ANALISIS PEMANFAATAN POLY1305-AES DALAM AUTENTIKASI PESAN DAN KELEBIHANNYA DIBANDING ALGORITMA MAC LAINNYA

Arif Nanda Atmavidya (13506083)

Program Studi Informatika, Institut Teknologi Bandung,
Email : atmavidy@yahoo.com

Abstrak - Message *Authentication Code* (MAC) adalah teknik autentikasi pesan dengan membandingkan nilai *authentication tag* yang telah dihitung oleh pengirim dengan *authentication tag* yang dihitung sendiri oleh penerima. MAC merupakan fungsi *hash* satu arah yang menggunakan kunci rahasia (*secret key*) dalam pembangkitan nilai *hash*. Kunci digunakan oleh penerima pesan untuk verifikasi nilai *hash*. Oleh karena key yang digunakan oleh pengirim dan penerima pesan sama, maka algoritma MAC ini tidak mendukung prinsip *non-repudiation*.

Dengan alasan itulah Daniel J. Bernstein melakukan penelitian dan menemukan algoritma baru yang diberi nama Poly1305-AES. Pada beberapa sumber dijelaskan bahwa nilai lebih utama dari algoritma Poly1305-AES ini jika dibandingkan dengan algoritma MAC biasa ialah terkait keamanannya dimana tingkat keamanannya setara dengan AES. Bahkan, jika terdapat kegagalan dalam kunci AES-nya, maka pengguna dapat mengubah algoritma Poly1305-AES ini menjadi Poly135 yang memanfaatkan algoritma lainnya.

Berdasarkan penjelasan singkat tersebut, penulis merasa tertarik untuk mengangkat algoritma Poly1305-AES ini dan mengujicoba serta menganalisisnya lebih jauh tentang penggunaannya dalam pengautentikasian pesan, serta kelebihannya dibandingkan dengan algoritma MAC lainnya, terkait performansi dan variabel pembanding lainnya.

Kata kunci : Poly1305-AES, *Message Authentication Code*, AES.

A. Pendahuluan

1. *Message Authentication Code* (MAC)

Message Authentication Code (MAC) merupakan teknik autentikasi pesan dengan cara membandingkan nilai *authentication tag* yang telah dihitung oleh pengirim dengan *authentication tag* yang dihitung sendiri oleh penerima. MAC dalam proses autentikasinya menggunakan sebuah *secret key* yang mutlak harus dimiliki oleh pengirim dan penerima pesan, berbeda dengan MD5 biasa yang tidak memerlukan kunci. MAC bukanlah tanda-tangan digital dan hanya menyediakan otentikasi pengirim dan integritas pesan saja.

Keunggulan utama MAC dibanding dengan algoritma MD5 biasa ialah terkait pada modifikasi oleh virus. Jika pada MD5, bisa saja setelah virus mengubah isi dokumen tersebut virus juga mengubah nilai MD5-nya dengan menghitung nilai *message diggest* baru dari dokumen yang telah diubah oleh virus tersebut. Namun pada MAC, walaupun virus telah mengubah dokumen aslinya. Namun tanpa mengetahui kunci rahasia untuk MAC-nya, virus tersebut tidak akan mampu menciptakan MAC baru untuk dokumen yang telah ia modifikasi.

Dalam implementasi MAC-nya sendiri, biasanya digunakan algoritma kriptografi modern yang berbasis aliran *(cipher block)*, misalnya CBC, CFB ataupun modifikasi dari algoritma *cipher block* tersebut. Contoh algoritma modifikasi dari *cipher block* ialah *Data Authentication Algorithm* (DAA) yang merupa-kan hasil modifikasi algoritma DES *(Data Encryption Standard)* dan CBC *(Cipher Block Chain)*. Adapun yang akan menjadi MAC-nya ialah bagian blok terakhir dari hasil enkripsi *cipher block* tersebut.

