MANAJEMEN KUNCI UNTUK BASISDATA TERENKRIPSI

Dini Armyta – NIM: 13503025

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail: if13025@students.if.itb.ac.id

Abstrak

Outsourcing basisdata saat ini telah menjadi suatu hal yang populer dan melahirkan suatu paradigma baru bernama database-as-a-service (DAS). DAS adalah organisasi basisdata yang disimpan pada penyedia layanan eksternal. Pada kasus ini, pengawasan pengaksesan data menjadi masalah yang penting, terutama bila pemilik data ingin mempublikasikan datanya untuk keperluan eksternal.

Maslah utama yang muncul dalam kegiatan *outsourcing* data pada penyedia layanan eksternal adalah data yang sifatnya penting disimpan pada situs yang tidak memiliki pengawasan langsung dari pemilik data yang bersangkutan. Karena itu kerahasiaan dan integritas data menjadi beresiko. Salah satu cara menanggulangi resiko tersebut adalah dengan melakukan enkripsi terhadap basisdata. Dengan mengenkripsi basisdata tersebut, pemilik data dapat yakin bahwa tidak ada siappun kecuali yang berwenang yang dapat membaca data tersebut. Bagaimanapun, solusi basisdata yang menyimpan informasi terenkripsi tidak sepenuhnya berhasil karena menyebabkan layanan eksternal tidak dapat mendukung pengaksesan yang selektif. Karena faktor kerahasiaan dibutuhkan dengan mampunya dilakukan dekripsi data hanya pada sisi *client*, teknik tertentu dibutuhkan untuk memungkinkan layanan eksternal dapat mengeksekusi query pada data terenkripsi, atau seluruh relasi yang telibat pada suatu query harus dikirimkan ke sisi *client* untuk eksekusinya.

Makalah ini kemudian akan membahas mengenai manajemen kunci sebagai suatu implementasi atas pengawasan pengaksesan terhadap basisdata terenkripsi.

Kata kunci: *key management, encrypted database*

1. Pendahuluan

Internet telah membuat sejumlah besar situs *server* yang mengirimkan informasi kepada individu-individu di seluruh dunia dengan kecepatan yang sangat tinggi. Dengan internet, organisasi dapat mengembangkan praktek bisnis yang efektif, yang dibangun sumber daya informasi yang tersedia dimana-mana. Aplikasi seperti e-Banking atau e-Commerceadalah contoh utama dari sumber daya *online real-time* yang dapat menghasilkan servis dengan nilai tambah melalui tersedianya basisdata dalam volume besar. Dengan pengembangan dari sistem berbasiskan pada *server*, kebutuhan akan keamanan terhadap penyimpanan data menjadi hal yang sangat penting dibandingkan sebelumnya. Dampak lain dari pertumbuhan sistem informasi *online* yaitu munculnya ancaman baru yang berkembang atas sistem tersebut: ancaman atas akses ilegal atau akses tanpa pengawasan terhadap sumber daya yang berharga tersebut. Di atas ancaman ini, terdapat pula pemikiran yang berkembang bahwa aset utama dari organisasi ini dapat dimodifikasi, dikompromisasi, atau bahkan dihapus yang dapat menyebabkan organisasi mengalami kerugian atau bahkan kebangkrutan.

Sebagai hasil dari ancaman tersebut, maka organisasi:

- perlu untuk menggunakan pengawasan tajam atas akses terhadap sumber daya informasi
- perlu untuk menjaga data terhadap perubahan, penambahan, atau bahkan penghapusan dari pihak yang tidak berwenang
- perlu untuk menyempurnakan pengawasan akses dan proteksi data hingga ke tingkat individual atas informasi pada level granular seperti kolom, *record*, atau *field* untuk menjamin penggunaan yang optimal atas sumber daya basisdata. Penjagaan hanya dilakukan kepada data yang perlu untuk diamankan dan tinggalkan

data yang tidak penting untuk tetap tidak dienkripsi.

