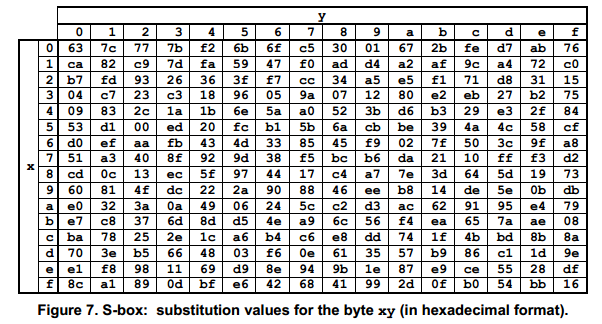
#### Cifrado (Cipher)

1. Al inicio del cifrado el input es copiado al array State, como se muestra en la figura 3, líneas arriba.
2. Adicción de Round Key inicial.
3. Luego el State es transformado implementado la función ronde 10, 12 o 14 veces según corresponda.
4. La ronda final difieres de las anteriores.
5. El State final luego es copiado al output, como dice la figura 3.

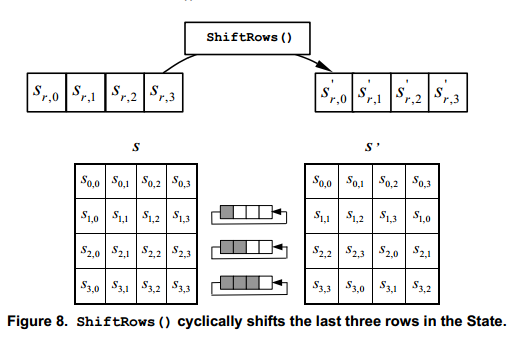
* La función round es parametrizada usando un key Schedule que consiste en un array unidimensional de palabras de 4 bytes, derivada de usar la rutina de Key Expansion.
* Pseudocódigo:  
  

A continuación se muestran consideraciones para las 4 operaciones que componen la función round:

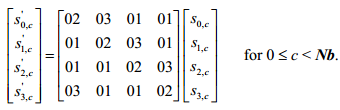
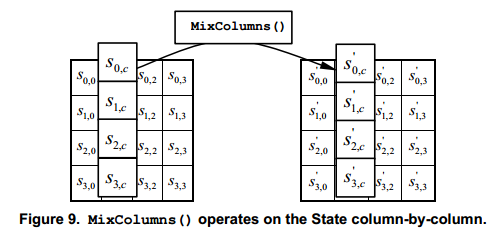
SubBytes() Transformation

* Es una sustitución de byte no lineal que opera independientemente en cada byte del State, usando una tabla de sustitución “S-box”.
* Esta S-box, que es invertible, es construida mediante la composición de 2 transformaciones:
  + Tomar el inverso multiplicativo del campo finito GF(2^8) (usando el algoritmo extendido de euclides); el elemento {00} es mapeado a si mismo.
  + Aplicar la función afin (sobre GF(2)):  
      
    En forma de matriz, la transformación afín elemento de la S-box puede expresarse como:  
    
* El efecto de la transformación SubBytes() en el State sería:  
  
* La S-box es:  
  

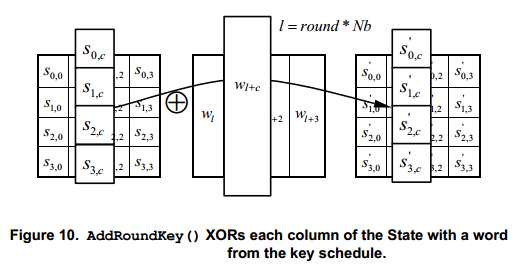
ShiftRows() Transformation

* Las 3 últimas filas del State son shifteadas.
* El algoritmo opera así:  
  

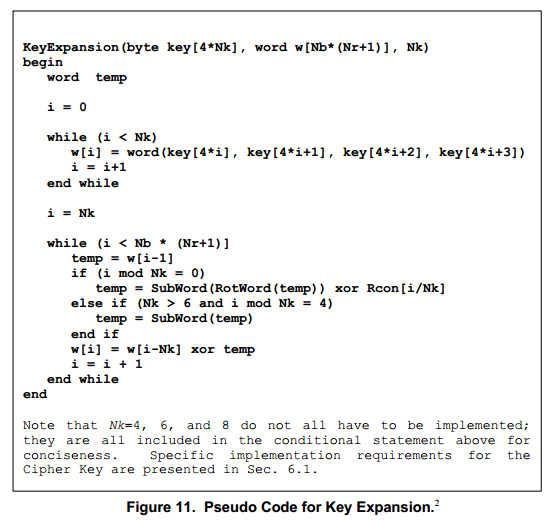
MixColumns() Transformation

* Las columnas del State son consideradas como polinomios sobre GF(2^8) y multiplicadas modulo x^4 + 1 por un polinomio fijo a(x):  
    
  Esto puede escribirse como:  
  
* MixColumns() Transformation puede ilustrarse como:  
  

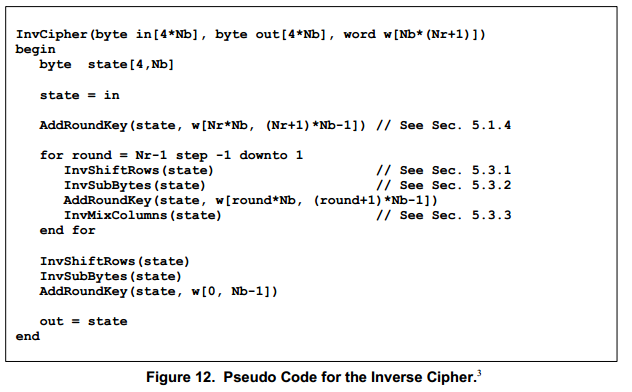
AddRoundKey() Transformation

* En esta transformación, una Round Key es sumada al State con un simple XOR.
* Cada Round Key consiste en Nb words del key Schedule.
* Esas Nb words con sumadas cada una en las columnas del State, tal que:  
    
  donde [wi] son las words key Schedule, y “round” es una valor en el rango de 0<=round<=Nr.
* En el Cipher, la adición Round Key inicial ocurre cuando round=0, previo a la primer aplicación de la función round.
* La aplicación de AddRoundKey() Transformation a las Nr rondas del Cifrado ocurre cuando 1<=round<=Nr.
* La acción de esta transformación se puede ilustrar (donde l = round\*Nb):  
  

#### Expansión de Llave (Key Expansion)

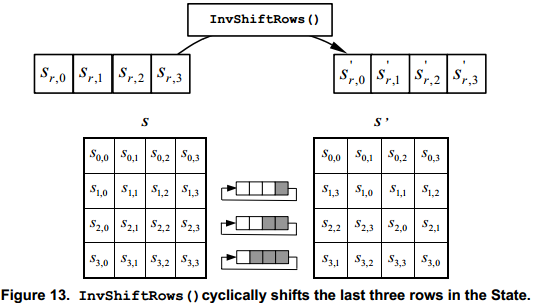
* AES toma la Cipher Key, K, y realiza una rutina Key Expansion para generar un key Schedule.
* La Key Expansion genera un total de Nb\*(Nr+1) palabras (words): el algoritmo requiere un set inicial de Nb words, y cada una de las Nr rondas requiere Nb words de key data.
* El key Schedule resultante consiste en un arrat lineal de palabras de 4 bytes, denotado como [wi], estando “i” en el rango 0<=i<=Nb\*(Nr+1).
* Pseudocódigo  
  
* Las funciones llamadas en el pseudocódigo son:
  + SubWord(): Es una función que toma un input Word de 4 bytes y le aplica la S-box a cada uno de los 4 bytes para producir una output Word.
  + RotWord(): Toma una palabra [a0,a1,a2,a3] como input, le realiza una permutación cíclica, y devuelve la palabra [a1,a2,a3,a0].
  + Rcon[i]: Es un Word array constante de ronda, que contiene valores dados por:  
     con x^i-1 siendo potencias de x (x denotada como {02} en hex) en el campo GF(2^8). (notar que i arranca en 1, no en 0).
* Es importante notar que la rutina Key Expansion para Cipher Keys de 256 bits (Nk = 8) es ligeramente diferente que para las de 128 y 192 bits. Si Nk = 8 y i-4 es un múltiplo de Nk, entonces SubWord() es aplicado a w[i-1] previo al XOR

