Studi Analisis dan Implementasi dari Pencarian Data Terenkripsi (*Search on Encrypted Data*)

Haris Amrullah Lubis / 13507038
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
haris.lubis@gmail.com

Abstract— **Pencarian data terenkripsi (Search on Encrypted Data) dilandasi oleh sebuah motivasi mengenai beberapa masalah berikut. Misalkan, Alice memiliki data dalam jumlah banyak. Data tersebut privat, namun dapat diakses kapan saja dan di mana saja, contohnya, email. Selain itu, data tersebut disimpan di dalam sebuah remote server dengan koneksi yang bagus dan biaya penyimpanan yang murah, tetapi tidak terpercaya, contohnya sistem cloud computing. Masalah timbul ketika, kita ambil contoh, Bob mengirim pesan kepada Alice dengan menggunakan kunci publik dari Alice. Gateway dari email ingin menguji apakah email mengandung kata kunci "PENTING" sehingga gateway dapat melakukan filter secara otomatis. Namun, Alice tidak ingin gateway melakukan dekripsi terhadap keseluruhan isi pesan. Mekanisme yang digunakan, seluruhnya akan dibahas di makalah ini, akan menyebabkan Alice hanya perlu menyediakan satu kunci khusus sehingga gateway akan menguji apakah kata "PENTING" merupakan kata kunci dari email tanpa perlu mengetahui sisa isi email. Dari dua paragraf di atas, sekilas sistem ini harus memiliki karakteristik sebagai berikut. Pertama, oleh karena Alice tidak mempercayai server, maka data harus disimpan terenkripsi. Kedua, sistem harus memiliki fitur pencarian data sehingga Alice dapat mencari data tersebut. Ketiga, sistem harus menyediakan pencarian data yang kuat dan efisien.**

***Index Terms*—encrypted data search, homomorphic encryption, block cipher, dan stream cipher.**

I. PENDAHULUAN

Tujuan utama dari diterapkannya pencarian data terenkripsi adalah (1) pengguna layanan *cloud* ingin menyimpan data dalam *servers* penyimpan data, seperti *mail servers* dan *file servers*, tanpa menghilangkan unsur privasi dan keamanan, dan (2) pengguna layanan *cloud* disediakan fungsionalitas-fungsionalitas umum untuk menambah dokumen, mengambil dokumen, mencari dokumen yang mengandung kata kunci yang diinginkan, dll.

Fitur-fitur yang ingin diimplementasikan tersebut merupakan sebuah tantangan tersendiri mengingat kontradiksi antara kenyamanan dan keamanan. Jika kita ambil sebuah analogi, kita menyimpan barang-barang kita di dalam sebuah rumah. Karena kita ingin barang tersebut aman, kita simpan masing-masing barang tersebut di dalam sebuah peti yang terkunci. Ketika kita ingin mencari barang yang diinginkan, misalnya sepatu, kita mau tidak mau harus membuka setiap peti dan melihat isinya, apakah peti yang dimaksud berisi sepatu. Tidak nyaman bukan? Mungkin kita bias mengakalinya dengan memberikan label di setiap peti untuk menyebutkan barang apa yang ada di dalam peti tersebut. Namun, hal tersebut menjadi tidak aman, karena orang-orang yang tidak berkepentingan bias mengambil barang tersebut. Apalagi jika rumah tersebut adalah tempat public yang setiap orang bias masuk di dalamnya.

Itulah analogi yang tepat untuk menggambarkan kondisi atau realita di dunia internet saat ini. Ketika kita sekarang berada di jaman semua hal terhubung ke dunia internet, maka terdapat kontradiksi yang menarik antara kedua hal tersebut, keamanan dan kenyamanan. Apalagi ketika era *cloud computing* telah datang, di mana paradigma orang tentang perangkat lunak dan informasi telah bergeser. Apakah tingkat efisiensi proses bisnis yang tinggi harus dibayar dengan tingkat keamanan yang rendah. Kalau kita bias membuat system yang menyebabkan tingkat keamanan menjadi tinggi sehingga system di atas tidak lagi menjadi sebuah *trade-off*, maka mengapa tidak kita lakukan. Kita ingin data dan informasi kita aman walaupun disimpan di tempat public. Sesuai dengan prinsip-prinsip keamanan informasi kita ingin data dan informasi terjaga integritasnya, tetap rahasia, namun dapat diakses oleh yang berhak di mana saja dan kapan saja.

