The Analysis of Galois Substitution Counter Mode (GSCM)

Mohamed Abo El-Fotouh and Klaus Diepold
Institute for Data Processing (LDV)
Technische Universität München (TUM)
80333 München, Germany
[mohamed,kldi]@tum.de

Abstract

In [9], GSCM mode of operation for authenticated encryption was presented. GSCM is based on the Galois/Counter Mode (GCM). GSCM is an enhancement of GCM, which is characterized by its high throughput and low memory consumption in network applications. In this paper, we propose some enhancements to GSCM and compare it with the different implementations of GCM. We present stability, performance, memory and security analyses of different implementations of GSCM and GCM.

1. Introduction

The number of internet users is increasing continually world wide. Recent statistics reported the current number of internet users is more than 1.5 billion. This number has increased more than 330% in the last eight years and is still increasing daily [28]. Consequently, internet and network applications need to serve an increasing number of concurrent clients. The enhanced quality and performance expected from modern applications require more bandwidth capacity to meet the clients' needs. Today, modern networks have to fulfill the demand of higher transmission rates and in the same time provide data security and especially data confidentiality [16].

Key agility is particularly important in applications where only several blocks of data are encrypted between two consecutive key changes. IPSec [17, 18, 19] and ATM [7], with small sizes of packets, and consecutive packets encrypted using different keys, are two widespread protocols in which the key setup latencies may play a very important role [11].

Galois/Counter Mode (GCM) [23] is a block cipher mode of operation that uses universal hashing over a binary Galois field to provide authenticated encryption. GCM is build on the counter mode (CTR) [22] and it has been standardized by NIST [26]. There is a number of different software algorithms that implements universal hashing over a binary Galois field, these implementations vary from their speed and memory requirements.

In this paper, we studied different software implementations of GCM. Our analysis pointed out some shortcomings in each scheme, as the scheme that uses on-the-fly computation is considered slow, on the other hand the schemes that uses pre-computed tables use more memory, which may limit the number of concurrent clients.

In [9], GSCM was proposed to overcome the shortcomings of the current schemes. GSCM is based on the Static Substitution Model (SSM) [8]. The SSM model can provide a block cipher with a secondary key. The secondary key is used to replace some bits of the cipher's expanded key. The SSM model is used to construct a variant of the AES, the name of this variant is AES2S. AES2S was used to construct an encryption scheme for network applications. In this scheme, each client possesses two keys. The first key is shared with a group of clients (a cluster) and it is expanded in memory (cluster key). The second key is the client's unique session key. Encryption and decryption are done using AES2S, where the cluster key is used as the expanded AES key and the client's session key is used as the secondary key. This scheme enjoys high throughput together with low memory consumption.

This paper is structured as follows. In Section 2, we present the current encryption scheme together with our general assumptions and propose a method to generate the secondary keys for GSCM. In Section 3, we extend the definition of AES2S and GSCM proposed in [9], to support all the key lengths of AES [25]. In Section 4, we present memory analysis of different implementations of GCM and GSCM. In Section 5, we present software simulations to help us better understand the behavior of the schemes in real systems. In Section 6, we present the security analysis of GCM and GSCM schemes . We conclude in Section 7.

2 Current schemes

2.1 General assumptions

1. We have a Server that serves N concurrent secure sessions.
2. We are going to use the AES [25, 6] for encryption and decryption.
3. The encryption/decryption is done in GCM mode [26], which has emerged as the best method for high-speed authenticated encryption [23].
4. Each client has two keys, one for each traffic flow (the same method used in IPSec [17, 18, 19], when The Internet Key Exchange (IKE) [14] is used to establish fresh keys).

2.2 Counter block format

Following the guidelines in [15, 29], we assume that the counter block used in counter mode, has the following format:

1. The first 32-bit are a nonce, which are random and unique for each client.
2. The next 64-bit are the initialization vector (IV), which are random and incremented with each packet.
3. The last 32-bit are initialized for each packet by one, and incremented for each 128-bit block within the packet.

2.3 Secondary keys generation

In our proposed models, we have two classes of keys: the cluster keys and the clients keys. We propose to use three keys for each cluster, two main keys (Cluster Encryption Key **CEK** and Cluster Decryption Key **CDK**) and the users' key (Cluster User Key **CUK**). These three keys are unique for each cluster and are generated using a cryptographic secure random number generator. **CEK** and **CDK** are used to generate the cluster encryption and decryption expanded keys using AES key scheduling. **CUK** is used to generate the clients' unique keys using the counter mode (CTR) [22]. In this way, we guarantee that each client has an unique pair of secondary keys.