2. *Advanced Encryption Standard* (AES)

Advanced Encryption Standard (AES), disebut juga algoritma Rinjndael, merupakan algoritma enkripsi yang ditemukan oleh Vincent Rijmen dan Joan Daemen dari Belgia dan telah ditetapkan sebagai algoritma kriptografi standard oleh *National Institute of Standard and Technology* (NIST). Algoritma ini termasuk dalam algoritma kriptografi *cipher block*. Rijndael mendukung panjang kunci 128 bit sampai 256 bit dengan step 32 bit.

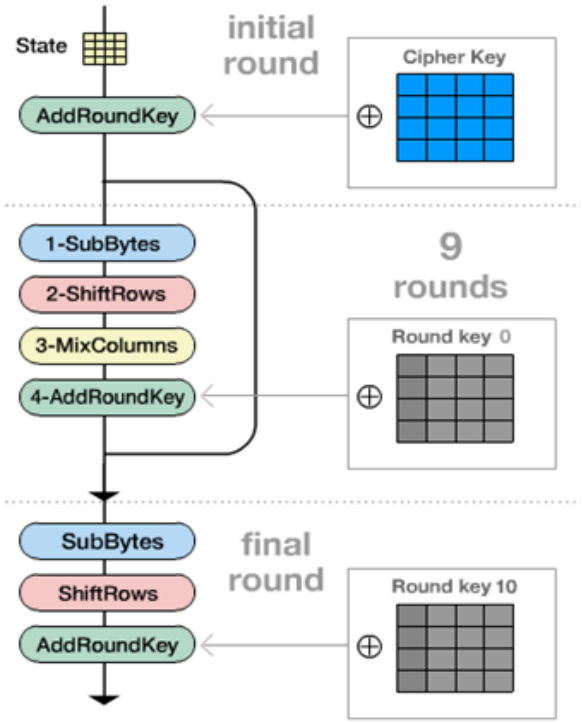
Panjang kunci dan ukuran blok dapat dipilih secara independen. Setiap blok dienkripsi dalam sejumlah putaran *(round)* tertentu, sebagaimana halnya pada DES. Karena *AES* ini menetapkan panjang kunci adalah 128, 192, dan 256, maka dikenal AES-128, AES-192, dan AES-256.

Garis besar Algoritma Rijndael yang beroperasi pada blok 128-bit dengan kunci 128-bit adalah sebagai berikut (di luar proses pembangkitan *round key*) :

1. AddRoundKey, melakukan XOR antara state awal (plainteks) dengan *cipher key*. Tahap ini disebut juga initial round.

2. Putaran sebanyak Nr – 1 kali. Proses yang dilakukan pada setiap putaran adalah :
   a. SubBytes, substitusi byte dengan menggunakan tabel substitusi (S-box).
   b. ShiftRows, pergeseran baris-baris array state secara wrapping.
   c. MixColumns, mengacak data di masing-masing kolom array state.
   d. AddRoundKey, melakukan XOR antara state sekarang round key.

3. Final round, proses untuk putaran terakhir :
   a. SubBytes
   b. ShiftRows
   c. AddRoundKey



B. Pembahasan Poly1305-AES

1. Spesifikasi Poly1305-AES

Pesan

Pesan yang diautentikasi oleh Poly1305-AES ini merupakan urutan byte-byte m[0]; m[1]; … m[l - 1] dimana tiap byte-nya merupakan bilangan integer positif antara 0 sampai 255.

Kunci

Kunci yang digunakan ialah 32-byte *secret key* yang di-*share* antara *sender* dan *receiver*, dan terbagi menjadi 2 bagian. 16-byte pertama kunci AES *k*, dan yang kedua ialah sebuah string 16-byte. Bagian kedua tersebut merepresentasikan 128-bit integer *r* dalam format *little endian*.

Beberapa bit tertentu dari r diset dengan nilai kosong, untuk r[3], r[7], r[11], r[15], empat buah bit awalnya dibuang, dan untuk r[4], r[8], r[12], dua buah bit akhirnya dibuang. Sehingga pada akhirnya terdapat 2^{106} kemungkinan dari *r*.