Selagi sistem basisdata menawarkan sejumlah servis keamanan seperti manajemen konten, pengawasan akses, atau enkripsi basisdata, keberadaan servis-servis tersebut memberi dampak biaya karena fokus pada sistem basisdata adalah pada "pengiriman dari servis" dan "organisasi data", maka penyediaan keamanan yang berlebihan pada level kolom menjadi lebih intensif dan meningkat yang menyebabkan tabrakan atas kemampuan basisdata. Karena penggunaan yang intensif atas sumberdaya komputer untuk menyediakan keamanan atas servis, kemampuan basisdata secara keseluruhan menjadi sangat menurun karena penambahan keamanan yang berlebihan. Sistem basisdata pada umumnya menawarkan pendekatan enkripsi "lakukan enkripsi terhadap seluruh basisdata atau tidak sama sekali". Sehingga keamanan yang disediakan kepada basisdata tidak dapat tidak menyebabkan dampak yang radikal pada keseluruhan level dari kemampuan basisdata dan peningkatan dari penanaman sumberdaya. Pergolakan yang akan selalu ada yaitu *issue* antara kemampuan dan keamanan yang menyerang sistem basisdata dalam dunia Internet. Solusi apakah yang tepat untuk masalah ini?

Seperti telah dijelaskan sebelumnya, saat ini basisdata telah memegang peranan yang sangat penting atas informasi yang sifatnya sensitif dan jumlah dari informasi ini meningkat dengan sangat drastis. Selain itu, banyak organisasi yang menambahkan penyimpanan data dalam tingkat yang sangat tinggi. Ledakan jumlah data ini berdampak pada dituntutnya kekuatan yang lebih dari aplikasi-aplikasi basisdata, yang dapat dikembangkan oleh organisasi untuk mengelola dan mengatur informasi yang dimilikinya. Dalam skenario tersebutlah *outsourcing* basisdata menjadi sangat populer. Basisdata *client* yang disimpan pada penyedia layanan eksternal harus memiliki suatu mekanisme bagi *client* untuk dapat mengakses basisdata tersebut.

Keuntungan utama dari *outsourcing* terkait dengan biaya antara penggunaan host *in-house* dan *outsourced*: *outsourcing* menyediakan:

i) penghematan biaya yang cukup signifikan dan layanan yang menguntungkan.
ii) menjanjikan ketersediaan yang lebih tinggi dan proteksi yang lebih efektif atas kerusakan yang mungkin terjadi dibandingkan operasi dengan menggunakan *in-house*.

Dan sebagai konsekuensi atas meningkatnya penggunaan *outsourcing* ini, data-data yang sifatnya sangat sensitif saat ini juga disimpan pada sistem yang berjalan pada lokasi yang tidak berada di bawah pengawasan pemiliknya. Karena itu, kerahasiaan suatu data bahkan integritasnya dapat menjadi beresiko. Masalah-masalah ini secara tradisional akan dipecahkan dengan menggunakan enkripsi. Dengan melakukan enkripsi terhadap informasi rahasia tersebut, *client* mendapatkan jaminan bahwa hanya dirinya yang dapat mengobservasi data tersebut. Masalah yang muncul kemudian yaitu bagaimana cara untuk melakukan temu balik secara selektif pada informasi yang terenkripsi. Karena kerahasiaan data membutuhkan syarat bahwa dekripsi data hanya boleh dilakukan pada sisi *client*, teknik-teknik tertentu diperlukan untuk memungkinkan *server* eksternal dapat mengeksekusi query pada data terenkripsi, atau bagaimana cara agar seluruh relasi yang berhubungan dengan query tersebut dapat dikirimkan ke sisi *client* untuk melakukan eksekusi query.

Salah satu solusi yang pernah diajukan yaitu dengan dilakukannya penyimpanan informasi indeks tambahan beserta dengan basisdata terenkripsi tersebut. Indeks tersebut kemudian dapat digunakan oleh DBMS untuk memilih data yang akan menjadi hasil dari eksekusi atas suatu query. Selain itu, ada pula solusi untuk melakukan metode berbasis hash untuk enkripsi basisdata tertentu untuk query pemilihan. Untuk mengeksekusi query berbasis interval, struktur pohon-B+ yang biasa digunakan pada DBMS kemudian diadaptasi. Homomorfisme privat juga diajukan untuk memungkinkan eksekusi atas query agregasi atas data terenkripsi. Pada kasus ini, *server* menyimpan tabel terenkripsi dengan sebuah indeks untuk setiap atribut agregasi (yaitu atribut yang dikenai operator agregasi) yang diperoleh dari atribut asli dengan homomorfisme privat. Sebuah operasi pada atribut agregasi lalu dapat dievaluasi dengan melakukan perhitungan agregasi pada situs *server* dan dengan mendekripsi hasilnya pada sisi *client*. Kerja lain dari homomorfisme privat dapat diilustrasikan melalui teknik dalam melakukan operasi aritmatik (+, -, x, /) pada data terenkripsi dan tidak memperhitungkan operasi perbandingan. *Order Preserving Encryption Schema* (OPES) diajukan untuk mendukung kesamaan dan jangka query terhadap data terenkripsi. Pendekatan ini hanya mengoperasikan nilai integer dan hasil dari suatu query pada atribut terenkripsi

menggunakan OPES adalah lengkap dan tidak mengandung tuple palsu.

