Studi Mengenai *Fully Homomorphic Encryption* dan Perkembangannya dari RSA sebagai Enkripsi Homomorfis Populer

Akhmad Ratriono Anggoro – NIM : 13505003

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if15003@students.if.itb.ac.id

Abstrak

Makalah ini membahas mengenai perkembangan Fully Homomorphic Encryption yang diajukan oleh Craig Gentry, sebuah terobosan baru dalam enkripsi yang menerapkan fungsi homomorfis dalam teknologi kriptografi.namun dengan menggunakan semua fungsi aljabar (perkalian dan penjumlahan) secara bersamaan dan jumlah operasi yang tidak terbatas. Metode yang diajukan beliau secara langsung telah menyelesaikan masalah dalam enkripsi homomorfis yaitu tidak mampunya teknik enkripsi sebelumnya untuk menerapkan fungsi-fungsi aljabar sepenuhnya untuk menghilangkan atau lebih tepatnya mendaya gunakan properti buruk pada enkripsi homomorfis (seperti RSA), yaitu dengan dua atau lebih *ciphertext* seseorang yang tidak diharapkan mampu mendekripsi *ciphertext* tersebut menjadi *plaintext* tanpa mengetahui fungsi untuk memetakan *plaintext* tersebut dalam *outsourcing* pada *cloud computing*.

***Kata kunci**: Homomorphic Encrption,* RSA, *Privacy homomorphic encryption, cloud computing,* enkripsi, dekripsi.

1. Latar Belakang

Homomorphic Encryption adalah teknik dalam kriptografi untuk mengenkripsi *plaintext* menggunakan satu atau lebih operasi aljabar. Secara kasar, homomorfisme berarti terdapat dua atau lebih fungsi yang memiliki struktur aljabar sama. Namun, dalam penerapannya pada kriptografi, pemetaan teks hanya bisa dilakukan dalam satu jenis operasi aljabar. Homomorfisme pada kriptografi kebanyakan hanya mampu menerapkan salah satu dari penjumlahan dan perkalian. Ini menjadi masalah besar untuk keamanan transmisi data karena seseorang atau mesin dapat menyerang *stream* data tanpa mengetahui fungsi pemetaan homomorfisnya sama sekali.

Padahal, seandainya ada yang mampu menerapkan banyak operasi perkalian dan penjumlahan sekaligus dalam enkripsi tanpa memperbesar ukuran dari *ciphertext*, enkripsi homomorfis menjadi sangat kuat. Program enkripsi yang dibangun dapat bekerja di operasi enkripsi terhadap input dan menghasilkan enkripsi dari output yang dihasilkan. Bisa dikatakan bila hal yang tidak bisa dipecahkan selama 30 tahun ini berhasil dibuat, tidak ada yang bisa menebak-nebak tipe operasi untuk kemudian merusak *stream* data yang menggunakan enkripsi homomorfis.

Namun setahun yang lalu, Craig Gentry berhasil menciptakan enkripsi yang memanfaatkan sepenuhnya homomorfisme dari operasi aljabar. Beliau berhasil menciptakan sebuah enkripsi, dengan memanfaatkan kriptografi *lattice-based*, yang mampu menggunakan sepenuhnya properti-properti dari prinsip homomorfisme sehingga mampu mempertahankan struktur ring dari pemetaan teks. Makalah ini bermaksud untuk menelaah lebih dalam temuan dari Craig Gentry beserta keunggulannya dibanding enkripsi homomorfis kuno

.

1.1 Rumusan Masalah

Makalah ini membahas kelemahan dari enkripsi homomorfis dan kelemahannya terutama enkripsi RSA yang sangat populer, bagaimana homomorfisme sejati sangat sulit dikembangkan dalam kriptografi, dan bagaimana Craig Gentry menemukan solusi dari masalah yang tidak dapat dipecahkan ahli kriptografi selama tiga puluh tahun.

1.2 Batasan Masalah

Makalah ini hanya menelaah secara detail mengenai properti pada *homomorphic encryption,* pengembangan *privacy homomorphism* berangkat dari kelemahan RSA, dan bagaimana gambaran umum desain *fully homomorphic encryption* karya Craig Gentry bekerja. LaluMakalah tidak akan menjelaskan secara rinci penggunaannya dalam aplikasi penghitungan pada *cloud computing*.