Mekanisme yang dilakukan adalah kita menyimpan data dan informasi dalam bentuk yang **terenkripsi** dalam server dan menyediakan mekanisme pencarian melalui data yang terenkripsi tersebut. Mungkin jika kita membayangkan cara yang termudah adalah dengan mendekripsi masing-masing informasi kemudian mencarinya dengan algoritma pencarian umum terhadap data yang telah terdekripsi. Jika ditemukan, maka system akan mengembalikan informasi yang terenkripsi ke pengguna. Namun, kembali ke pernyataan penulis sebelumnya, bahwa data tersimpan di ruang public. Melakukan dekripsi di ruang public merupakan suatu hal yang riskan dari segi keamanan. Informasi hasil dekripsi akan mudah disadap. Ibarat analogi penulis sebelumnya, kita bias membuka peti satu per satu, namun besar

kemungkinan saat mengecek isi peti tersebut, orang lain melihat isi peti tersebut dan menyebabkan kerahasiaan informasi menjadi bocor. Ingat, bahwa dalam dunia internet, server **tidak dapat dipercaya**.

II. MEKANISME UMUM DENGAN PENDEKATAN BLOK CIPHER

Dalam bab ini akan dipaparkan salah satu pendekatan umum yang digunakan dalam teknik pencarian data terenkripsi. Pendekatan yang digunakan ini berdasarkan penemuan Dawn Song, dkk dan dari jurnal IEEE di bidang Security and Privacy pada tahun 2000. Tujuan dari pendekatan ini adalah (1) mekanisme yang ditawarkan harus terbukti aman, (2) Pencarian memiliki kompleksitas waktu O(N), di mana N adalah jumlah total dokumen (pencarian linier), dan (3) mendukung *controlled searching* (server tidak dapat mencari kata yang tidak disediakan oleh pengguna), *hidden queries* (kata yang dicari tidak diketahui oleh server), dan *query isolation* (server hanya mengetahui hasil pencarian).

A. Terbukti Aman

Berbeda dengan *perfect secrecy* dalam teori informasi, system ini lebih memikirkan masalah sumber daya yang dibutuhkan oleh penyerang dalam memecahkan kriptosistem, atau dengan kata lain keamanan secara komputasi. Jadi, kita tidak memakai pendekatan teori informasi dalam membuktikan keamanan dari system ini, tetapi kita lebih menekankan pada jumlah sumber daya yang dibutuhkan untuk memecahkannya. Kita ingin menghitung tingkat keuntungan dari penyerang. Secara formal kita akan menggunakan notasi berikut

- A: $\{0,1\}^n$ _ $\{0,1\}$ di mana A adalah algoritma
- X dan Y adalah variable acak yang terdistribusi dalam $\{0,1\}^n$
- Adv A = |Prob[A(X)=1]-Prob[A(Y)=1] |

B. Primitif

Dalam implementasinya, kriptosistem ini tetap menggunakan primitive-primitif yang lumrah digunakan dalam algoritma enkripsi pada umumnya. Berikut akan dipapakan primitive-primitif apa saja yang dibutuhkan dalam kriptosistem ini.

- Pembangkit pseudorandom G (misalkan *stream cipher*)
- Fungsi pseudorandom F (misalkan fungsi hash)
- Permutasi pseudorandom E (misalkan *block cipher*)

C. Inisiasi

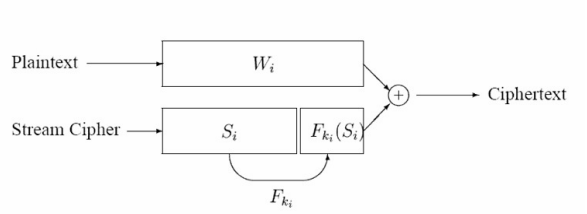
Misalkan Alice ingin mengenkripsi sebuah dokumen yang memiliki rangkaian kata-kata W_1, …, W_l. Untuk memudahkan kriptosistem, kita asumsikan semua kata memiliki panjang yang sama. Panjang dari kata ini, misalkan n, ditetapkan di awal oleh pengguna. Dalam kasus kata yang pendek, digunakan *padding*. Sedangkan dalam kasus kata yang panjang, kata tersebut dibagi menjadi kata yang lebih kecil. Penggunaan panjang yang sama memudahkan system, terutama dalam memiripkan system ini dengan system *block cipher*.

Mengutip dari poin B di atas, Alice sudah memiliki primitive-primitif yang dibutuhkan. Pertama, $G:K_G\_X^l$ adalah pembangkit pseudorandom pada beberapa l dan $X=\{0,1\}^{n-m}$. Kedua, $F:K_F x X\_Y$ adalah fungsi pseudorandom pada $X=\{0,1\}^{n-m}$ dan $Y=\{0,1\}^m$. $E:K_E x Z\_Z$ adalah permutasi pseudorandom pada $Z=X x Y=\{0,1\}^n$.