#### Cifrado Inverso (Inverse Cipher) “Descifrado”

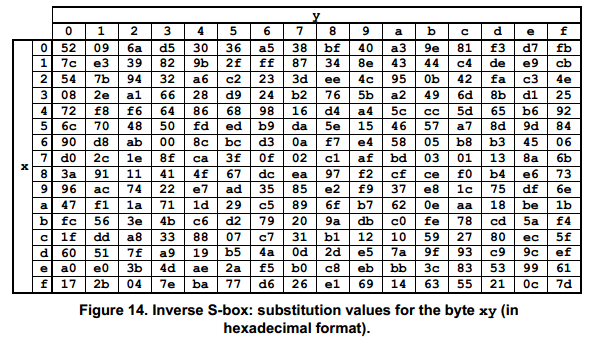
* Las Cipher Transformation pueden ser invertidas y luego ser implementadas en orden inverso para producir un Cifrado Inverso directo para AES (descifrar)
* Las funciones individuales usadas en el descifrado serian: InvShftRows(), InvSubBytes(), InvMixColumns(), y AddRoundKey().
* Pseudocódigo:  
  

A continuación se muestras como operan las transformaciones inversas:

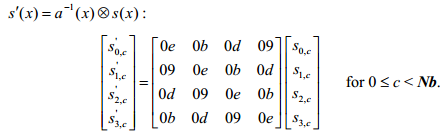
InvShiftRows() Transformation



InvSubBytes() Transformation

* Se aplica la inversa de la S-box  
  

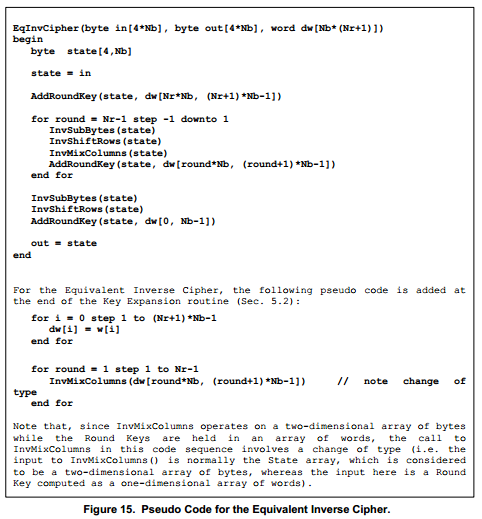
InvMixColumns() Transformation

* Opera sobre la State, columna por columna, tratando cada columna como un polinomio de 4 terminos.
* Las columnas son consideradas polinomios sobre el campo GF(2^8) y multiplicadas módulo x^4 + 1 por un polinomio fijo a^-1(x):  
    
  Se puede escribir como:  
  

Inv de AddRoundKey() Transformation

AddRoundKey() es su propia inversa, dado que sólo involucra la implementación de la operación XOR.

#### Cifrado Inverso Equivalente (Equivalent Inverse Cipher) “Descifrado Equivalente”

* En el Inverse Cipher directo, la secuencia de transformación difiere del Cipher, mientras que la forma de las key schedules para la encriptación y desencriptación permanecen iguales. Sin embargo, varias propiedades de AES, permiten la aparición de un Equivalente Inverse Cipher, que tiene la MISMA secuencia de transformación que el Cipher (con las transformaciones reemplazadas por sus inversas). Esto es acompañado con un cambio en el key Schedule.
* Hay 2 propiedades que permiten que exista este Equivalent Inverse Cipher:
  + Las transformaciones SubBytes() y ShiftRows() conmutan, esto es, una transformación SubBytes() inmediatamente seguida por una ShiftRows() es EQUIVALENTE a una transformación ShiftRows() inmediatamente seguida por una SubBytes(). Esto se aplica a sus inversas: InvSubBytes() e InvShiftRows().
  + Las operación de mezcaldo de columnas – MixColumns() e InvMixColumns – son lineales con respecto a la columna de entrada (input), lo que significa que:  
    InvMixColumns(state XOR RoundKey) = InvMixColumns(state) XOR InvMixColumns(RoundKey)
* Estas propiedades permiten que el orden de InvSubBytes() e InvShiftRows() pueda reservarse. El orden de AddRoundKey() e InvMixColumns() también puede reservarse, siempre y cuando las columnas (words) de la decryption key Schedule sea modificado usando la transformación InvMixColumns().
* Pseudocódigo:  
    
  La primera y última Nb words de la decryption key Schedule NO debe ser modificado de esta manera.
* Dados estos cambios, el resultante Equivalente Inverse Cipher ofrece una estructura MAS EFICIENTE que el Inverse Cipher.

### Problemas de Implementacion

#### Requerimiento del largo de Key

* AES debe soportar al menos uno de los siguientes largos de key: 128, 192, o 256 bits (NK = 4, 6 u 8 respectivamente).
* La implementación puede opcionalmente soportar 2 o 3 key lenghts, lo que promovería la interoperabilidad de la implementación de los algoritmos.

#### Restricciones de Key

* No hay restricciones en la elección de la key.

#### Parametrización de Key Lenght, Block Size y Round Number

* El estándar define los valores Nk, Nb y Nr, sin embago los diseñadores pueden elegir diseñar la implementación AES teniendo en mente una futura FLEXIBILIDAD.

## SOBRE RECOMENDACIONES SOBRE MODOS DE OPERACIÓN DE CIFRADO POR BLOQUES.- Métodos y Técnicas (NIST SP 800-38A.- Ed. 2001)

### Autoridad

* Esta recomendación no es ni un estándar ni una guía, ni mandatorio.

### Introducción

* Esta recomendación especifica 5 modos de operación confidenciales para algoritmos de cifrado por bloque con clave simétrica (como AES):
  + Electronic Codebook (ECB)
  + Cipher Block Chaining (CBC)
  + Cipher Feedback (CFB)
  + Output Feedback (OFB)
  + Counter (CTR)
* Ellos proveen Confidencialidad de datos

### Definiciones, Abreviaciones y Símbolos

#### Definiciones y Abreviaciones

* Confidenciality Mode: Es un modo que es usado para cifrar texto plano (plaintext) y descifrar texto cifrado (ciphertext). En este documento los Confidenciality Modes son ECB, CBC, CFB, OFB, CTR.
* Initialization Vector (IV): Es un bloque de datos (data block) que algunos modos de operación requieren como input inicial adicional.
* Nonce: Es un valor que sólo se usa una vez.

#### Símbolos

* X|Y: Es la concatenación de 2 bit strings X e Y.
* XY: XOR entre X e Y (bit string del mismo largo)
* CIPH\_K(X): Función de cifrado (cipher) del algoritmo de cifrado por bloques bajo la llave K aplicado al bloque de datos X.
* CIPH\_K^-1(X): Función de cifrado inversa (inverse cipher) del algoritmo de cifrado por bloques bajo la llave K aplicado al data block X.
* LSB\_m(X): Bit string conformado por los “m” bits menos significativos de bit string X.
* MSB\_m(X): Bit string conformado por los “m” bits más significativos de bit string X.
* [x]\_m: Representación binaria del entero no negativo x, en “m” bits, donde x<2^m

### Premilinares

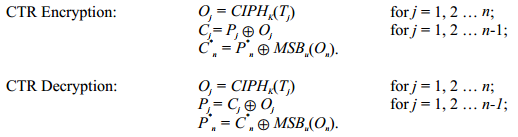
#### Representación del Plaintext y el Ciphertext

* Para OFB y CTR, el plaintext no necesita ser múltiplo del block size.
* El plaintext está conformado por bit strings. El último de ellos puede no ser un bloque completo.
* El ciphertext es una secuencia de data blocks o segmentos. El último de ellos puede ser un bloque parcial.

#### Initialization Vectors

* CBC, CFB, y OFB necesitan un IV.
* ECB y CTR no necesitan IV.