1.3 Metode Penelitian

Penelitian dilakukan dengan studi literatur mendalam mengenai perkembangan *homomophic encryption* beserta aplikasinya terutama RSA yang membuat terobosan tidak sengaja dalam kelemahan desainnya yang dapat digunakan untuk keperluan *cloud computing.*

2. RSA

Enkripsi homomorfis adalah sebuah enkripsi yang menerapkan fungsi operasi aljabar spesifik pada plaintext dan kemudian melakukan fungsi aljabar lain (bisa sama ataupun berbeda) pada ciphertext-nya dengan mempertahankan properti homomorfis matematisnya. Misalkan terdapat plaintext P yang dienkripsi sehingga menghasilkan ciphertext C, mendekripsi 2C akan menghasilkan 2P, inilah yang disebut properti homomorfis. Salah satu contoh dari enkripsi homomorfis yang populer adalah RSA. RSA adalah sebuah terobosan dalam enkripsi dengan kunci publik yang digunakan terutama dalam protokol E-commerce untuk memvalidasi kebenaran data-data pribadi yang dikirimkan melalui jaringan internet.

2.1 Operasi pada RSA

Operasi pada RSA terdiri dari pembangkitan kunci (publik dan privat, mengingat metode kunci asimetris yang dipakai dalam RSA), enkripsi dari plaintext menggunakan public key yang dibangkitkan, dan dekripsi menggunakan kunci privat.

2.2 Pembangkitan Kunci

Pembangkitan kunci pada RSA memanfaatkan operasi-operasi sebagai berikut,

1. Pilih dua integer prima berbeda p dan q, untuk alasan keamanan, p dan q dibangkitkan secara random dan memiliki jumlah bit yang sama.
2. Hitung n=pq, n akan digunakan sebagai modulus untuk kunci publik dan privat.
3. Hitung totient $\varphi(pq) = (p-1)(q-1)$.
4. Pilih bilangan e sehingga $1<e<\varphi(pq)$, fpb dari e dan $\varphi(pq)$ hanya 1; sebagai catatan, e sebaiknya memiliki banyak nol dan jumlah bit kecil.
5. Pilih d dengan aritmatika modular sehingga d memenuhi

$$de \equiv 1 \pmod{\varphi(pq)}.$$

Kunci publik yang dibangkitkan terdiri dari (n,e) dan dipublikasikan umum sedangkan kunci privatnya, adalah d yang harus dirahasiakan oleh pemegang kunci.

2.3 Enkripsi dan Dekripsi

Misalkan Jono hendak memberikan pesan plaintext P kepada Joni yang telah mengumumkan kunci publiknya (n,e), Maka Jono harus mengenkripsi pesan P tersebut dengan sebelumnya mengubah P menjadi integer p yang memenuhi $0<p<n$ dengan operasi *padding* reversibel sehingga menghasilkan ciphertext C dengan formula,

$$C = p^{e} \bmod n \quad (i)$$

Jono kemudian mengirimkan ciphertext tersebut ke Joni melalui koneksi internet. Joni, yang sedang berada di belahan dunia lain, ketika hendak mengetahui isi pesan P dari Jono haruslah memanfaatkan kunci privatnya untuk mendekripsi C,

$$C^{d} \equiv p \pmod{n} \quad (ii)$$

Setelah mendapat p, Joni dapat mengubahnya menjadi plaintext P dengan membalik operasi *padding* pada p.

2.4 Masalah pada RSA

RSA hanya menggunakan operasi multiplikasi, tentu saja mengingat dasar operasi pada RSA, ini bisa menjadi masalah besar. Hanya dengan memiliki kunci publik, seseorang dapat melakukan dekripsi pada plaintext, bahkan yang telah di padding sekalipun.
Secara umum masalah RSA adalah sebagai berikut, Misal terdapat kunci publik (n,e) yang menghitung P menjadi C seperti pada *(i)*, karena basis dari desain RSA yang mengharuskan n adalah produk dari dua bilangan prima besar, $2 < e < N$ bersekutu dengan $\varphi(N)$ hanya pada 1, dan $0 \leq C < N$, maka terdapat cara untuk memperoleh kunci privat yaitu dengan memfaktorisasi N sedemikian rupa sehingga memperoleh p dan q. P dan Q sementara ini kemudian dapat diturunkan sehingga mendapat kunci privat yang bisa diuji untuk mendekripsi ciphertext. Permasalahan ini juga berlaku untuk semua enkripsi homomorfisme lainnya karena kecenderungannya untuk memanfaatkan hanya satu jenis operasi. Satu-satunya yang bisa diharapkan untuk menghentikan serangan yang memanfaatkan desain *malleable* pada enkripsi homomorfisme seperti RSA adalah memperbesar jumlah bit kunci sehingga serangan bisa diminimalkan dengan menambah jumlah waktu untuk dapat 'mencoba-coba' kunci privat hasil faktorisasi dari kunci.