Note that if one of the two secondary keys of the client is known, the second key can be calculated by knowing only one pain of plaintext/ciphertext. Thus, the secondary keys MUST be unique and unpredictable.

2.4 GCM software implementations

GCM mode combines the well-known counter mode [22] of encryption with the Galois mode of authentication. The key feature is that the Galois field multiplication used for authentication can be easily computed in parallel thus permitting higher throughput than the authentication algorithms that use chaining modes, like CBC [24].

We tested GCM with the different GHASH implementation strategies, using on-the-fly strategy and pre-computed tables using 256 bytes, 4Kb tables with Shoups method [**?**], 8Kb tables [12] and 64Kb with the straightforward method. We also implemented AES with pre-computed subkeys and generating the subkeys on-the-fly.

2.5 GCM-Pre(x)

GCM-Pre(x) represent a group of five schemes, each scheme uses different value of x. These values are 0, 256, 4096 (4K), 8192 (8K) and 65536 (64K) and they represent the amount of memory required in bytes for GHASH computation for each traffic flow per client. The pre-computed tables are computed at the beginning of the session, stored in memory and then recalled whenever a GHASH computation is needed. It executes as follows:

- **Setup Routine,** is executed for every client number i (C_i), once it is connected:
  - Two unique cryptographic random keys (k_i^1 and k_i^2) of length 128-/256-bit are generated.
  - Two random IVs (IV_i^1 and IV_i^2) of length 64-bit are generated.
  - Two unique nonces (n_i^1 and n_i^2) of size 32-bit are generated.
  - k_i^1, k_i^2, IV_i^1, IV_i^2, n_i^1 and n_i^2 are send to the client.
  - k_i^1 and k_i^2 are expanded, using AES encryption key setup algorithm to produce E_i and D_i.
  - T_i^1 and T_i^2 (tables of size x bytes used to compute the GHASH function in the encryption and decryption directions respectively) are computed, using k_i^1 and k_i^2.
  - E_i, D_i, IV_i^1, IV_i^2, n_i^1, n_i^2, T_i^1 and T_i^2 are stored in the server's memory.
- **Encryption Execution Routine,** to encrypt a plaintext (PT) for C_i:
  - E_i, IV_i^1, n_i^1 and T_i^1 are fetched from the server's memory.
  - IV_i^1 is incremented by one.

- IV_i^1 and n_i^1 are used to construct the initial counter block (ICB_i) for the CTR mode (as explained in Sect. 2.2).
- ICB_i is used to encrypt PT using GCM mode to produce CT, where E_i serves as the encryption expanded key and T_i^1 is used to calculate the authentication tag t.
- CT, IV_i^1 and t are send to the client, note that IV_i^1 is send to the client to ensure that the client can generate the key stream needed for decryption, even when some packets are lost or reordered [15].

- **Decryption Execution Routine,** to decrypt a ciphertext (CT) for C_i:
  - D_i, n_i^2 and T_i^2 are fetched from the server's memory.
  - IV_i^2, CT and t are received from the client.
  - The authentication tag t' is calculated using T_i^2 and if t=t' the next step is executed, otherwise an authentication error is returned.
  - IV_i^2 and n_i^2 are used to construct the initial counter block (ICB_i) for the CTR mode (as explained in Sect. 2.2).
  - ICB_i is used to decrypt CT using GCM mode to produce PT, where D_i serves as the encryption expanded key (note that the encryption function of the cipher is used in the decryption process of the CTR mode).