### El Modo Counter (CTR)

* Aplica un cifrado a un set de input blocks, llamados “COUNTERS”, para producir una secuencia de output blocks que luegos son XOReados con el plaintext para producir el ciphertext y viceversa.
* La secuencia de contadores (Counters) deben cumplir la propiedad de que cada bloque es diferente a cualquier otro.  
  Esto no está limitado a un solo mensaje, sino que a través de todos los mensajes que son encriptados bajo la misma llave (key), todos los counters deben ser distintos.
* En esta recomendación los counters para un mensaje dado son denotados T1, T2,…, Tn.
* Los métodos para generar contadores se discuten más adelante.
* Dada una secuencia de conunters, el modo CTR se define como:  
  

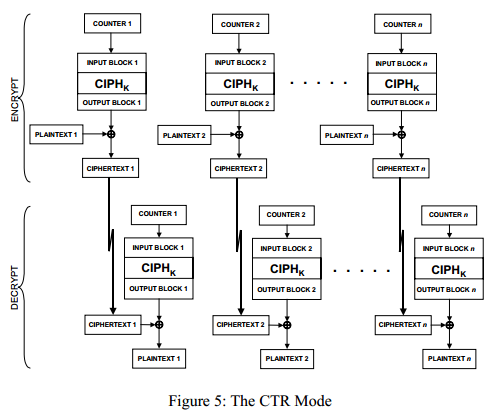
#### CTR Encryption

* La función de cifrado en invocada en cada bloque counter, y los resultantes ouputs blocks son XOReados con el correspondiente bloque de plaintext para producir ciphertext blocks.
* Para el últmo bloque, el cual puede ser un bloque parcial de “u” bits, los u bits más significativos de ultimo output block es usado para hacer el XOR. Los restantes b-u bits de último output block son descartados.

#### CTR Decryption

* IDEM CTR Encryption.
* Ocurre lo mismo que en CTR Encryption con el último bloque.

#### CTR Encryption & Decryption

* Las funciones de cifrado pueden ser realizadas en paralelo; similarmente, el bloque de texto plano que corresponde a algún ciphertext block en particular, puede ser recuperado independientemente de los otros plantext blocks, si el correspondiente counter block puede ser determinado.
* Las funciones de cifrado pueden ser aplicadas a los contadores, previo a que los datos del plaintext o el ciphertext estén disponibles.
* Ilustración del Modo CTR  
  

### Rellenado (Padding)

* No es necesario para el modo CTR.

### Generación de los Counter Blocks

* Si el counter block es usado repetidamente, entonces la confidencialidad de todos los bloques de plaintext correspondiente a ese counter block se ve comprometido.
* Existen 2 aspectos para satisfacer el requerimiento de unicidad:
  + Una función incremente para generar counter blocks a partir de un counter block inicial, para asegurar que los counter blocks no se repitan dado un cierto mensaje.
  + El counter block inicial T1, debe ser elegido de tal manera que asegure que los counters son únicos a través de todos los mensajes que son encriptados bajo una misma key.

#### The Standart Incrementing Function

* Toma como entrada [x]\_m (es decir el entero no negativo “x” expresado en m bits) y retorna a la salida [x+1 módulo 2^m]\_m.
* Básicamente le suma 1 a los m bits menos significativos, módulo 2^m

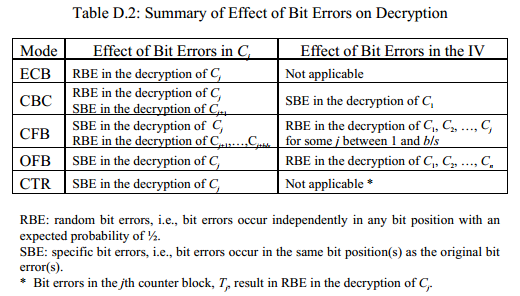
#### Elección del Counter Block Inicial

* Existen 2 enfoques para la elección del counter block inicial:
  + Primer enfoque:
    - Para una key dada, todos los mensajes del plaintext son encriptados secuencialmente.
    - De cada mensaje, el mismo set fijo de “m” bits del counter block es incrementado por la función incremental estándar.
    - El Counter Block Inicial puede ser CUALQUIER string de “b” bits.
    - El Counter Block Inicial para cualquier mensaje subsecuente puede ser obtenido mediante la aplicación de la función incremental estándar al set fijo de “m” bits al final del counter block del bloque anterior.
    - El número total de plaintext block no debe exceder 2^m.
  + Segundo Enfoque
    - Se le asigna a cada mensaje un string único de “b/2” bits, (redondeando hacia arriba si es necesario); en otras palabras un MESSAGE NONCE, y luego incorporar ese message nonce dentro de cada counter block para ese mensaje.
    - Los primeros b/2 bits para cada counter block será el message nonce
    - Al resto de los “m” bits se les aplica la función incremental estándar para proveer un índice a los counter blocks del mensaje.
    - El número de bloques, “n”, en cualquier mensaje debe cumplir n<2^m.
    - Un procedimiento debe establecerse para asegurar la unicidad de los message nonces.
* Esta recomendación permite otros métodos y acercamientos para lograr la unicidad pertinente.

### Generación de Initialization Vectors (IV)

* El Modo CTR no necesita IVs.

### Propiedades de los Errores

* Para los Modos ECB, OFB y CTR, un bit error en el ciphertext block no afecta el descifrado de los otros bloques.
* En el Modo CTR, si hay un bit error en un counter block, entonces puede ocurrir un bit error, independientemente, en cualquier posición de bit del descifrado del correspondiente ciphertext, con un error rate estimado del 50%.
* En los Modos OFB y CTR, el descifrado de cualquier ciphertext block es vulnerable a la introducción de un bit error específico en el ciphertext block si su integridad no esta protegida.
* Sumario del Efecto de los Bit Errors en el Descifrado:  
  

## SOBRE RECOMENDACIONES SOBRE MODOS DE OPERACIÓN DE CIFRADO POR BLOQUES: GALOIS/COUNTER MODE (GCM) and GMAC.- (NIST SP 800-38D.- Pub. 2007)

### Abstracto

* Esta recomendación especifica el modo Galois/Counter (GCM), un algoritmo para encriptación autenticada (authenticated encryption) con datos asociados (associated data), y su especialización, GMAC, para generar un código de autenticación de mensaje (MAC) sobre los datos que no son encriptados.
* GCM y GMAC son modos de operación para cifrados por bloques con clave simétrica.

### Autoridad

* No está sujeto a copyright.

### Introducción

* GCM es construido a partir de un cifrado por bloque con clave simétrica de 128 bits, como AES.
* GCM es un modo de operación del algoritmo AES.
* GCM provee confidencialidad de datos, usando una variante de CTR para encriptar.
* GCM provee autenticidad de los datos confidenciales (hasta 64 GB por invocación) usando una función universal Hash que es definida sobre una campo de Galois binario.
* GCM puede también provee autenticación para datos adicionales (de largo prácticamente ilimitado por invocación) que no son encriptados.
* Si el input es solo datos que no van a ser encriptados, entonces GCM se llama GMAC; que es simplemente un modo de autenticación sobre la input data.
* GCM provee una autenticación más fuerte que un checksum (no criptográfico) o un código de detección de errores. En particular, GCM puede detectar ambos:
  + Modificaciones accidentales de datos y
  + Modificaciones intencionales no autorizadas.
* Las 2 funciones de GCM se llaman:
  + Encriptación Autenticada (authenticated encryption)
  + Desencriptación Autorizada (authenticated decryption)

Cada una de estas funciones el relativamente eficiente y paralelizable, consecuentemente es posible realizar implementaciones con alto throughput, tanto en HW como en SW.