Nonces

Poly1305-AES membutuhkan 16-byte *nonce* yang bernilai unik, artinya *sender* seharusnya tidak memakai *nonce* yang sama untuk dua buah pesan yang berbeda. Poly1305-AES memasukkan *nonce n* melalui AES_k untuk membangkitkan 16-byte string AES_k (n).

Konversi dan *Padding*

Sebuah pesan m[0]; m[1]; … m[l - 1] dan q = $\lceil l/16 \rceil$. Kemudian didefinisikan integer c_1, c_2, …, $c_q \in \{1, 2, 3, \ldots, 2^{129}\}$.

Jika 1 < i < $\lfloor l/16 \rfloor$, maka didapat :

$$c_i = m[16i - 16] + 2^8\, m[16i - 15] + 2^{16}\, m[16i - 14] + \ldots + 2^{120}\, m[16i - 1] + 2^{128};$$

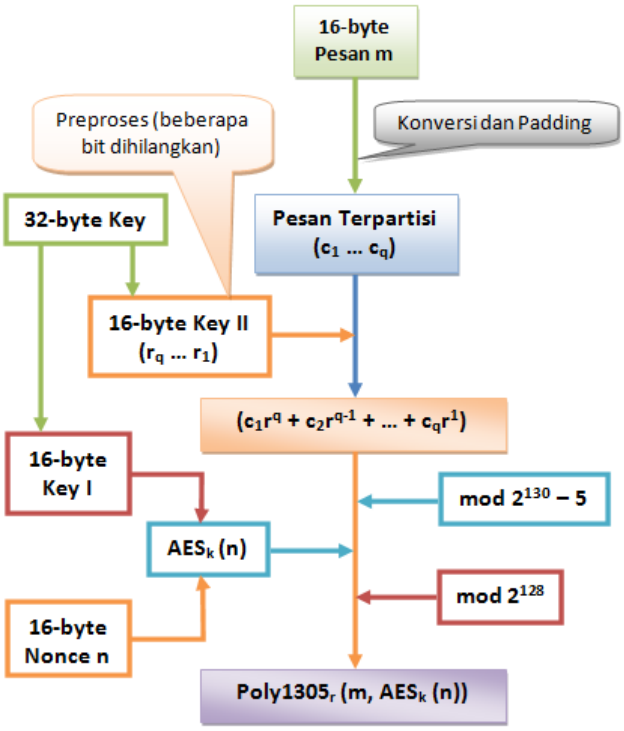
Namun jika l bukan kelipatan 16 :

$$c_q = m[16q - 16] + 2^8\, m[16q - 15] + \ldots + 2^{8\,(l \bmod 16) - 8}\, m[l - 1] + 2^{8\,(l \bmod 16)};$$

Authenticator

$Poly1305_r(m, AES_k(n))$, *authenticator* Poly1305-AES dari sebuah pesan m dengan *nonce* n dalam *secret key* (k,r) dapat didefinisikan sebagai 16-byte *unsigned little-endian* dari :

$$\mathbf{(((c_1 r^q + c_2 r^{q-1} + \ldots + c_q r^1) \bmod 2^{130} - 5) + AES_k(n)) \bmod 2^{128}}$$



Bagan di atas memperlihatkan bagaimana pemrosesan pesan sepanjang 16-byte dengan algoritma Poly1305-AES menggunakan 16-byte nonce, dan 32-byte *secret key* yang dibagi menjadi 16-byte kunci pertama, dan 16-byte string kunci kedua.