Kelemahan lain yang dapat ditemukan adalah
Dengan dua ciphertext yang berbeda seseorang bisa menghitung dengan tepat kombinasi dari dua plaintext. Hal ini didasarkan pada sifat enkripsinya yang homomorfis,

$$C2 = p1^{e} \bmod n$$
$$C1 = p2^{e} \bmod n$$
$$C1*C2 = (p1*p2)^{e} \bmod n$$

Semakin banyak ciphertext diperoleh maka semakin besar celah keamanan yang dapat ditembus.

3. Fully Homomorphic Property

Homomorfisme pada kriptografi seharusnya memanfaatkan semua properti homomorfisme matematis sebagai contoh enkripsi P yang menghasilkan C maka dekripsi 2+2C akan menghasilkan 2+2P, operasi penjumlahan dan perkalian yang tidak terbatas akan menghilangkan properti *malleable* seperti enkripsi-enkripsi homomorfisme pada umumnya yang hanya memanfaatkan satu operasi aljabar.

Salah satu desain yang diberikan Craig Gentry adalah *Fully Homomorphic Encryption*

4 Fully Homomorphic Encryption

Adalah desain untuk enkripsi homomorfis lengkap yang memanfaatkan jaringan vektor merentang ideal. Secara umum, pertama-tama merancang sebuah skema enkripsi yang dapat mengevaluasi sirkuit dekripsinya sendiri. Kemudian mendesain skema enkripsi kunci dengan *ideal lattice*. Karena sifat dari *ideal lattice* yang merupakan kombinasi linear dengan integer dari dirinya sendiri, operasi penjumlahan dan perkalian dapat dilakukan pada enkripsinya. Desain baru yang dikembangkan peneliti IBM Craig Gentry sebenarnya adalah langkah penerus *privacy homomorphism* yang didesain untuk cloud computing.

4.1 Privacy Homomorphism

Rivest, Adleman, dan Dertouzos berangkat dari kelemahan mendasar pada desain RSA yang memungkinkan pihak yang tidak berhak memanipulasi data tanpa mengetahui kunci sama sekali berfikir, bagaimana seandainya keadaan ini dimanfaatkan untuk penghitungan bersama, seperti pada *cloud computing* yang mengizinkan siapapun untuk ikut menghitung tanpa harus mendekripsikan ciphertext?

Adapun desain yang diterapkan Rivest et al. adalah sebagai berikut,

1. N=pq (N sebagai kunci publik, r dan q sebagai kunci privat).
2. Enkripsi; untuk plaintext P terdapat sub-himpunan a, fungsi enkripsi E adalah

$$E(a) = (a \bmod r, a \bmod q)$$

   Ciphertext = Pq x Pr
3. Proses dekripsi diselesaikan dengan teorema *chinese remainder,* dengan fungsi dekripsi D,

$$D(a_1,a_2) = a_1 q q^{-1} + a2 r r^{-1} (\bmod N)$$
$$a_1 = a \bmod q,\ a_2 = a \bmod r$$

Desain yang diberikan oleh Rivest ini membuka paradigma baru dalam mendesain sebuah enkripsi yang memungkinkan data diproses di tempat tidak aman tanpa harus mendekripsi data tersebut. Namun, sebuah penjumlahan dalam desainnya terbukti telah menjadi kelemahan utama dalam keamanannya.

4.2 Desain Fully Homomorphic Encryption

Craig Gentry memulai desain skema enkripsi barunya dengan berdasar pada *lattice based encryption*. Seperti yang telah diterangkan sebelumnya, enkripsi homomorfis lengkap menggunakan operasi penjumlahan dan perkalian. Sekarang coba berfikir dalam binary dan menganggap operasi ini adalah sirkuit XOR (penjumlahan) dan sirkuit AND (perkalian), maka skema yang didesain adalah sekumpulan sirkuit ini dan membatasi faktor dari operasi sebelumnya dengan jumlah AND saja. Skema ini mengijinkan untuk mendapatkan operasi XOR dan/atau AND yang tidak terbatas.