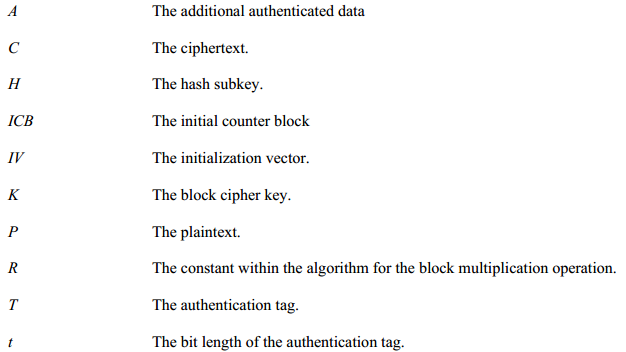
* GCM tiene otras características como:
  + Las funciones de GCM están “online” en el sentido de que los largos de los datos confidenciales, los datos adicionales y los datos no confidenciales pueden ser calculados a medida que éstos van llegando y siendo procesados.
  + Las funciones GCM requieren solamente la dirección directa (en un solo sentido) del block cipher (algoritmo) subyacente.
  + La autenticidad de los datos protegidos puede ser verificado independientemente desde la recuperación de los datos confidenciales de su forma encriptada.
  + Si el string de inicialización único es predecible y el largo de los datos confidenciales es conocido, entonces las invocaciones del block cipher (algoritmo) dentro del mecanismo de encriptación de GCM se pueden pre computar.
  + Si un parte del total de los datos adiciones no confidenciales es fijo, entonces los correspondientes mecanismos de autenticación dentro de GCM pueden pre computarse.
* Es importante ser cuidadoso con la unicidad del string de inicialización, pues puede comprometer la seguridad casi enteramente. Por lo tanto este modo de operación no debe utilizarse a menos que el requerimiento de unicidad este asegurado.

### Definiciones, Abreviaciones y Símbolos

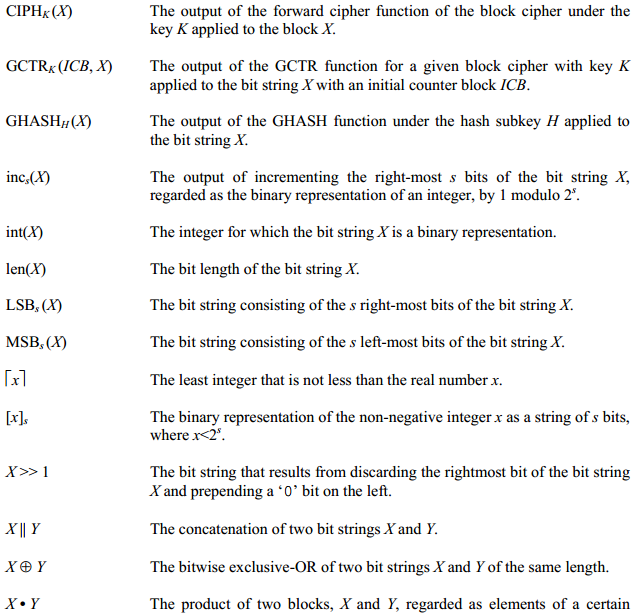
#### Definiciones y Abreviaciones

* AAD: Additional Authenticated Data.
* Additional Autheticated Data: Es la input data a la authenticated encription function que es autenticada pero no encriptada.
* Authenticated Decryption: Es la función de GCM en cual es ciphertext es desencriptado retornando el plaintext y la autenticidad del del ciphertext y del AAD es verificada.
* Authenticated Encryption: Es la función de GCM en cual el plaintext es encriptado retornando el ciphertext y un tag de autenticación es generado a partir del AAD y el ciphertext.
* Tag de Autenticación (Authentication Tag): Es un checksum criptográfico sobre los datos. Está designado a revelar errores accidentales y modificaciones intencionales sobre los datos.
* Direct Random String: En la construcción del IV basado en RGB (Random Bit Generator) el output string del RGB que es usado en el random field de un IV se llama direct random string.
* Fixed Field: En la construcción determinística de los IVs, el campo que identifica el device o el contexto para la instancia de la authenticated encryption function.
* Foward Cipher Function: Una permutación de bloques que está determinada por el elección de una key para un determinado block cipher (algoritmo)
* Free Field: En la construcción del IV basado en RGB (Random Bit Generator) el campo cuyo contenido no está restringido.
* Fresh: Para una key recién generada, es la propiedad de ser distinta a cualquier otra key usada previamente.
* ICB: Initial Counter Block.
* Initialization Vector (IV): Es un “nonce” (number used once) que está asociado con la invocación de authentication encryption sobre un plaintext y AAD particular.
* Inverser Cipher Function: Es la función que es la inversa al Foward Cipher Function para una key determinada.
* Invocation Field: En la construcción determinística de los IVs, el campo que identifica el set de inputs a la auth. encryption function en un device o contexto particular.
* IT: Information Technology.
* IV: Initialization Vector.
* Key. Es el parámetro de un block cipher (algoritmo) que determina la selección de la foward cipher function de la familia de permutaciones.
* Nonce: Numer used once dentro de un determinado contexto.
* Permutation: Es una función invertible.
* Random Field: En la construcción del IV basado en RGB (Random Bit Generator), puede ser un direct randob string o uno de sus sucesores (successors).
* RGB: Random Bit Generator.
* Successor: En la construcción del IV basado en RGB (Random Bit Generator), el resultado de una más aplicaciones de la apropiada función incremental a un direct random string.

#### Simbolos

* Variables  
  



* Operaciones y Funciones  
    
  

### Elementos de GCM

#### Block Cipher

* Las operaciones de GCM dependen de la elección del block cipher de clave simétrica subyacente y por tal puede considerarse un modo de operación (modo para acortar) del block cipher.
* La clave GCM es la block cipher key (key para acortar).
* Para cualquier key dada, el block cipher subyacente del modo consiste de 2 funciones que son inversas una a la otra. Sin embargo GCM no utiliza la inverse cipher function.
* La key debe ser generada uniformemente aleatoria o lo más cercano a uniformemente aleatoria.
* La key debe ser fresca (“fresh”), es decir, distinta a cualquier key previa.
* La key debe ser secreta y debe ser usada exclusivamente para GCM con el block cipher seleccionado.

#### Las 2 Funciones GCM

* Las 2 funciones que componen GCM se llaman: Authenticated Encryption y Authenticated Decryption.
* La función authenticated encryption encripta los datos confidenciales y computa el authentication tag en ambos, los datos confidenciales y cualquien dato adicional no confidencail.
* La función authenticated decryption desencripta los datos confidenciales, contingente con la verificación del tag.
* Si se implemente GCM de tal manera que el input solo sea datos no confidenciales, entonces GCM se llama GMAC.
* Para GMAC, las funciones authenticated encryption y authenticated decryption se transforman en funciones que generan y verifican el authentication tag sobre los datos no confidenciales.

Authenticated Encryption Function

* Input Data:
  + Está conformado por:
    1. Plaintext, denotado “P”.
    2. Additioanl Authenticated Data (AAD), denotado “A”.
    3. Initialization Vector (IV), denotado “IV”.
  + El plaintext y el AAD son las 2 cagtegorias de datos que GCM proteje.
  + GCM protege la autenticidad del plaintext y la AAD.
  + GCM también `protege la confidencialidad del plaintext, mientras que la AAD se deja limpio. Por ejemplo, dentro de un protocolo de red, la AAD podría incluir la dirección, puertos, número de secuencia, y otro tipo de información que indiquen como debe tratarse el plaintext.
  + El IV es en esencia un nonce, es decir un valor que es único dentro de un contexto especifico, que determina la invocación de la authenticated encryption function sobre la input data para ser protegida.
  + Las IVs (y las keys) deben ser únicas. Se tratará más adelante este aspecto.
  + Length(P) <= 2^39 – 256;
  + Length(A) <= 2^64 – 1;
  + 1 <= length(IV) <= 2^64 – 1.
  + A pesar que GCM es definido sobre bit strings, tanto el plaintext, como el AAD y las IVs deben ser todos múltiplos de 8, de tal manera que estos valores sean byte strings.
  + Un único set de largo de bit soportado para cada uno de los 3 inputs debe ser establecido para la totalidad de la implementación, independientemente de la key.
* Output Data:
  + Está conformado por:
    1. Un ciphertext, denotado “C”, cuyo largo en bits es el mismo que el plaintext.
    2. Un authentication tag, o tag, denotado “T”.
  + El largo en bits del tag, denotado “t”, es un parámetro de seguridad. En general puede ser: 128, 120, 112, 104, o 96. Para algunas aplicaciones t puede ser 64 o 32.
  + La implementación elegida no debe soportar valores de t que sean diferentes a las 7 opciones nombras anteriormente.
  + Un único valor fijo de t de todas las opciones soportadas debe ser asociado a cada key.