Bagaimanapun, pada skenario ini, yang disebut dengan *database-as-a-service* (DAS) mengajukan tantangan tambahan baru yang lebih berbasiskan pada kegunaan dari sistem itu sendiri. Salah satu tantangannya yaitu untuk mengembangkan teknik pengawasan akses yang efisien. Hal tersebut dikarenakan seluruh teknik yang diajukan untuk merancang dan melakukan query terhadap data *outsourced* terenkipsi/terindeks mengasumsi-kan bahwa sisi *client* memiliki akses penuh terhadap hasil query. Asumsi ini tidak dapat digunakan pada dunia nyata karena pengguna-pengguna yang berbeda mungkin memiliki hak akses yang berbeda pula. Sebagai contoh, pada basisdata kesehatan yang mengandung informasi mengenai dokter dan pasien. Setiap pengguna (dokter atau pasien) atau sekelompok pengguna seharusnya memiliki hak akses yang berbeda untuk sejumlah data spesifik yang ada. Memaksakan pengaksesan selektif dengan definisi eksplisit dari autorisasi membutuhkan pemilik data untuk memotong dan memproses setiap permintaan query (dari pengguna kepada *server*) dan setiap jawaban (dari *server* kepada pengguna) untuk menyaring data yang sesuai dengan hak akses autorisasi pengguna yang bersangkutan. Pendekatan ini bagaimanapun akan menyebabkan *bottleneck* karena meningkatkan proses dan komunikasi pada situs pemilik data.

Salah satu langkah yang menjanjikan untuk menghindari masalah *bottleneck* tersebut adalah dengan melakukan melakukan enkripsi data secara selektif sehingga seorang pengguna (atau sekelompok pengguna) dapat mendekripsi data yang hanya berhak mereka akses. Solusi ini membutuhkan definisi dan perawatan yang berkesinambungan, baik pada sisi *client* maupun pada sisi *server*, informasi tambahan pada level dari metadata juga dibutuhkan untuk memaksakan akses selektif. Makalah ini akan memfokuskan pada metadata yang dibutuhkan untuk mengakses basisdata *outsourced* berdasarkan pada ketentuan-ketentuan yang didefinisikan oleh pemilik data. Secara lebih jelasnya, akan dideskripsikan mengenai metadata dan perbandingan stategi penyimpanan yang berbeda yang masing-masing dikarakteristikkan dengan penggunaan penyimpanan dan kapasitas *bandwith* yang berbeda. Selain itu, makalah ini juga akan membahas mengenai skenario DAS dan secara singkat mengilustrasikan solusi untuk memungkinkan adanya pengawasan akses melalui kriptografi. Akan dibahas pula mengenai jenis-jenis strategi yang berbeda untuk menyimpan dan mengatur metadata yang dibutuhkan untuk kebutuhan dalam basisdata *outsourced*. Setelah itu, akan dilakukan pembahasan yang mengilustrasikan bagaimana suatu query pada basisdata *plaintext* ditransformasi menjadi query pada basisdata terenkripsi bersangkutan.

2. Manajemen Kunci

Sebelum membahas lebih lanjut mengenai metode manajemen kunci yang digunakan pada basisdata terenkripsi, terlebih dulu akan dibahas mengenai pengertian dari manajemen kunci itu sendiri. Manajemen kunci berhubungan dengan keamanan, distribusi, dan penyimpanan dari kunci. Metode keamanan dari manajemen kunci adalah sangat penting. Bahwa sekali kunci secara acak dibangkitkan, maka nilai dari kunci tersebut harus menjadi rahasia untuk mencegah adanya kecelakaan yang tidak diinginkan. Dalam prakteknya, kebanyakan serangan yang terjadi pada sistem kunci publik dapat ditangani dalam level manajemen kunci, daripada oleh algoritma kriptografi itu sendiri.