Dalam kriptografi berbasis jaringan vektor, enkripsi bekerja pada jaringan multi dimensi dan menempatkan pada suatu tempat dekat dengan sebuah titik pada rangkaian vektor tersebut, sedangkan dekripsi bertujuan untuk menemukan titik tersebut. Walaupun terlihat mudah, setiap jaringan vektor memiliki struktur internal, sehingga penyerang tidak akan pernah tahu jalan yang paling dekat untuk menuju titik tersebut. Kriptografi ini bermasalah untuk operasi AND yang terlalu banyak dimana error bisa terjadi. Dalam *lattice based*, properti homomorfisme terbatas hanya pada kedalaman tertentu.

Gentry sendiri telah mendesain skema miliknya sedemikian rupa sehingga berbeda dengan *lattice based encryption* biasa. Error yang diperoleh dapat dikurangi dengan apa yang diajukan Gentry, yaitu mengijinkan server untuk menerima kunci yang telah dimasukkan melalui email mendekripsi data secara homomorfis. Jadi ketika data dienkripsi dengan kunci k1,maka client juga harus menyediakan kunci kedua k2 dan juga menyerahkan kopi dari k1 yang telah dienkripsi oleh dengan k2, mesin kemudian dapat bekerja cukup dengan versi k1 yang telah dienkripsi karena sifat homomorfis yang dimiliki oleh kunci-kunci tadi.

Jadi, pada bentuknya yang paling sederhana client harus menyediakan himpunan kunci publik k1, k2, k3,… yang bagian privat dari masing-masing kunci tersebut terdapat dalam bentuk terenkripsi pada key selanjutnya. Jumlah kunci ini tentunya mewakili kedalaman sirkuit operasi dalam satu proses homomorfis lengkap.

4.3 Hal yang Mungkin Dicapai Dari Fully Homomorphic Algorithm

Pada dasarnya, *Fully Homomorphic Algorithm*, memberikan solusi mendasar yang telah dipertanyakan oleh Rivest et al. yaitu bagaimana mengijinkan orang tak dikenal melakukan proses penghitungan terhadap data pribadi terenkripsi tanpa mengetahui isi dari data itu sendiri, dengan kata lain, kita mengijinkan *party* yang tidak harus kita kenal untuk mengolah data terenkripsi untuk menghasilkan perhitungan yang efisien. Jadi, *cloud computing* tidak hanya terbatas pada *party* yang memang memiliki autoritas untuk mengakses data, namun siapapun boleh melakukannya karena isi data terenkripsi itu tidak akan diketahui dan diproses semata-mata hanya karena fitur homomorfisnya.

Anggap terdapat ciphertext C yang benar-benar homomorfis. Operasi yang dilakukan pada C untuk mengolah data juga akan berpengaruh langsung pada ciphertextnya.

Sebagai contoh, dalam query yang *client* berikan ke search engine misal ke Google, query ini bentuknya terenkripsi dan diproses google masih dalam bentuk ciphertext. Proses dekripsi baru akan dilakukan setelah query yang telah terproses kembali ke *client*.

Sistem seperti ini memiliki potensi yang tidak terbatas dalam dunia jaringan internet. Spam dalam email bisa terdeteksi dengan mudah dan bahkan *client* bisa meminta, katakanlah, data-data kesehatan atau jaminan investasinya diproses bank atau penyedia jasa kesehatan tanpa mengekspos data pribadi itu sendiri.

4.4 Kelemahan

Desain yang terkesan sangat sempurna bukan berarti tanpa kelemahan. Hingga saat ini, Gentry hanya menyatakan bahwa desain dari enkripsinya sangat sulit untuk diimplementasikan dalam teknologi yang ada saat ini. Berbanding lurus dengan keamanan dari jumlah sirkuit operasi yang diletakkan dalam

suatu enkripsi homomorfis lengkap, semakin banyak sirkuit yang dipakai, semakin panjang kunci dan operasi enkripsi dari kunci-kunci tersebut berantai sehingga jika diimplementasikan pada mesin akan membengkakkan operasinya sampai 30 juta kali lebih lama.

Kemudian karena setiap sirkuit operasi sifatnya penting dalam enkripsi homomorfis lengkap ini, optimasi sederhana dan bahkan Hukum Moore tidak akan mampu diterapkan pada mesin yang menjalankan enkripsi ini.