D. Skema Dasar

Langkah-langkah pada tahap ini sebagai berikut. Alice membangkitkan serangkai nilai pseudorandom S_1, …, S_l dengan menggunakan G di mana setiap S_i memiliki panjang n-m bits. Kemudian dia menghitung $T_i=<S_i,F_{ki}(S_i)>$. Setelah itu dihasilkan cipherteks $C_i=W_iT_i$.

Kunci k_i dapat sama atau berbeda untuk setiap kata. Jika F dan G terbukti aman, maka rangkaian T_i adalah pembangkit pseudorandom yang aman. Berikut adalah gambar dari skema yang dimaksud



Jika Alice ingin mencari kata W, dia dapat memberitahukan Bob (server) kata W dan k_i yang bersesuaian dengan masing-masing lokasi l sehingga W mungkin muncul. Kemudian Bob mencari dokumen dalam cipherteks dengan mengecek apakah C_iW_i merupakan bentuk dari $<s,F_{ki}(s)>$ untuk beberapa s.

Salah satu property dari skema dasar ini adalah adanya *limited controlled search*. Dengan kata lain, Bob hanya dapat mencari di beberapa bagian dari teks (sesuai dengan yang dipersilahkan oleh Alice). Hal ini menyebabkan Bob hanya memiliki akses terbatas pada teks yang diproses tersebut.

Masalah yang timbul adalah Alice seharusnya mengetahui posisi di mana W akan muncul atau jika tidak maka akan berakhir pada semua kunci k_i diketahui Bob.

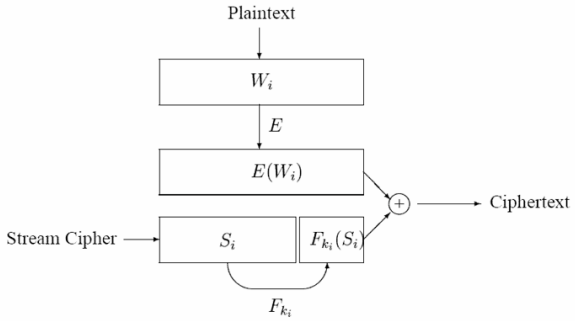
E. Skema II: Controlled Searching

Untuk mengatasi masalah di atas, pembuat skema ini menawarkan sebuah *improvements*. Caranya adalah sebagai berikut. Alice menggunakan fungsi pseudorandom tambahan $f:K_F x\{0,1\}^*\_K_F$ yang dikunci dengan k' dan independent terhadap F. Sekarang untuk membangkitkan k_i dia menggunakan $k_i=f_{k'}(W_i)$. Jika Alice ingin Bob mencarikan W, dia memberitahukan W dan $f_{k'}(W)$ dan Bob tidak akan mengetahui lokasi i di mana W_i!=W. Namun, pembangkitan k_i masih fleksibel dan masalah utama tetap muncul karena Alice masih memberi tahukan W terhadap Bob.

F. Skema III: Mendukung Pencarian Tersembunyi

Improvements lebih lanjut yang ditawarkan oleh pembuat system adalah sebagai berikut. Alice ingin mencari untuk sebuah kata W, tetapi tidak siap untuk

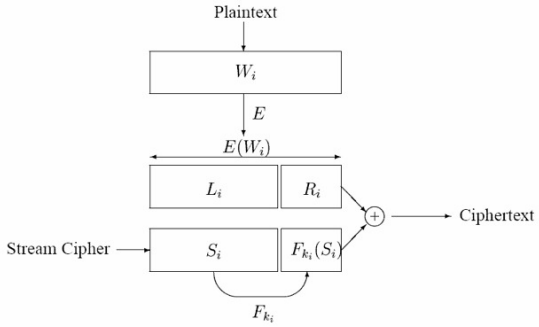
memberitahukannya ke Bob. Maka, dia seharusnya melakukan preenkripsi setiap kata W dari teks dengan menggunakan algoritma enkripsi yang telah ditentukan (misalnya enkripsi ECB dari kata tersebut) sehingga $X_i=E_{k'}(W_i)$. Kemudian kita bangkitkan T_i sesuai dengan $T_i=<S_i,F_{ki}(S_i)>$. Lalu, keluaran cipherteks yang diharapkan sesuai dengan $C_i=X_iT_i$. Semua property yang disebutkan sebelumnya telah terpenuhi dalam skema ini. Gambar dari skema yang dimaksud adalah sebagai berikut.