Authenticated Decryption Function

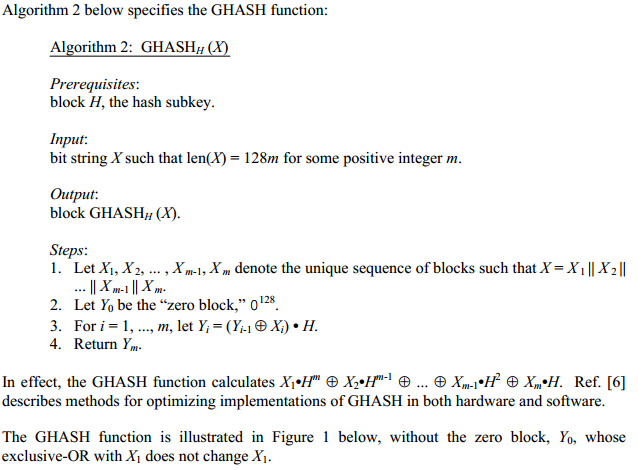
* Input Data:
  1. IV
  2. A
  3. C
  4. T
* Output Data: El output de uno de los siguientes, según corresponda:
  + El plaintext “P” que corresponde al ciphertext C, o bien
  + Un código especial de error, denotado “FAIL”.
* El output P indica que T es el correcto tag para IV, A y C; de lo contrario, el output es FAIL.
* Los valores para len(C), len(A), y len(IV) para la authentication decryption function deben ser los mismos que los valores de len(P), len(A), y len(IV) en la authentication encryption function.

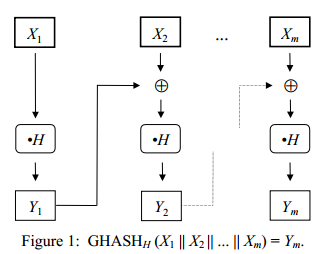
#### Primitivas para la Confidencialidad y Autenticación

* El mecanismo para la Confidencialidad del plaintext dentro de GCM es una variación del CTR, con una función incremental particular, denotada “inc\_32” para generar la necesaria secuencia de counter blocks.  
  El primer counter block para la encriptación del plaintext es generado mediante el incremento de un bloque que es generado a partir del IV.
* El mecanismo para la Autenticación dentro de GCM está basado en una Función Hash, llamada GHASH, que implementa la multiplicación por un parámetro fijo llamado “Hash SubKey”, dentro de un campo binado de Galois.  
  La Hash SubKey, denotada “H”, es generada mediante la aplicación del block cipher (algoritmo) al bloque “Cero”. La instancia resultante de esta función hash, denotada GHAS\_H, es usada para comprimir un AAD codificado y el ciphertext en un solo bloque, el cual es luego encriptado para producir el authentication tag.  
  GHASH es una función hash con clave, pero no es en sí una función hash criptográfica.
* Los valores intermedios en la ejecución de las funciones en GCM deben ser secretos.