2.6 GCM-On(x)

GCM-On(x) represent a group of five schemes, each scheme uses different value of x. These values are 0, 256, 4096 (4K), 8192 (8K) and 65536 (64K) and they represent the amount of memory required in bytes for GHASH computation for each traffic flow per client. The pre-computed tables are computed at the beginning of the session, stored in memory and then recalled whenever a GHASH computation is needed. It executes as follows:

- **Setup Routine,** is executed for every client number i (C_i), once it is connected:
  - Two unique cryptographic random keys (k_i^1 and k_i^2) of length 128-/256-bit are generated.
  - Two random IVs (IV_i^1 and IV_i^2) of length 64-bit are generated.
  - Two unique nonces (n_i^1 and n_i^2) of size 32-bit are generated.
  - k_i^1, k_i^2, IV_i^1, IV_i^2, n_i^1 and n_i^2 are send to the client.
  - T_i^1 and T_i^2 (tables of size x bytes used to compute the GHASH function in the encryption and decryption directions respectively) are computed, using k_i^1 and k_i^2.
  - k_i^1, k_i^2, IV_i^1, IV_i^2, n_i^1, n_i^2, T_i^1 and T_i^2 are stored in the server's memory.

- **Encryption Execution Routine,** to encrypt a plaintext (PT) for C_i:
  - k_i^1, IV_i^1, n_i^1 and T_i^1 are fetched from the server's memory.
  - IV_i^1 is incremented by one.
  - IV_i^1 and n_i^1 are used to construct the initial counter block (ICB_i) for the CTR mode (as explained in Sect. 2.2).
  - ICB_i is used to encrypt PT using GCM mode to produce CT, where k_i^1 serves as the encryption key (the subkeys are calculated on-the-fly) and T_i^1 is used to calculate the authentication tag t.
  - CT, IV_i^1 and t are send to the client, note that IV_i^1 is send to the client to ensure that the client can generate the key stream needed for decryption, even when some packets are lost or reordered [15].

- **Decryption Execution Routine,** to decrypt a ciphertext (CT) for C_i:
  - k_i^2, n_i^2 and T_i^2 are fetched from the server's memory.
  - IV_i^2, CT and t are received from the client.
  - The authentication tag t' is calculated using T_i^2 and if t=t' the next step is executed, otherwise an authentication error is returned.
  - IV_i^2 and n_i^2 are used to construct the initial counter block (ICB_i) for the CTR mode (as explained in Sect. 2.2).
  - ICB_i is used to decrypt CT using GCM mode to produce PT, where k_i^2 serves as the encryption key (the subkeys are calculated on-the-fly), note that the encryption function of the cipher is used in the decryption process of the CTR mode.

3 New Scheme

3.1 AES2S

AES2S is a variant of AES presented in [9]. It is constructed using the SSM model [8]. It accepts two 128-bit secondary keys. We redefine AES2S to support 128-bit keys. The listing of AES2S is found in table 1, where:

Table 1. AES2S encrypting function.

```
Encrypt-AES2S( P , EK , K1 , K2 )
KL=len(EK)
if(KL=1408)
      x=5
else
      x=9
end if
Substitute( EK , K1 , 0 )
Substitute( EK , K2 , x )
C=Encrypt-AES( P , EK )
return C
```

X: is the input plaintext, that will be encrypted using AES2S.

EK: is the expanded AES encryption key.

K1: is the first part of the secondary key of size 128-bit.

K2: is the second part of the secondary key of size 128-bit.

Substitute(EK,K1,i): replaces the i^{th} 128-bit of EK with K1 (Note that: the first round of the AES is round zero and it is the pre-whitening process).

len(X): returns the size of X in bits.

Encrypt-AES(X,EK): encrypts X using AES encryption routine, with EK as the expanded encryption key and return the result.

C: the output ciphertext.

In AES2S, two rounds subkeys are replaced:

1. The subkeys of the pre-whitening round are replaced with **K1**.
2. The subkeys of the x^{th} round are replaced with **K2**.

3.2 GSCM(x)^n

AES2S is used to build a scheme for high-speed networks, where:

- (K_{c1} and K_{c2}) are the cluster keys used as the AES secret keys, and are shared by n clients (where n is the maximum size of a cluster).
- Each client i (C_i) has its own two unique 128-bit keys (k_i^1) and (k_i^2).