F. Skema IV: Skema Final

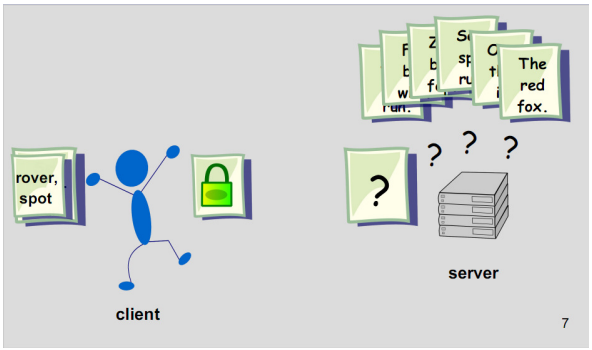
Masalah yang muncul dari skema III adalah sebagai berikut. Jika diberikan hanya cipherteks, Alice tidak dapat mendekripsi kata-kata yang dimaksud. Untuk memperjelas pernyataan di atas, kita asumsikan $C_i=X_iT_i$ dan $k_i=f_{k'}(X_i)$, dia harus mengetahui X_i sebelum menghitung T_i dan mendekripsi C_i.

Solusi yang ditawarkan oleh pembuat system adalah dengan cara membagi X_i ke dalam $<L_i,R_i>$, di mana L_i memiliki panjang n-m bits dan Ri memiliki panjang m bits. Sekarang kita gunakan $k_i=f_{k'}(L_i)$. Untuk mendekripsi *random entry*, Alice menemukan $L_i=C_iS_i$ (n-m bits pertama) sehinnga dapat membangkitkan k_i dan pada akhirnya dia dapat mengembalikan R_i. Gambar dari skema yang dimaksud adalah sebagai berikut.



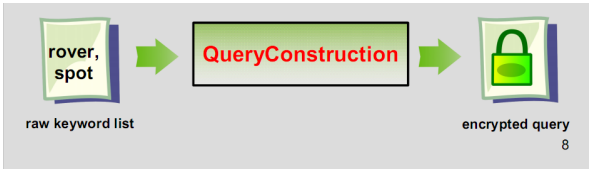
Dari skema final tersebut disimpulkan bahwa kriptosistem ini mudah untuk menambahkan entri baru. Kemudian, kunci master yang dibutuhkan sangat kecil sehingga overhead yang timbul dalam memanage kunci sangat kecil. Jika *wildcards* digunakan, maka semua kemungkinan kombinasi harus dibangkitkan dan ditanya. Lalu, dengan memiliki kata yang panjangnya tidak tetap, memungkinkan penyerangan secara statistical oleh server. Sebuah tambahan kata kunci dapat dienkripsi dan disimpan untuk meningkatkan efisiensi dari pencarian.

III. MEKANISME UMUM DENGAN PENDEKATAN STREAM CIPHER

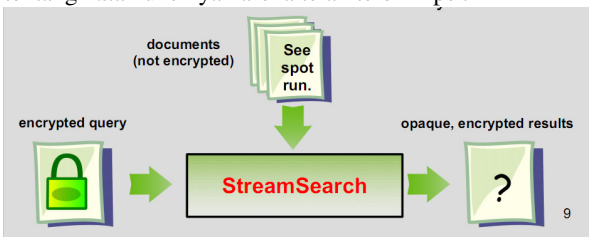


A. Arsitektur Dasar

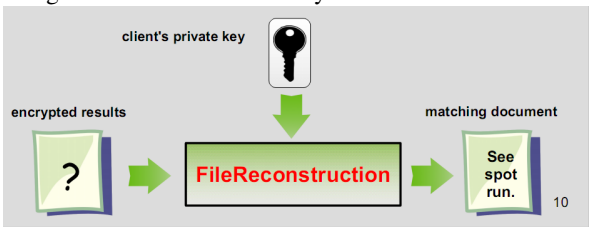
Langkah pertama yang ditawarkan oleh pembuat skema adalah tahap *query construction*. Dalam tahap ini klien menjalankan algoritma *QueryConstruction* untuk menghasilkan query yang terenkripsi. Query terpisah dari kata kunci tekstual atau dapat juga bervariasi menggunakan kata sambung lainnya (AND, OR, NOT). Kemudian klien mengirimkan query yang terenkripsi tadi ke server.