Pengguna harus mampu untuk menjaga keamanan dari pasangan kunci yang dimilikinya untuk kebutuhan efisiensi dan keamanannya. Bagaimanapun, akan selalu ada cara untuk melihat kunci publik orang lain atau bahkan mempublikasikannya ke khalayak umum. Karena itu pengguna harus mampu untuk melegitimasi kunci publik orang lain, atau seorang pengacau dapat saja mengubah daftar kunci publik yang ada. sertifikat dibutuhkan untuk menjaga keamanan dari kunci tersebut. Sertifikat tersebut sendiri sebaiknya tidak dapat ditiru atau dipalsukan. Masalah dari sertifikat ini sendiri harus ditangani dengan keamanan yang baik, dan agar tidak dapat diserang. Pengguna harus mengautentikasikan identitas dan kunci publik miliknya masing-masing sebelum memperoleh sertifikat individual tersebut. Jika kunci privat seseorang hilang, maka orang lain harus waspada atas hal ini, sehingga mereka tidak lagi mengenkrip pesan di bawah kunci publik yang tidak valid ataupun menerima pesan yang ditandatangani dengan kunci privat yang tidak valid tadi. Pengguna harus mampu untuk menyimpan kunci privatnya secara aman, sehingga tidak ada pengacau yang dapat mengaksesnya, namun kunci tersebut juga harus dapat diakses oleh pihak yang memang berwenang untuk mengetahuinya. Kunci harus terus valid hingga tanggal kadaluarsanya tiba namun tanggal kadaluarsa tersebut juga harus

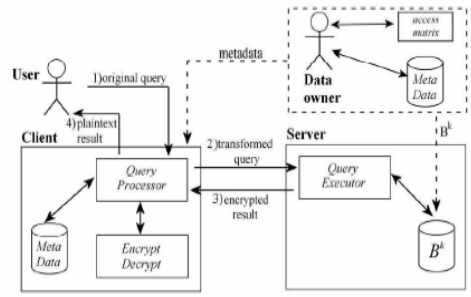
ditentukan dengan tepat dan mempublikasikannya pada saluran yang dapat dijamin kebenarannya.

3. Skenario DAS

Berikut adalah penjelasan singkat mengenai skenario pada DAS, struktur basisdata terenkripsi, dan solusi untuk memungkinkan ketentuan pengawasan akses pada analisa basisdata yang digunakan.

3.1. Organisasi Data

Skenario DAS melibatkan 4 entitas (lihat Gambar 1):



Gambar 1. Skenario DAS

- Pemilik data: organisasi yang menyediakan data agar dapat digunakan untuk oleh pihak eksternal.
- Pengguna: entitas manusia yang menyediakan permintaan (query) kepada sistem.
- *Client*: *front-end* yang mentransformasi query pengguna menjadi query pada penyimpanan data terenkripsi pada *server*.
- *Server*: organisasi yang menerima data terenkripsi dari pemilik data dan membuatnya tersedia untuk distribusi kepada *client*.

Client dan pemilik data diasumsikan mempercayai *server* sepenuhnya untuk mengelola data *outsourced*. Secara spesifik, *server* bertumpu pada ketersediaan basisdata *outsourced*. Bagaimanapun, *server* juga diasumsikan untuk tidak dipercaya atas kerahasiaan dari isi basisdata yang sesungguhnya. Karena itu, perlu dilakukan perlindungan kepada *server* dari membuat akses yang tidak berwenang pada data yang tersimpan pada basisdata.

Patients

PatientId	Surname	Name	Disease	Doctor
125YP894	Carter	Andrew	Tonsillitis	Wayne
5896GT26	Rogers	Mark	Gastritis	Becker
654ED231	Wise	Paul	Hypertension	Wayne
442HN718	Brown	Luke	Hypertension	Lean
942MD745	Fisher	Robert	Tonsillitis	Becker
627IF416	Rogers	Alice	Arthritis	Wayne
058PI175	Brown	Mark	Hypertension	Lean
305EJ186	Rogers	Paul	Gastritis	Morris
276DL557	Fisher	Luke	Hypertension	Lean
364UK784	Rogers	Laura	Tonsillitis	Wayne

Gambar 2. Contoh relasi pasien yang belum dienkripsi

$Patients^k$

Counter	Etuple	IdKey	I_1	I_2	I_3	I_4	I_5
1	r*tso/yui+	AC	ω	γ	δ	π	η
2	hai4de-0q1	AB	ϑ	α	λ	π	μ
3	nag+q8*L	C	ω	γ	ϵ	ρ	η
4	K/ehim*13-	BCD	τ	β	δ	ρ	μ
5	3gia*ni+aL	BD	ω	β	λ	π	μ
6	F0/rab1DW*	BCD	ϑ	α	ϵ	ρ	η
7	Bid2*k1-l0	AB	ϑ	β	λ	ρ	μ
8	/bur21/*-D	BC	τ	α	ϵ	π	η
9	O/c*yd-q2+	C	ω	β	δ	ρ	μ
10	bew0"!DE1a	ACD	ϑ	α	λ	π	η