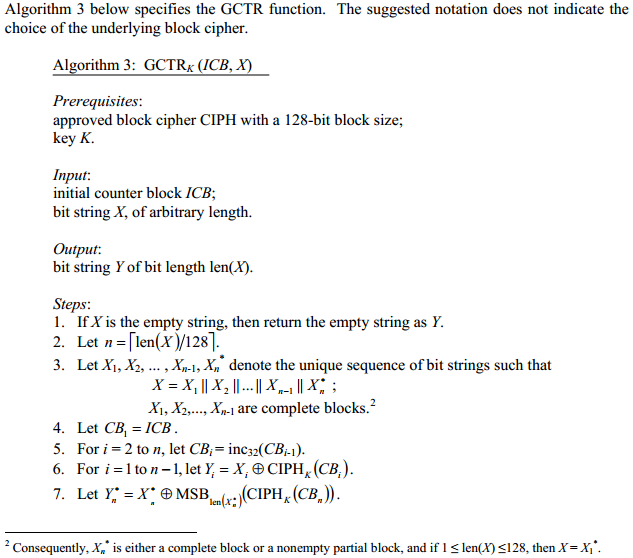
### Componentes Matemáticos de GCM

#### GHASH Function

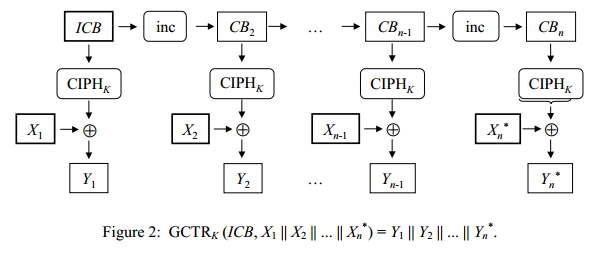




#### GCTR Function

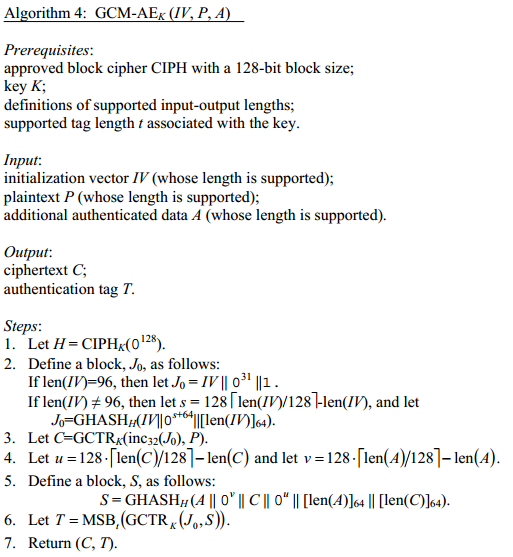






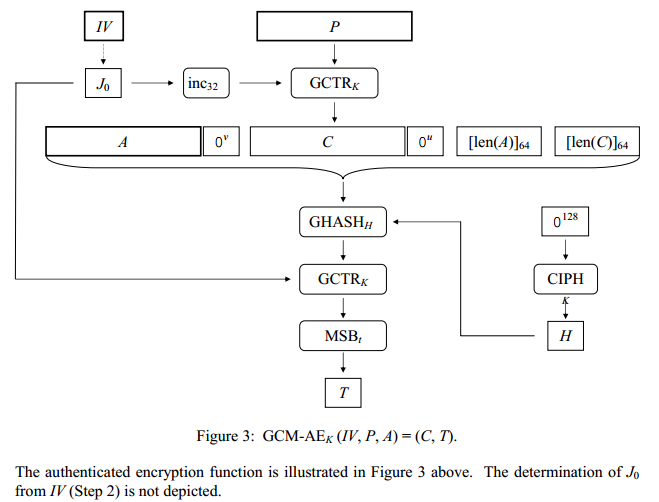
### Especificación GCM

#### Algoritmo para la Authenticated Encryption Function

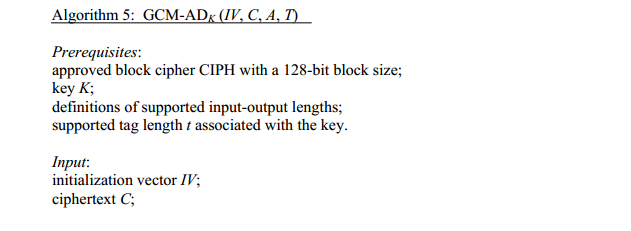


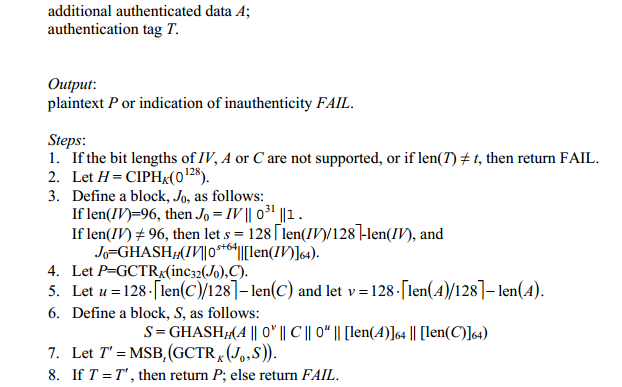
Pasos

1. La hash subkey para la función GHASH es generada mediante la aplicación del block cipher al “zero” block.
2. El pre-counter block (J0) es generado a partir del IV.  
   En particular cuando len(IV)=96 bits, entonces el string de relleno 0^31||1 es concatenado al IV para formar el pre-counter block.  
   De otro modo, IV es rellenado con el mínimo número de 0 bits (puede ser ninguno) para que el largo del string resultante sea múltiplo de 128 bits (el block size). El string de turno es concatenado con 64 0 bits adicionales, seguido de la representación en 64 bits del largo del IV, y luego la función GHASH es aplicado al string resutlante para formar el pre-counter block.
3. La función incremental de 32-bits es aplicada al pre-counter block para producir el counter block inicial para una invocación de la función GCTR sobre el plaintext.  
   El output de dicha invocación es el ciphertext.
4. En este paso y en el…
5. La ADD y el ciphertext son ambos concatenado con el mínimo numero de 0 bits (puede ser ninguno) de tal manera que el largo en bits del resultante string sea múltiplo del block size.  
   La concatenación de estos strings es adjuntada con la representación en 64 bits de los largos de la AAD y el ciphertext, y luego la función GHASH es aplicado al resultado para producir un único output block.
6. Este output block es encriptado usando la función GCTR con el pre-counter block generado en el Paso 2, y el resultado es truncado al largo del tag especificado para formar el authentication tag.
7. El ciphertext y el tag son retornado como la salida en el paso 7.



#### Algoritmo para la Authenticated Decryption Function

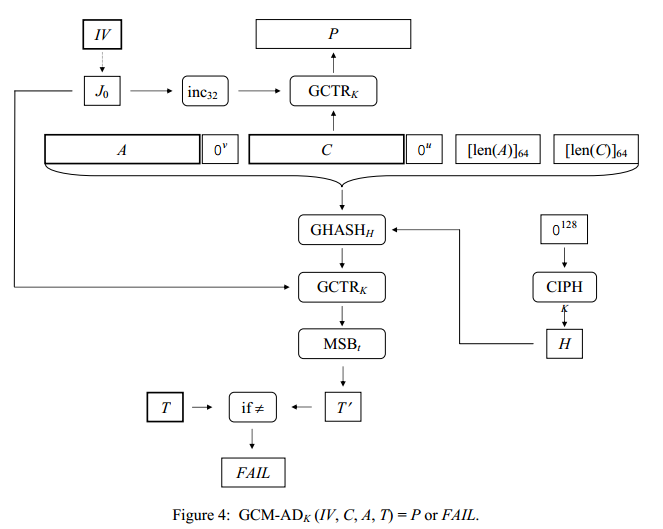




Pasos

1. Se verifica si el largo del IV, el ciphertext, la AAD t el authentication tag están soportadas para esa implementación.
2. La hash subkey para la función GHASH es generada mediante la aplicación del block cipher al “zero” block.
3. El pre-counter block (J0) es formado como en la authentication encription function en la sección anterior.
4. La función incremental de 32-bits es aplicada al pre-counter block para producir el initial counter block para una invocación de la función GCTR sobre el ciphertext.  
   El output de dicha invocación es el plaintext que corresponde al ciphertext para el IV dado.
5. En este paso y el…
6. La AAD y el ciphertext son ambos concatenado con el mínimo numero de 0 bits (puede ser ninguno) de tal manera que el largo en bits de los string resultantes sean múltiplos del block size.  
   La concatenación de dichos strings es adjuntada con la representación en 64 bits del largo de la AAD y el ciphertext, y luego la función GHASH es aplicada a ese resultado para producir un único output block.
7. Dicho output block es encriptado usando la función GCTR con el pre-counter block generado en el Paso 3, y el resultado es truncado al largo del tag especificado para formar el authentication tag.
8. El resultado del Paso 7 es comparado con el authentication tag que fue recibido como input:
   1. Si son idénticos, entonces el plaintext es retornado, sino
   2. Se retorna FAIL.

* Un set de pasos equivalente que produzca el output correcto es permitido. En particular, la verificación del tag puede preceder la computación del plaintext.



### Requerimiento de Unicidad en las IVs y Keys

* Las IVs en GCM deben cumplir con el siguiente requerimiendo de unicidad:  
  *“La probabilidad de que la authenticated encryption function sea invocada alguna vez con la misma IV y la misma key en 2 o más distintos sets de input data no debe ser mayor que 2^-32”*
* Que se cumpla dicho requerimiento es crucial para la seguridad de GCM. En la práctica dicho requerimiento es tan importante como el secreto de la clave para evitar ataques de falsificación (forgery attaks)
* Se presentan más adelante 2 formas de construir las IVs de modo tal que satisfagan el requerimientos antes mencionado.
* Las restricciones sobre la cantidad de invocaciones de la authenticated encryption function también se detallan más adelante.

#### Establecimiento de Key

* *“Cada GCM key que establecida por los usuarios debe, con alta probabilidad, ser fresca (“fresh”)”.*
* En la práctica los requerimientos del block cipher (algoritmo) deben asegurar que una key es fresh cuando es generada, si el mecanismo de generación es resistente a manipulaciones.
* Si el mecanismo de generación de calve es determinístico, entonces el manejo del mecanismo debe proveer una fuerte resistencia contra cualquier entidad ajena que pueda inducir la repetición de un set de inputs previo al mecanismo, porque de otra maneja eso causaría la repetición de un output previo.  
  Por ejemplo, las GCM keys pueden ser establecidas usando las funciones de derivación de claves de los siguientes protocolos: Transpor Layer Security, Internet key Exchange v1 y v2, y Secure Shell.
* Similarmente si una nueva clave debe ser transportada, el método de transporte/distribución debe proveer una fuerte resistencia contra “replay”, de tal manera de que ninguna fuerza externa pueda inducir la sustitución de una key previa en vez de la nueva.
* Las GCM keys deben ser establecidas dentro de una estructura que asegure su frescura así como también su confidencialidad y autenticidad. Los detalles de dicha estructura esta fuera del alcance de esta recomendación.

#### Construcción de IVs

* En esta recomendación se proveen 2 marcos para la construcción de IVs:
  + La primera se basa en elementos determisticos para lograr su requerimiento de unicidad.
  + La segunda se basa en un output string de una RGB (Random Bit Generator) lo suficientemente largo para que de la suficiente fuerza de seguridad.
* Para cualquier largo de IV soportado que sea estricamente menor de 96 bits, la contruccion determinística debe ser usada a lo largo de todas las instancias de la authenticated encription function con la key dada.
* Para cualquier largo de IV que sea de 96 bits o mayor, una y sólo una de las construcciones debe ser usada a lo largo de todas las instancias de la authenticated encryption function con la key dada.

Deterministic Construction

* El IV es la concatenación de 2 campos llamados:
  + “Fixed Field” e
  + “Invocation Field”.
* El Fixed Field debe identificar el device, o más generalmente, el contexto para la instancia de la authenticated encryption function.
* El Invocation Field debe identificar el set de inputs para la authenticated encryption function en ese device particular.
* Para cada key dada, no puede haber 2 devices distintos que compartan el mismo fixed field, y no puede haber 2 set de inputs distintos para cualquier device particualr que puedan compartir el mismo invocation field. Si se cumplen estos 2 requerimientos, entonces se cumple con el requerimiento de unicidad sobre la IV.
* Similarmente, el fixe field completo debe consistir de bit arbitrarios cuadno hay sólo un contexto que identificar, como cuando una fresh key es limitada a una única sesión de un protocolo de comunicación.   
  En este caso, si diferentes participantes de la sesión comparten un fixed field común. Entonces el protocolo debe asegurar que los invocation fields sean distintos para cada data input.
* El invocation field es típicamente:
  + O un contador entero, o
  + Un lineal feedback shift register que está dado por un polinomio primitivo para asegurar el máximo largo de ciclo.

En cualquier caso, el invocation field incrementa sobre cada invocación de la authenticated encryption function.

* Los largos y posiciones de los fixed field e invocation field debe ser fijado para cada largo de IV soportado por la vida de la key.
* Con el objetivo de promover la interoperabilidad para el largo por default de IV de 96 bits, esta recomendación sugiere, pero no requiere, que los 32 bits más significativos del IV sean el fixed field, y que el resto (los 64 bit menos significativos) sean el invocation field.