2. Algoritma Poly1305-AES

Kode C++ di bawah ini ialah proses pembacaan k dari k[0], k[1], …, k[15], membaca r dari r[0], r[1], …, r[15], membaca $AES_k(n)$ dari s[0], s[1], …, s[15], membaca m dari m[0], m[1], hingga m[15], dan meletakkan $Poly1305_r(m, AES_k(n))$ ke dalam out[0], out[1], hingga out[15].

```
#include <gmpxx.h>

void poly1305_gmpxx(unsigned char *out,
const unsigned char *r,
const unsigned char *s,
const unsigned char *m,unsigned int l)
{
    unsigned int j;
    mpz_class rbar = 0;

    for (j = 0;j < 16;++j)
        rbar += ((mpz_class) r[j]) << (8 * j);

    mpz_class h = 0;
    mpz_class p = (((mpz_class) 1) << 130) - 5;

    while (l > 0) {
        mpz_class c = 0;
        for (j = 0;(j < 16) && (j < l);++j)
            c += ((mpz_class) m[j]) << (8 * j);
        c += ((mpz_class) 1) << (8 * j);
        m += j; l -= j;
        h = ((h + c) * rbar) % p;
    }

    for (j = 0;j < 16;++j)
        h += ((mpz_class) s[j]) << (8 * j);

    for (j = 0;j < 16;++j) {
        mpz_class c = h % 256;
        h >>= 8;
        out[j] = c.get_ui();
    }
}
```

Library gmpxx.h merupakan library yang menangani tipe data komposit integer pada algoritma Poly1305-AES ini. Kode tersebut di atas hanyalah kode sederhana yang menggambarkan (model) pengaplikasian algoritma Poly1305-AES, sedangkan pada kenyataannya, untuk mengaplikasikan algoritma kriptografi tersebut butuh kode yang lebih banyak dan kompleks sehingga keunggulan algoritma tersebut dari algoritma MAC lainnya dapat terlihat.

3. Keamanan Poly1305-AES

Jaminan Keamanan

Poly1305-AES menjamin bahwa jalan satu-satunya bagi *attacker* untuk menemukan (n,m,a) sehingga terpenuhi $a = Poly1305_r(m, AES_k(n))$ dalam autentikasi pesan ialah dengan memecahkan AES (selain mendapatkan sendiri kuncinya dari *sender*). Jika *attacker* tidak mampu memecahkan AES (mengubah nilai MAC lama menjadi MAC baru) dan hanya mengubah isi dokumennya, maka $a \neq Poly1305_r(m, AES_k(n))$.

Berikut ini pembuktian kuantitatifnya. Diasumsikan *attacker* mengetahui pesan asli C dan mencoba untuk melakukan pemalsuan pesan sebesar D. Diasumsikan pula *attacker* memiliki peluang δ dalam membedakan AES_k dari permutasi acak setelah C + D. Dengan panjang keseluruhan pesan ialah L. Maka, dengan probabilitas sebesar :

$$1 - \delta - \frac{\left(1 - \frac{C}{2^{128}}\right)^{-(C+1)/2} 8D\lceil L/16 \rceil}{2^{106}}$$

semua pemalsuan yang dilakukan oleh *attacker* dapat ditolak. Jika $\leqslant 2^{64}$, maka kemungkinan perubahan yang dilakukan oleh *attacker* akan sukses ialah sekitar :

$$\delta + \frac{1{,}649.8D\left\lceil \frac{L}{16} \right\rceil}{2^{106}} < \delta + 14D\left\lceil \frac{L}{16} \right\rceil / 2^{106}$$

Tujuan utama dari AES dalam lingkup probabilitas ialah mendapatkan nilai δ yang sekecil mungkin. Jika suatu saat nanti sudah ada yang membuktikan dengan metode tertentu bahwa δ tidak lagi kecil, mungkin AES dapat mudah dipecahkan. Namun demikian, Poly1305-AES tetap bisa diganti dengan Poly1305-XX dengan XX merupakan algoritma *cipher block* yang diyakini dapat lebih kuat dari AES saat itu. Untuk keamannya pun, maka keamanan dari Poly1305-XX akan tergantung (relatif) terhadap algoritma XX tersebut.