GSCM(x)^n tries to eliminate the setup latencies, it executes as follows:

- **Cluster setup routine:** is used to prepare the system and is executed for each cluster of n clients.
  - Two cryptographic random keys (K_{c1}) and (K_{c2}) with length 128-/256-bit are generated.
  - K_{c1} and K_{c2} are expanded, using AES encryption key setup algorithm to produce the cluster's shared encryption expanded subkeys (E_c) and the cluster's shared decryption expanded subkeys (D_c).
  - T_{c1} and T_{c2} (tables of size x bytes used to compute the GHASH function in the encryption and decryption directions respectively) are computed, using K_{c1} and K_{c2}.
  - E_c, D_c, T_{c1} and T_{c2} are stored in the server's memory.
- **Client setup routine:** is executed for every client number i (C_i), once it is connected:
  - Two unique cryptographic random keys (k_i^1 and k_i^2) of length 256-bit are generated.
  - Two random IVs (IV_i^1 and IV_i^2) of length 64-bit are generated.
  - Two unique nonces (n_i^1 and n_i^2) of size 32-bit are generated.
  - k_i^1, k_i^2, K_{c1}, K_{c2}, IV_i^1, IV_i^2, n_i^1 and n_i^2 are send to the client.
  - k_i^1, k_i^2, IV_i^1, IV_i^2, n_i^1 and n_i^2 are stored in the server's memory.
- **Encryption execution routine,** to encrypt a plaintext (PT) for C_i:
  - E_c, IV_i^1, n_i^1 and T_{c1} are fetched from the server's memory.
  - IV_i^1 is incremented by one.
  - IV_i^1 and n_i^1 are used to construct the initial counter block (ICB_i) for the GCM mode (as explained in Sect. 2.2).
  - ICB_i is used to encrypt PT using GCM mode to produce CT using AES2S (with k_i^1 and k_i^2 as its secondary keys), where E_i serves as the encryption expanded key and T_{c1} is used to calculate the authentication tag t.
  - CT, IV_i^1 and t are send to the client, note that IV_i^1 is send to the client to ensure that the client can generate the key stream needed for decryption, even when some packets are lost or reordered [15].
- **Decryption execution routine,** to decrypt a ciphertext (CT) for C_i:

– D_c, n_i^2 and T_{c2} are fetched from the server's memory.

– IV_i^2, CT and t are received from the client.

– The authentication tag t' is calculated using T_{c2} and if t=t' the next step is executed, otherwise an authentication error is returned.

– IV_i^2 and n_i^2 are used to construct the initial counter block (ICB_i) for the CTR mode (as explained in Sect. 2.2).

– ICB_i is used to decrypt CT using GCM mode to produce PT using AES2S (with k_i^1 and k_i^2 as its secondary keys), where D_i serves as the encryption expanded key (note that the encryption function of the cipher is used in the decryption process of the CTR mode).

4 Memory Analysis

The less the memory the scheme needs, the more available memory to other applications and the larger the number of concurrent clients the server can serve. Memory access has a great role in the overall scheme performance. If a server has insufficient physical memory space to cache all of the data it needs, it performs page replacements. Although virtual memory makes the server able to complete the process, this process is very slow compared to the cost of data computation [20].

Table 2 represents the memory requirements in bytes for the GCM schemes per client to hold the key material and other data required to perform encryption and decryption (e.g IVs, nonces, counter, ...etc). From these results, it is clear that GSCM(x) schemes possess the lowest memory requirements and thus can server the maximum number of concurrent clients. GSCM(x) schemes can server at least twice the number of clients of the current schemes and in some cases this factor increases to about 1029.

Table 2. Memory requirements for GCM schemes per client (in bytes)

	128-bit key	256-bit key
GCM-Pre(64k)	131624	131752
GCM-Pre(8k)	16936	17064
GCM-Pre(4k)	8744	8872
GCM-Pre(256)	1064	1192
GCM-Pre(0)	552	680
GCM-On(64k)	131336	131400
GCM-On(8k)	16648	16712
GCM-On(4k)	8456	8520
GCM-On(256)	776	840
GCM-On(0)	264	328
GSCM(x)	128	128

5 Simulation Analysis

5.1 Server Configuration

We implemented NoCrypto and the other tested schemes using C++ language and run a simulation to examine their practical behavior. Table 3 shows the server configuration used in our simulations. Note that NoCrypto does not perform any encryption or decryption functions, it is illustrated here to show the cryptographic overhead and in our proposed schemes all the clients share the same cluster.

Table 3. Server configuration.