RGB-based Construction

* En esta contruccion, el IV es la concatenación de 2 campos llamados:
  + “Random Field” y el
  + “Free Field”.
* Para cada largo de IV soportado por la implementación y usado en la RGB-bases construction, los largos de estos 2 campos debe ser fijada por la vida de la key. Es más, el largo del random field debe ser al menos de 96 bits; el free field puede estar vacío.
* Si “i” es un largo de IV soportado que esta soportado por la RGB-bases construction, entonces que “r(i)” denote el largo en bit del random field.  
  El random field puede consistir:
  + O bien de un output string de r(i) bits de un RGB aprobado con suficiente fuerza de seguridad, o bien
  + Puede ser el resultado de aplicar una función incremental a los r(i) bits del random field de la IV precedente, para una key dada. A estos r(i) bits del output string del RGB se le llama “Direct Random Field”, y a los random fields que resulten de aplicar la función incremental a los r(i) bit se les llama sus “Successors”.
* No hay requerimientos sobre los bits del free field.  
  Por ejemplo, ellos pueden identificar el device, similar al fixed field de la deterministic construction, excepto cuando en la RGB-bases construction estos identificadores no son requeridos que sean distintos para cada device.  
  Para cualquier largo de IV que este asociado al RGB-construction, el free field es recomendable que quede vacío, para que el random field sea el total del IV.
* Las instanciaciones de los RBGs en 2 distintos devices deben ser independientes, para que la distribución de los direct random strings a lo largo de todas las instanciaciones RGB sean uniformes.  
  Por ejemplo, si la inicialización de las instanciaciones RGB dependen sólo de una semilla secrete, entonces cada instanciación debe ser inicializada con una semilla distinta.

#### *Restricciones sobre el Número de Invocaciones*

* Los siguiente requerimientos aplican a todas las implementaciones, ya sea que:
  + Usen la deterministic construction con IVs cuyos largos no sean 96, o bien
  + Usen la RGB-based construction, para IVs de cualquier largo.

En otras palabras, a menos que la implementacion SOLO use IVs de 96 bits que sean generadas usando la deterministic construction:  
*“El número total de invocaciones de la authenticated encryption function no debe exceder 2^32, incluyendo todos los largos de IV y todas las instancias de la authenticated encryption función con la key dada”*.

* Para la RGB-based construction de IVs, el requerimiento de arriba, en conjunto con el requerimiento de que r(i)>=96, es suficiente para asegurar el requerimiento de unicidad.
* Para la deterministic construction, los largos de los 2 campos implica 2 restricciones operacionales. Estas restricciones aplican para cualquier largo de IV soportado, incluyendo 96 bits:
  + El largo en bits del invocation field, limita el número de invocaciones de la authenticated encryption function.  
    Si “s” denota el número de bits el invocation field, entonces la authentication encryption función no se puede invocar sobre más de 2^s distintos input sets sin violar el requerimiento de unicidad.
  + Similarmente, un fixed field de “s” bits implica un límite en el número de distintos devices/contextos que pueden implementar la authenticated encryption function para la key y largo de IV dados.

### Consideraciones Prácticas para Validar Implementaciones

#### Consideraciones de Diseño

* Con el objetivo de inhibir a un tercero no autorizado de controlar o influencia la generación de IVs, GCM debe ser implementado sólo dentro de una módulo criptográfico que cumpla los requerimientos de FIPS Pub. 140-2. En particual, el límite criptográfico del módulo debe contener un “generation unit” que produzca IVs acordes a una de las construcciones nombradas anteriormente.
* El FIPS 140-2 describe cómo un módulo cumple con los requerimientos de unicidad sobre los IVs. Como mínimo la documentación debe tener consideraciones en esta sección, y claramente documentar las responsabilidades del profesional IT que configura, despliega y mantiene las implementaciones GCM a lo largo de un sistema más grande.
* Las siguientes son 3 consideraciones importantes de diseño para los módulos GCM:
  + La frescura de las keys debe ser asegurada.
  + La IV debe ser un parámetro de seguridad crítica, hasta que la authnticated encryption function sea invocado con la IV. Previo a dicha invocación, la IV debe ser proveída de la misma protección que otros parámetros críticos de seguridad del módulo validado según los requerimientos del FIPS Pub. 140-2.
  + La pérdida de energía al módulo no debe causar la repetición de IVs. Si la unidad de generación no puede recuperarse de una pérdida de energía, entonces la authenticated encryption funtion entraría en un estado de falla hasta que una key fresca pueda ser establecida.
* La construcción de un IV, utilizando los métodos antes explicados, afecta las opciones de recuperación de una pérdida de energía:
  + Para la deterministic construction, todos los elementos determinísticos que son necesarios para construir la IV deben estar disponibles para cuando la energía se restaure. Por ejemplo, estos elementos pueden ser restaurado de una memoria no volátil.  
    Cuando la energía se restaure, ni la IV precedente ni cualquier otra IV previa debe ser inmediatamente repetida para la key en uso. Una forma de evitar dicha repetición puede ser asegurarse que el valor del invocation field que es periódicamente guardado en la memoria no volátil sea siempre uno o más valores por delante del valor operacional en la secuencia.
  + Una potencial ventaja del RGB-based construction es que el RGB puede ser diseñado para recuperse de una pérdida de energía en una manera directa, es decir, sin requerir de la acción de ningún profesional de IT que mantenga el sistema. Por ejemplo, el RGB puede incorporar una fuente no determinística de bits que puede automáticamente estar disponible para el RGB cuando la energía sea restaurada.  
    Alternativamente, el estado entero de la RGB puede ser guardada periódicamente en una memoria no volátil. En este caso, similar a la deterministic construction, cuando la energía se restaure, el diseño debe asegurar que el RGB no entregue strings para el uso de nuevas IVs hasta que el estado del RGB esté lo suficientemente avanzado sobre el estado que generó el último IV para la clave dada. En otras palabras, el direct random string para la primer IV nueva debe ser, con alta probabilidad, diferente que el direct random string en cualquier IV que haya sido generada antes de la pérdida de energía.  
    Incluso si el proceso no es automático, el profesional de IT que mantiene el sistema puede simplemente reinicializar el RGB cuando la energía se restaure, por ejemplo, con una semilla fresca.

#### Consideraciones Operacionales

* Algunas de la consideraciones operacionales típicas para el requerimiento de unicidad son:
  + ¿Es la configuración de cualquier módulo GCM, o algún valor operacional, vulnerable para ser controlado o influenciado por un tercero no autorizado?
  + Para una key dada, ¿Cómo son las opciones de configuración impuestas a través de todos los módulos que son alguna vez implementado en la authenticated encryption function?
  + ¿Cómo es la frescura de la key asegurada?
  + Si una implementación no usa exclusivamente IVs de 96 bits que son generados mediante la deterministic construction, ¿cómo es el requerimiento de la cantidad total de invocaciones asegurado?
  + ¿Cómo la generación de IVs dentro de los módulos se recupera de una pérdida de energía sin violar los requerimientos de unicidad sobre los IVs?
* Las siguientes consideración son específicas para la deterministic construction:
  + ¿Cómo son los identificadores de devices instalados dentro del fixed field, de tal manera que la conformidad con los requerimientos sobre el fixed field, tanto para los módulos inicialmente desplegados como por cualquier otro módulo desplegado subsecuentemente?
  + Para cualquier key y largo de IV dada, ¿es el largo del fixed field suficiente para soportar el número de módulos que implementarán authenticated encryption?
  + Si la configuración por default para los largos de los fixed e invocation fields pueden ser alterado, entonces ¿Cómo es la elección impuesta a través de todos los módulos con cualquier key dada?
* Las siguiente consideración es específica para el RGB-bases construction:
  + ¿Cómo son las RGBs para la generación de IVs inicializadas, de modo tal que la conformidad con el requerimiento de independencia para los módulos inicialmente desplegados y para los subsecuentes módulos desplegados sea asegurada?