Langkah kedua yang ditawarkan oleh pembuat skema adalah tahap eksekusi pencarian. Dalam tahap ini, server mendapatkan query yang telah terenkripsi. Kemudian server menjalankan algoritma *StreamSearch*. Algoritma ini memproses dokumen-dokumen untuk menghasilkan buffer yang terenkripsi. Server tetap tidak mengetahui tentang kata kuncinya karena telah terenkripsi.



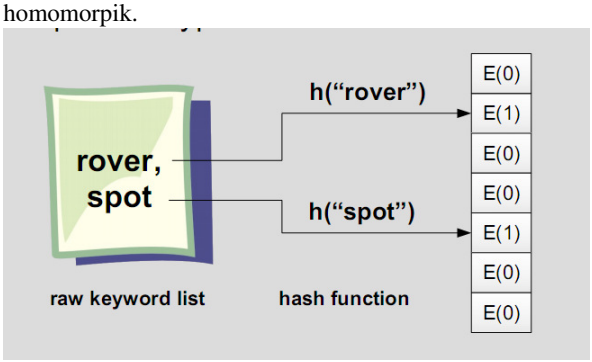
Langkah ketiga yang ditawarkan oleh pembuat skema adalah tahap pengambilan dokumen. Dalam tahap ini, klien mendapatkan hasil yang terenkripsi. Sistem menjalankan algoritma *FileReconstruction* dengan menggunakan kunci privat dari klien untuk mengembalikan dokumen aslinya.



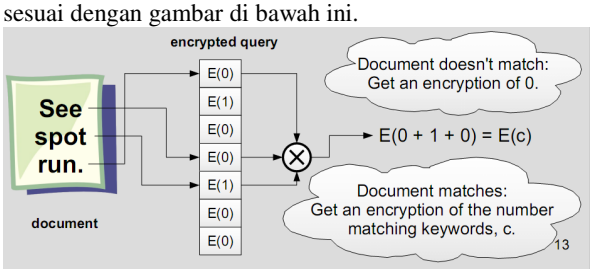
B. Highlights pada Skema

Untuk menjalankan algoritma-algoritma tersebut, terlebih dahulu harus dipecahkan masalah-masalah berikut. Pertama, masalah enkripsi homomorpik. Enkripsi homomorpik merupakan system berbasis kunci public. Sistem ini memenuhi persamaan berikut E(a).E(b)=E(a+b). Sistem ini menyebabkan adanya kemungkinan untuk melakukan komputasi pada data terenkripsi. Mengenai enkripsi homomorpik akan dijelaskan lebih lanjut.

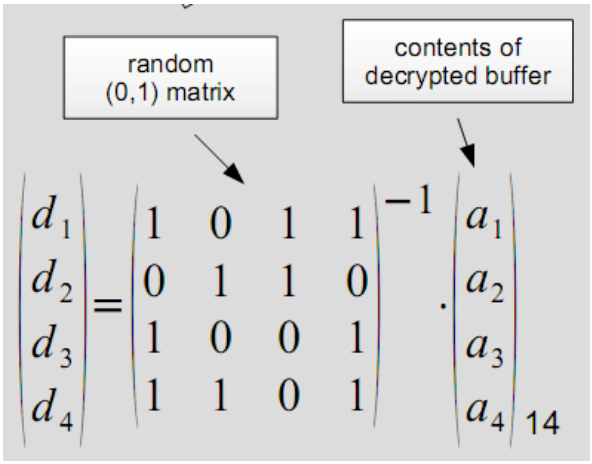
Kedua, masalah *encrypted queries*. Query dienkripsi dengan menggunakan table hash. Setiap cell adalah enkripsi dari 0 atau 1. Enkripsi yang dimaksud dapat menggunakan enkripsi probabilistic ataupun enkripsi homomorpik.



Ketiga, masalah pencarian. Server dapat menemukan hasil enkripsi dari jumlah kata kunci yang cocok. Hal ini sesuai dengan gambar di bawah ini.



Keempat, masalah pengambilan dokumen. Klien mendekripsi data yang dikembalikan dengan kunci privat. Klien juga dapat membangun system linear dalam pencocokan dokumen. Untuk memecahkan system linear dalam pencocokan dokumen digunakan pendekatan sebagai berikut



REFERENCES

[1] Dawn Song, *Practical Techniques for Searches on Encrypted Data*
[2] Robert Brinkman. *Efficient Tree Search in Encrypted Data*
[3] Craig Gentry. *Fully Homomorphic Encryption Over The Integers*
[4] John Bethencourt. *New Construction and Practical Applications for Private Stream Searching*
[5] Bruce Schneier. *Applied Cryptography*.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 17 Mei 2010

Haris Amrullah Lubis 13507038