Processor	Intel Xeon Quad-Core 2.33GHz(64-bit)
RAM	4096 MB
Processor Cache	12 MB
Paging file	4096 MB
OS	Microsoft Windows Server 2008
Compiler	Visual C++ 2005
Code optimization	Maximum speed

5.2 Parameters

- α is the tested packet size, we have chosen α to be either 40 or 1500 bytes. As the current packet sizes seem mostly bimodal at 40 and 1500 bytes [13, 27].
- UNIT is the current number of clients.
- STEP is a number used to calculate the current number of clients in a given simulation instance.
- Z_i is the current number of clients served by the serve, We construct the set Z = { Z_i }, where Z_i = (i $\times$ STEP).

5.3 The Simulation

We constructed a multi-client/server TCP socket application to demonstrate the behavior of the schemes. The server and the clients are connected via a LAN (100 Mbps). The scenarios work as follows:

1. The server allocates 90% of its RAM as a shared data pool. This pool hold the encrypted and decrypted data for all the client and is used to illustrate the effect of reading and writing to the RAM.
2. The server allocates the memory needed by the tested scheme to serve Z_i clients, where for each client, the server allocates M bits, where M is the number of bits required by each client using the tested scheme (see table 4).
3. The server waits until 50 computers are connected, then send the start command to all the computers (each computer simulates Z_i/50 clients). Note that each computer is served using a different thread, to illustrate the effect of multi-threading.
4. As the computer receives the start command, it sends a packet of size α to the server.
5. The packet is decrypted for client C_i and encrypted to client C_x, where i and x are positive random numbers less than Z_i.
6. The server sends the encrypted packet (from the previous step), to the computer that serves client C_x.
7. When the packet is processed by the server and received by the computer, the computer starts to send a packet for the next client it simulates.
8. The average time (in milliseconds) for processing a packet is reported.
9. We ran the simulation, where STEP=100,000 with the following termination conditions:

Table 4. Maximum reported number of clients and stable number of clients for GCM schemes (in 10,000).

	MaxR		MaxS	
	128-bit	256-bit	128-bit	256-bit
GCM-Pre(0)	85	70	70	55
GCM-On(0)	190	150	175	135
GCM-Pre(256)	45	45	35	30
GCM-On(256)	60	50	45	40
GCM-Pre(4k)	20	20	5	5
GCM-On(4k)	20	20	5	5
GCM-Pre(8k)	20	20	5	5
GCM-On(8k)	20	20	5	5
GCM-Pre(64k)	NA	NA	NA	NA
GCM-On(64k)	NA	NA	NA	NA
GSCM(*)	200	195	185	180

(a) When the average packet processing time for a packet exceeds that of NoCrypto with a factor of 2 (using the same parameters).

(b) The server can not allocate memory for Z_i clients.

(c) The server begins to loss its connections.

5.4 Maximum Stable Number of Clients

Table 4 summarizes the simulation results, where we reported the maximum number of clients reported in our simulation, together of the maximum number of stable clients reported. These numbers are the average of that with α=40 and α=1500. It is worth to mention that the maximum number of stable clients with α=1500 is greater than that with α=40.

From these results, all GSCM schemes can serve the same number of stable clients. In case of K=128, our simulation reported that GSCM schemes can server about 265% and 105% stable clients as GCM(0)-Pre and GCM(0)-On respectively, these values increase to 320% and 130% (when K=256). For the schemes that uses 256 pre-computed tables, GSCM schemes can server about 530% and 410% stable clients as GCM(256)-Pre and GCM(256)-On respectively (when K=128), and these persantages increase to 600% and 450% (when K=256). For both schemes that uses 4K and 8K pre-computed tables, GSCM schemes can server from 3600% to 3700% stable clients than the other schemes. In case of GCM(64k)-Pre and GCM(64k)-On, the simulation program failed to allocate the required memory.

5.5 Network Performance Analysis

We performed a similar simulation like in 5.3 , in this simulation we set Z = { Z_i }, where Z_i = 10,000 and $1 \leq i \leq 10$. To measure the performance of the schemes in real network.

Figure 1, presents the simulation results for the fastest schemes of each category. GSCM(64k) is considered the fastest scheme of GSCM(x), GCM(4k)-Pre is the fastest GCM(x)-Pre and GCM(4k)-On is the fastest GCM(x)-On. Where GSCM(64k) is faster than GCM(4k)-Pre and GCM(4k)-On specially in the small packets size case, where it is about 5% faster than GCM(4k)-Pre, almost as fast as GCM(4k)-On (K=128) and 8% faster than GCM(4k)-On (K=256). It is worth to mention that for large packest all the schemes have almost the same speed.