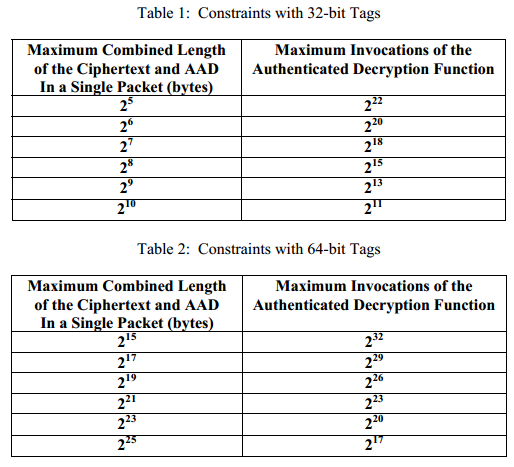
### Importancia sobre el Requerimiento de Unicidad sobre las IVs

* En esta sección se resumen porqué este requerimiento es crucial para la seguridad de GCM.
* Un ejemplo obvio para cualquier algoritmo de clave simétrica es que la key necesita ser mantenida en secreto. Para GCM, el requerimiento de unicidad para las IVs es casi tan importante, sin embargo no tan obvio.
* En el Modo CTR la decryption function posee una vulnerabilidad que es argumentalmente seria. El ciphertext del CTR es “maleable”, en el sentido que alterando cualquiera de sus bits, inducirá que los correspondientes bits en el plaintext también se alternen tras la desencriptación. Por lo tanto, si un adversario conoce el par plantext-ciphertext y puede inducir a la decryption function a usar el counter block del par conocido, entonces, mediante la elección de cuales bits alternan en el ciphertext conocido, el adversario puede controlar el resultado de la desencriptación, hasta el largo del plaintext conocido.  
  Esta vulnerabilidad motiva combinar el mecanismo de encriptación CTR con un mecanismo de autenticación, tal como ocurre en GCM. La aseguración de la autenticación en GCM depende crucialmente de la unicidad de las IVs.  
  En particular, si cualquier IV es alguna vez repetida para la authentication encryption function de GCM para una key dada, entonces es probable que el adversario sea capaz de determinar la hash subkey a partir de los ciphertexts resultantes.

### Aseguración de la Autenticación

* La creación de un authentication tag provee el mecanismo por lo cual la conformidad de la autenticidad del plaintext y del AAD (y del IV) puede ser obtenida sobre el output de la authenticated decryption function. La naturaleza de esta conformidad depende del output de la authenticated decryption function:
  + Si el output es el plaintext, entonces el diseño del modo provee fuerte, pero no absoluta conformidad que la fuente pretendida de los datos creó el tag, en decir, que el plaintext, la AAD, el IV, y el tag son autenticos.
  + Si el output es FAIL, entonces es con certeza que al menos uno de los inputs brindados no es auténtico.
* La conformidad no es absoluta, pues las falsificaciones son posibles, en principio.
* GCM no está bien equipado para el uso de tags cortas o mensajes demasiado largos.
* Si “n” denota el número total de bloquesde ccodificado del ciphertext y la AAD, entonces existe un método de construir un ciphertext “dirigido” para falsificar, el cual se espera que tenga éxito con una probabilidad de aproximadamente n/2^t.
* Un adversario puede intentar sistemáticamente a adivinar la muchas diferentes tags para un input dado para la authenticated decryption, y eventualmente acertará. Por esa razón, el sistema o protocolo que implemente GCM debe monitoreas y, si es necesario, limitar el número de intentos de verificación no exitosos para cada key.
* Como la mayoría de los modos de operación por cifrado de bloque, la certeza de seguridad de GCM se degrada a medida que más datos son procesados con una misma key. Por lo tanto, el número total de bloques de plaintext y AAD que son protegidos por invocación de la authentication encryption function durante el tiempo de vida de la key debe ser limitado. Un límite razonable para la mayoría de las aplicaciones suele ser 2^64, consiste con el requerimiento del número de invocaciones antes nombrado.

### Requerimientos y Guías para el Uso de Tags Cortos

* Para algunas aplicaciones devoz y video, authentication tags cortos pueden ser apropiados.
* Sin embardo, incluso en aplicaciones de voz y video, los tags cortos pueden ser problemáticos para GCM, debido a los ataques de falsificación dirigida. De no cumplir con los requerimientos y guías en esta sección, puede ser sencillo para el atacante reproducir la hash subkey H, después de lo cual la conformidad de autenticación se pierde completamente.
* Guía:
  1. Los paquetes que fallan el chequeo de integridad dentro de la authenticated decryption function deben ser silenciosamente descartados. En otras palabras, el protocolo/sistema de control sobre el cual los paquetes son recibidos no debe proveer respuesta ACK/NACK respecto a la integridad de los paquetes individuales. Sin embargo, el receptor deber registrarlos errores de autenticación internamente –en una manera que sea indetectable del otro lado del canal de comunicación- y terminar la conexión o notificar al usuario si el porcentaje de errores excede lo que se considera normal. Ésta es la práctica de seguridad estándar con cualquier protocolo/sistema y cualquier elección de algoritmo.
  2. La AAD dentro de los paquetes debe limitarse a ser la información de cabecera necesaria y no debe contener mensajes que deban ser autenticados junto con los datos encriptados.
  3. Los paquetes puede llevar una secuencia de fragmentos de voz o datos visuales, pero un paquete individual no debería llevar un .txt o .doc entero. Idealmente, los datos del plaintext que subyace la encriptación no debe ser demasiado estereotipado o adivinable.
  4. El protocolo/sistema de control debe establecer una nueva key GCM –y por lo tanto una nueva hash subkey, H- frecuetnemente, dependiendo del largo máximo combinado del ciphertext y la AAD que puede existir dentro de un solo paquete. Además, para tags de 32 bits, este largo combinado debe ser muy pequeño –en el orden de decenas o centenas de bytes; tags de 64 bits extienden el largo combinado hasta millones de bytes.
* Las tablas 1 y 2 cuantifican las recomendaciones para tags de 32 y 64 bits respectivamente.  
  Para cualquier implementación que soporte tags de 32 o 64 bits, una de las filas en la tabla 1 o 2, respectivamente, debe ser aplicada:  
  

### Protección Contra la Reproducción de Mensajes

* GCM, así como muchos otros mecanismos de autenticación, no previenen a un adversario de interceptar el output de una invocación de la authenticated encryption y “replicarla” para la authenticated decryption después.  
  Es por ello que el protocolo/sistema de control debe proteger contra tales eventos, mediante el monitoreo de duplicación de cualquier IV que se haya presentado ante la authenticated decryption. Ejemplos de tal información incluyen un mensaje secuencia o un timestamp.

## SOBRE MACsec

* Ver macsec.pdf
* Ver 802.1AEbn-2011
  + Uso de GCM-AES en MACsec: Cipher Suite es el nombre que toma AES-GCM en MACsec para encriptar el User Ethertype y los datos del paquete.
  + AES necesita K, IV, A, P, C, T. MACsec le provee dichos parámetros de la siguiente manera:
    - K es el SAK de 128 buts
    - Los 64 MSB del IV de 96 bits son los octetos del SCI, codificados como numero binario. Los restantes 32 bits son los octetos del PN, como numero binario.
    - T es el ICV, y su largo es de 128 bits.
    - Cuando El Cipher Suite es usado solo para proteger la Integridad (AES-GMAC):
      * A es la MAC Address de Destino, Origen, y los octetos del SecTAG y User Data, concatenados en ese orden.
      * P es nulo.
      * La Secure Data son los octetos del User Data, sin modificar.
    - Cuandos el Cipher Suite es usado para proteger Confidencialidad, sin un offset del confidencialidad:
      * A es la MAC Address de Destino, Origen, y los octetos del SecTAG, concatenados en ese orden.
      * P son los octetos del User Data.
      * La Secure Data es C.
    - Cuandos el Cipher Suite es usado para proteger Confidencialidad, con un offset del confidencialidad:
      * A es la MAC Address de Destino, Origen, y los octetos del SecTAG y los primeros octetos del offset de confidencialidad del User Data, concatenados en ese orden.
      * P son los restantes octetos del User Data.
      * La Secure Data son los primeros octetos del offset de confidencialidad del User Data concatenado con C, en ese orden.