4. Kelebihan Poly1305-AES Dibanding Algoritma MAC Lainnya

Kelebihan algoritma Poly1305-AES ini dibanding MAC lainnya ialah :

- *Guaranteed security if AES is secure*
  Untuk dapat memecahkan Poly1305-AES ini jalan satu-satunya ialah dengan memecahkan AES, sedangkan di satu sisi, hingga saat ini AES masih dianggap sebagai algoritma kriptografi yang cukup kuat saat ini dan telah dijadikan standard kriptografi setidaknya selama 10 tahun.
- *Cipher replaceability*
  Jika terjadi suatu kegagalan pada AES, pengguna dapat mengganti Poly1305-AES dengan Poly1305 fungsi lain.
- *Extremely high speed*
  Dari pengujian pengautentikasian 1024-byte pesan pada beberapa komputer yang berbeda spesifikasi, pada AMD Athlon dibutuhkan 3843 *cycles*, Pentium III 5361 *cycles*, Pentium 4 5464 *cycles*, Pentium M 4611 *cycles*, serta PowerPC seri 7410 8464 *cycles*. Artinya kecepatan autentikasi tersebut konsisten untuk semua jenis CPU.
- *Low per-message overhead*
  Dari pengujian pengautentikasian 64-byte pesan pada beberapa komputer yang berbeda spesifikasi, pada Pentium 4 1232 *cycles*, PowerPC seri 7410 1264 *cycles*, serta pada UltraSPARC III 1077 *cycles*. Artinya kecepatannya tersebut konsisten untuk berbagai ukuran pesan, tidak hanya pesan yang panjang saja.

C. Kesimpulan

Dari pembahasan yang telah dilakukan, diperoleh kesimpulan sebagai berikut :

- Poly1305-AES merupakan jenis algoritma MAC yang dipergunakan untuk mengautentikasi keaslian pesan dengan menggunakan 32-byte *secret key* yang di-*share* antara *sender* dan *receiver* supaya *receiver* dapat membandingkan nilai *authentication tag* yang telah dihitung oleh pengirim *(sender)* dengan *authentication tag* yang dihitung sendiri oleh penerima *(receiver)*.
- Poly1305-AES ini menggunakan algoritma kriptografi *cipher block* AES dalam proses enkripsi pesannya. Sehingga, tingkat keamanannya setara dengan tingkat keamanan AES, dan untuk memecahkan Poly1305-AES ini jalan satu-satunya ialah dengan memecahkan AES.
- Algoritma ini mengkomputasi 128-bit (16 bytes) pesan, menggunakan 128-bit kunci AES, 128-bit kunci tambahan, dan 128-bit *nonce*.
- Poly1305-AES ini memiliki kelebihan dibanding algoritma MAC lain dalam sisi keamanan yang lebih terjamin, proses komputasi yang lebih cepat, *cipher* yang dapat diganti dari AES menjadi algoritma kriptografi *cipher block* lain, serta kecepat komputasinya yang tidak hanya terbatas pada pesan berukuran pendek saja.

D. Pustaka

[1] Munir, Rinaldi. 2008. *Kriptografi*. Bandung : Penerbit Informatika
[2] Poly1305-AES, a state-of-the-art message-authentication code. http://cr.yp.to/mac.html
[3] Bernstein, Daniel. 2005. *The Poly1305-AES message-authentication code*. Chicago : The University of Illinois at Chicago
[4] Poly1305-AES. http://en.wikipedia.org/wiki/Poly1305-AES.html

E. Lampiran

Tabel berikut memperlihatkan proses autentikasi untuk string dengan panjang 2, 0, 32, dan 63. Notasi m(r) berarti $c_1r^q + c_2r^{q-1} + \ldots + c_qr^1$. Data-data ditampilkan dalam bentuk hexadecimal.