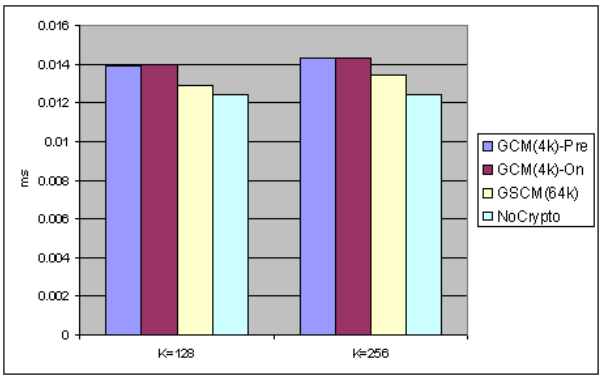
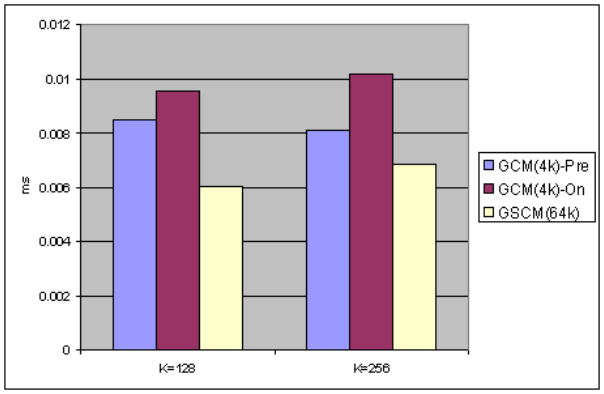


Figure 1. The fastest GCM schemes, average time (in ms) needed to process a packet (α = 40)

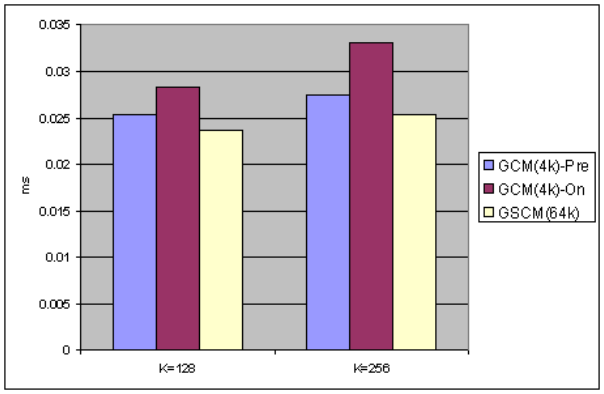
5.6 CPU Utilization Analysis

We performed a similar simulation like in Sect. 5.5 , but we only use the multi-threading to simulate the clients (i.e. we removed the network overhead), this is to measure the CPU utilization for each scheme.

Figure 2, presents the simulation results for the fastest schemes of each category. GSCM(64k) is considered the fastest scheme of GSCM(x), GCM(4k)-Pre is the fastest GCM(x)-Pre and GCM(4k)-On is the fastest GCM(x)-On. Where GSCM(64k) is faster than GCM(4k)-Pre and GCM(4k)-On specially in the small packets size case, where it is about 19 to 40% faster than GCM(64k)-Pre and 49% to 58% faster than GCM(4k)-On. On the other hand, it seems that GSCM(64k) is faster than GCM(4k)-Pre with about 7 to 8% and it is faster than GCM(4k)-On with about 20% to 30% (when α=1500).



(a) (α=40 and STEP=10,000).



(b) (α=1500 and STEP=10,000).

Figure 2. The fastest GCM schemes, average time (in ms) needed to process a packet

6 Security Analysis

The security of GCM is based on that of CTR and AES, as GCM uses AES in CTR mode in its encryption. The security of GSCM is based on that of CTR and AES2S, as GSCM uses AES2S in CTR mode in its encryption.

6.1 Security of CTR.

Birthday attacks on CTR mode remain possible even when the underlying block cipher is ideal [1], and CTR encryption becomes insecure once 2^{64} (in case of AES) blocks have been encrypted, in the sense that at this point partial information about the message begins to leak, due to birthday attacks [2]. Therefore, the server MUST generate a fresh key (random and unused key) before 2^{64} blocks are encrypted with the same key for each client.