Sebagai contoh, pada kasus query google yang disampaikan pada bab sebelumnya. Waktu operasi dari desain yang diajukan oleh gentry akan bertambah sebesar satu bilyun lebih lama dibandingkan dengan proses query yang dipakai saat ini. Berdasarkan hukum Moore, akan dibutuhkan sekitar 40 tahun lagi agar enkripsi homomorfis lengkap dapat se-efisien query processing search engine saat ini.

5 Kesimpulan

Perkembangan teknik komputasi baru seperti pada *cloud computing* membutuhkan suatu teknik enkripsi baru untuk memanfaatkan resource yang tidak terbatas dari pengguna komputer dari seluruh dunia untuk menghitung dan memproses data tanpa harus mengetahui isi dari data itu sendiri.

Salah satu pemecahannya adalah memanfaatkan fitur homomorfis matematis pada kriptografi homomorfis. Hal ini didasarkan pada kelemahan properti desain kriptografi homomorfis seperti yang ditunjukkan pada RSA. Bagaimana seandainya kelemahan ini didaya gunakan untuk keperluan *cloud computing* sehingga data bisa diubah, diproses, dimanipulasi oleh siapapun namun esensi data itu sendiri tidak akan pernah diketahui kecuali oleh mereka yang berhak. Permasalahan yang muncul adalah bagaimana mendesain sebuah enkripsi yang sangat kuat namun juga mempertahankan properti homomorfisme secara penuh.

Desain pertama yang diberikan adalah *privacy homomorphism* yang dirancang oleh Rivest et al. walaupun telah memiliki fitur homomorfis penuh, keamanannya diragukan. Sampai tiga puluh tahun kemudian Craig Gentry membuat desain *fully homomorphic encryption* yang bukan hanya memiliki sifat homomorfis tapi juga sangat aman.

Namun, desain yang diajukan Gentry sangat tidak praktis. Semakin tinggi tingkat keamanannya, semakin banyak jumlah sirkuit operasi yang dibutuhkan untuk mengenkripsi data, dan waktu prosesnya bisa membengkak.

Walaupun tidak praktis untuk saat ini, desain yang diberikan Gentry bisa menjadi sebuah terobosan baru dalam teknologi enkripsi sehingga bukannya tidak mungkin *cloud computing* yang benar-benar aman nantinya dapat diterapkan.

Referensi

[1] Gentry, Craig. *Fully Homomorphic Encryption Using Ideal Lattices,* Stanford University, 2009.
.

[2] W. B. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr. *RSA/Rabin functions: certain parts are as hard as the whole*. SIAM J. Computing, April 1988.

[3] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. *Relations among notions of security for public-key encryption.* Springer-Verlag, 1998. Lecture

[4] Bellare, Mihir & Rogaway, Phillip. *Optimal asymmetric encryption—how to encrypt with RSA.* Springer-Verlag, 1994.

[6] Bellare, Mihir & Rogaway, Phillip. *The exact security of digital signatures—how to sign with RSA and Rabin.* Springer Verlag, 1996.

[7]http://www.computerweekly.com/Articles/2009/06/30/236695/cryptography-breakthrough-paves-way-to-secure-cloud.htm, 11 Maret 2010

[8]http://en.wikipedia.org/wiki/Fully_homomorphic_encryption 11 Maret 2010

[9]http://en.wikipedia.org/wiki/Homomorphic, 11 Maret 2010

[10]http://en.wikipedia.org/wiki/Lattice-based_cryptography, 11 Maret 2010

[11]Bricke, Ernest F. & Yacobi, Yacov, *On Privacy Homomorphism,* Bell Communication Research. Springer-Verlag, 1998.

[12] Rivest, R. L., Adleman, L. and Dertouzos, M. L., *On data banks and privacy homomorphisms*, Foundations of Secure Computation, Academic Press, New York, 1978.

[13]Domingo-Ferrer, J, *A new privacy homomorphism and applications in Information Processing Letters*, Dec. 1996

[14] Johan Hastad and Mats Nslund. *The security of individual RSA bits.* In IEEE Symposium on Foundations of Computer Science, pages 510–521, 1998.

[15] Stefan Katzenbeisser. *Recent Advances in RSA Cryptography. Kluwer* Academic Publishers, 2001.

[16] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lov´sz. *Factoring polynomials with rational coefficients.* Mathematische Ann., 261:513–534, 1982.