m	f3 f6
c_1	00000000000000000000000000001f6f3
r	85 1f c4 0c 34 67 ac 0b e0 5c c2 04 04 f3 f7 00
$\underline{m}(r) \bmod 2^{130} - 5$	321e58e25a69d7f8f27060770b3f8bb9c
k	ec 07 4c 83 55 80 74 17 01 42 5b 62 32 35 ad d6
n	fb 44 73 50 c4 e8 68 c5 2a c3 27 5c f9 d4 32 7e
$\mathrm{AES}_k(n)$	58 0b 3b 0f 94 47 bb 1e 69 d0 95 b5 92 8b 6d bc
$\mathrm{Poly1305}_r(m, \mathrm{AES}_k(n))$	f4 c6 33 c3 04 4f c1 45 f8 4f 33 5c b8 19 53 de
m	
r	a0 f3 08 00 00 f4 64 00 d0 c7 e9 07 6c 83 44 03
$\underline{m}(r) \bmod 2^{130} - 5$	00000000000000000000000000000000
k	75 de aa 25 c0 9f 20 8e 1d c4 ce 6b 5c ad 3f bf
n	61 ee 09 21 8d 29 b0 aa ed 7e 15 4a 2c 55 09 cc
$\mathrm{AES}_k(n)$	dd 3f ab 22 51 f1 1a c7 59 f0 88 71 29 cc 2e e7
$\mathrm{Poly1305}_r(m, \mathrm{AES}_k(n))$	dd 3f ab 22 51 f1 1a c7 59 f0 88 71 29 cc 2e e7
m	66 3c ea 19 0f fb 83 d8 95 93 f3 f4 76 b6 bc 24
	d7 e6 79 10 7e a2 6a db 8c af 66 52 d0 65 61 36
c_1	124bcb676f4f39395d883fb0f19ea3c66
c_2	1366165d05266af8cdb6aa27e1079e6d7
r	48 44 3d 0b b0 d2 11 09 c8 9a 10 0b 5c e2 c2 08
$\underline{m}(r) \bmod 2^{130} - 5$	1cfb6f98add6a0ea7c631de020225cc8b
k	6a cb 5f 61 a7 17 6d d3 20 c5 c1 eb 2e dc dc 74
n	ae 21 2a 55 39 97 29 59 5d ea 45 8b c6 21 ff 0e
$\mathrm{AES}_k(n)$	83 14 9c 69 b5 61 dd 88 29 8a 17 98 b1 07 16 ef
$\mathrm{Poly1305}_r(m, \mathrm{AES}_k(n))$	0e e1 c1 6b b7 3f 0f 4f d1 98 81 75 3c 01 cd be
m	ab 08 12 72 4a 7f 1e 34 27 42 cb ed 37 4d 94 d1
	36 c6 b8 79 5d 45 b3 81 98 30 f2 c0 44 91 fa f0
	99 0c 62 e4 8b 80 18 b2 c3 e4 a0 fa 31 34 cb 67
	fa 83 e1 58 c9 94 d9 61 c4 cb 21 09 5c 1b f9
c_1	1d1944d37edcb4227341e7f4a721208ab
c_2	1f0fa9144c0f2309881b3455d79b8c636
c_3	167cb3431faa0e4c3b218808be4620c99
c_4	001f91b5c0921cbc461d994c958e183fa
r	12 97 6a 08 c4 42 6d 0c e8 a8 24 07 c4 f4 82 07
$\underline{m}(r) \bmod 2^{130} - 5$	0c3c4f37c464bbd44306c9f8502ea5bd1
k	e1 a5 66 8a 4d 5b 66 a5 f6 8c c5 42 4e d5 98 2d
n	9a e8 31 e7 43 97 8d 3a 23 52 7c 71 28 14 9e 3a
$\mathrm{AES}_k(n)$	80 f8 c2 0a a7 12 02 d1 e2 91 79 cb cb 55 5a 57
$\mathrm{Poly1305}_r(m, \mathrm{AES}_k(n))$	51 54 ad 0d 2c b2 6e 01 27 4f c5 11 48 49 1f 1b