6.2 Security of AES.

We assume that, the best attack known against the full AES is to try every possible 128-/256-bit key (i.e., perform an exhaustive key search), this requires about 2^{127}/2^{255} trails [6].

6.3 Security of AES2S

For an attacker that does not know either the primary key and the secondary key, she tries to attack a full round AES. After the first secondary key substitution, 4 to 8 rounds of the AES are performed, these rounds assures that not only full confusion and diffusion after the injection are achieved [21], but also that any differential and linear propagation is completely destroyed in the encryption direction (as the secondary keys for each client are unique and random (Sect. 2.3)). Using two subkeys substitution that are random and unique, prevent the attacker to introduce any controlled difference in the middle of the cipher, thus eliminating the possibility of a chosen plaintext attack. Thus, the attacker can not mount linear, differential, chosen plaintext and AES2S is considered secure.

Note that in GSCM(x) schemes, only the encryption of AES2S is needed, as the counter mode use the encryption function in both the encryption and decryption direction.

6.4 Security of the schemes

The security of GCM(x)-Pre and GCM(x)-On schemes are inherited from the security of the CTR mode and that of the AES. As the attacker can either attack the mode of operation or the cipher itself. The server should generate a fresh key (for each client) before 2^{64} blocks are encrypted with the same key, to avoid birthday attacks on CTR. We recommend to encrypt maximum 2^{32} blocks for each client (which is sufficient to encrypt the largest possible IPv6 jumbogram [4]), then generates a fresh key for that client.

The security of GSCM(x)^n schemes is inherited from that of the CTR mode and that of AES2S. As CTR mode security is not based on the used block cipher. There are two kind of attackers on GSCM(x)^n, when attacking AES2S (key search attack):

1. An outside attacker **A** (not a client served by the server) that watches the ciphertext. For an external attacker (that does not possess the primary key), she needs to attack AES with two random subkeys.

2. An inside attacker **B** (most probably a current user or a recent revoked user from the cluster), that would like to attack another client that is luckily in the same cluster. This attacker knows the primary key:

   (a) The attacker decrypts the known ciphertext with the known primary key until the 5^{th} or 10^{th} round (depending on the primary key size) to produce the intermediate state γ.

   (b) By knowing the input to AES2S, the attacker can reduce AES2S to an Even-Mansour construction [10], where K1 and K2 are the key and the reduced AES (4 or 9 rounds) is considered as the (Pseudo)random permutation.

   (c) The security of Even-Mansour is:

      i. About 2^{255}, using exhaustive search over the key space (K1 and K2 are both of size 128-bit), which is considered large enough by todays standards.

      ii. Daemen demonstrated in [5] that a known plaintext attack, will take on average 2^{127} calculations, which has the same complexity as attacking AES with 128-bit key, which is considered secure with todays technology.

      iii. Daemen also demonstrated in [5] that a chosen plaintext attack, will take on average 2^{64} calculations using 2^{64} stored blocks. By limiting the number of encrypted blocks per client, this attack can be avoided (see Sect. 6.1, as each client encrypts maximum 2^{32} blocks using the same secondary key, if for some application more data is needed to be encrypted the client can join a new cluster "using a fresh secondary key" or new fresh secondary key can be generated for that client).

      iv. Biryukov-Wagner demonstrated in [3], that a "sliding with a twist" attack allows an adversary to recover the key using $\sqrt{2} \times 2^{64}$

known plaintexts and $\sqrt{2} \times 2^{64}$ work. By limiting the number of blocks encrypted per client using the same secondary key, this attack can be avoided (see Sect. 6.1).

So GSCM(x)^n is upper bounded with the security of AES2S and lower bounded with the security of Even-Mansour.

7 Conclusion

In this paper, we analyze the new authenticated encryption GSCM, with its predecessor GCM. Our analysis illustrates that GSCM is superior than GCM, by possessing high throughput, consuming the lowest amount of memory, serving the largest number of concurrent clients and it is also considered secure. The simulation results demonstrate the high-throughput of GSCM, as it is faster than the current schemes up to 60% in processing small packets and up to 30% in processing large packets, in the same time GSCM can serve up to 1029 times the maximum number of clients served by the current schemes. We recommend to use GSCM(64k) to benefit of its high performance.

References

[1] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS '97)*, page 394, Washington, DC, USA, 1997. IEEE Computer Society.

[2] M. Bellare, T. Krovetz, and P. Rogaway. Luby-Rackoff backwards: Increasing security by making block ciphers non-invertible. *Lecture Notes in Computer Science*, 1403, 1998.

[3] A. Biryukov and D. Wagner. Advanced Slide Attacks. In *Advances in Cryptology—Eurocrypt '00 Proceeding*, 2000.

[4] D. Borman, S. Deering, and R. Hinden. IPv6 Jumbograms. RFC 2675, August 1999.

[5] J. Daemen. Limitations of the Even-Mansour Construction. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*. LNCS, Springer-Verlag, 1991.

[6] J. Daemen and V. Rijmen. AES Proposal: Rijndael. http://citeseer.ist.psu.edu/daemen98aes.html, 1998.

[7] J. Dunn and C. Martin. Terminology for ATM Benchmarking. RFC 2761, Feb 2000.

[8] M. El-Fotouh and K. Diepold. Dynamic Substitution Model. In *The Fourth International Conference on Information Assurance and Security (IAS'08)*, Naples, Italy, September 2008.

[9] M. El-Fotouh and K. Diepold. Galois Substitution Counter Mode (GSCM). In *International Workshop on Security and Privacy in Enterprise Computing (InSPEC 2008) in conjunction with the 12th IEEE International EDOC Conference (EDOC 2008)*, Munich, Germany, September 2008.

[10] S. Even and Y. Mansour. A Construction of a Cipher from a Single Pseudorandom Permutation. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 10(3):151–161, Summer 1997.

[11] K. Gaj and P. Chodowiec. Hardware performance of the AES finalists - survey and analysis of results. http://ece.gmu.edu/crypto/AES_survey.pdf, 2000.

[12] B. Gladman. `http://fp.gladman.plus.com/AES/index.htm`, August 2008.

[13] C. Greg. The nature of the beast: Recent Traffic Measurements from an Internet backbone. http://citeseer.ist.psu.edu/673025.html, 1998.

[14] D. Harkins and D. Carrel. The Internet Key Exchange (IKE), 1998.

[15] R. Housley. Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP), 2004.

[16] O. Jung, S. Kuhn, C. Ruland, and K. Wollenweber. Enhanced Modes of Operation for the Encryption in High-Speed Networks and Their Impact on QoS. In *ACISP '01: Proceedings of the 6th Australasian Conference on Information Security and Privacy*, pages 344–359, London, UK, 2001. Springer-Verlag.

[17] S. Kent and R. Atkinson. IP Authentication Header. RFC 2402, Nov 1998.

[18] S. Kent and R. Atkinson. IP Encapsulating Security Payload (ESP). RFC 2406, Nov 1998.

[19] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, Nov 1998.

[20] T. Liang, Y. Liu, and C. Shieh. Adding Memory Resource Consideration into Workload Distribution for Software DSM Systems. In *CLUSTER*, pages 362–369, 2003.

[21] L. May, M. Henricksen, W. Millan, G. Carter, and E. Dawson. Strengthening the Key Schedule of the AES. In *ACISP '02: Proceedings of the 7th Australian Conference on Information Security and Privacy*, pages 226–240, London, UK, 2002. Springer-Verlag.

[22] D. McGrew. Counter Mode Security: Analysis and Recommendations. http://citeseer.ist.psu.edu/mcgrew02counter.html, 2002.

[23] D. McGrew and J. Viega. The Galois/Counter Mode of Operation (GCM). http://citeseer.ist.psu.edu/mcgrew04galoiscounter.html, 2004.

[24] A. Menezes, P. V. Oorschot., and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[25] J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti, and E. Roback. Report on the Development of the Advanced Encryption Standard (AES). Technical report, 2000.

[26] NIST. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D, November 2007.

[27] R. Sinha, C. Papadopoulos, and J. Heidemann. Internet Packet Size Distributions: Some Observations. Technical Report ISI-TR-2007-643, USC/Information Sciences Institute, May 2007. Orignally released October 2005 as web page `http://netweb.usc.edu/~rsinha/pkt-sizes/`.

[28] I. W. Stats. WORLD INTERNET USAGE AND POPULATION STATISTICS. http://www.internetworldstats.com/stats.htm, March 2009.

[29] J. Viega and D. McGrew. The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP), 2005.