Gambar 3. Contoh relasi pasien yang telah dienkripsi

Untuk melakukan perlindungan tersebut, pemilik data harus mengenkripsi datanya dan memberikan basisdata terenkripsi tersebut kepada *server*. Sebagai catatan bahwa enkripsi basisdata dapat dilakukan pada lever granularitas yang berbeda-baik, baik pada level relasi, level atribut, level tuple, atau bahkan pada level elemen. Kedua level relasi dan level atribut berdampak pada komunikasi kepada pengguna dari seluruh relasi yang berhubungan dengan query tersebut. Dan pada sisi lain, mengenkripsi pada level elemen akan membutuhkan kerja yang lebih besar untuk pemilik data dan *client* dalam mengenkripsi atau mendekripsi data. Untuk menyeimbangkan kerja *client* dan efisiensi eksekusi query, diasumsikan bahwa basisdata dienkripsi pada level tuple.

Usaha utama pada skenario ini yaitu untuk merancang suatu mekanisme yang memungkinkan melakukan query langsung kepada basisdata terenkripsi. Metode yang telah ada sebelumnya yaitu dengan menggunakan informasi indeks yang berasosiasi dengan setiap relasi pada basisdata terenkripsi. Indeks-indeks tersebut dapat digunakan oleh *server* untuk memilih data yang kemudian akan dikembalikan sebagai respon atas suatu query. Secara lebih terperinci, *server* akan menyimpan sebuah tabel terenkripsi dengan sebuah indeks untuk setiap atribut dimana suatu query dapat mengandung suatu kondisi. Tipe indeks yang berbeda dapat didefinisikan tergantung pada query yang digunakan. Sebagai contoh, metode

berbasis hash cocok digunakan untuk query yang mengandung persamaan, dan metode berbasis pohon-B+ cocok digunakan untuk query yang mendukung *range*. Pada makalah ini diasumsikan indeks telah dibentuk melalui metode berbasis hash dan bahwa terdapat sebuah indeks untuk setiap atribut pada setiap relasi. Secara formal, setiap relasi ri pada skema:

Ri(Ai1, Ai2, ..., Ain)
pada basisdata *plainteks* B kemudian dipetakan pada relasi rki pada skema:

Ri^k*(Counter, Etuple, I1, I2, ..., In)*

pada basisdata terenkripsi B^k dimana:

- Counter: *primary key*
- Etuple: atribut dari tuple terenkripsi yang nilainya diperoleh menggunakan fungsi enkripsi Ek (k adalah kunci)
- Ii: indeks yang berasosiasi dengan atribut ke-i

Sebagai contoh, relasi Patients pada Gambar 2, relasi terenkripsi ditampilkan pada Gambar 3. Seperti dapat dilihat bahwa tabel terenkripsi memiliki jumlah baris yang sama dengan tabel yang asli. Pemrosesan query kemudian akan dilakukan dengan langkah sebagai berikut (seperti pada Gambar 1):

1. Setiap query dipetakan kepada suatu query pada data terenkripsi.
2. Dikirimkan kepada *server* yang memiliki tanggung jawab untuk mengeksekusinya.
3. Hasil dari query tersebut adalah sebuah set dari tuple terenkripsi kemudian diproses oleh *client front-end* untuk mendekrip data dan mengabaikan tuple palsu yang mungkin menjadi bagian dari hasil.
4. Hasil akhir kemudian ditampilkan kepada pengguna.

3.2. Akses Selektif pada Basisdata Terenkripsi

Seluruh metode yang telah diajukan untuk merancang dan melakukan query terenkripsi/terindeks kepada basisdata terenkripsi difokuskan pada tantangan untuk melindungi data pada sisi *server*, dan mengasumsikan bahwa *client* memiliki akses penuh kepada hasil query. Dengan kata lain, tuple diasumsikan dienkripsi menggunakan sebuah kunci tunggal; kunci ini memungkinkan akses komplit kepada seluruh basisdata. Asumsi seperti ini tidak dapat diterapkan pada aplikasi di dunia nyata yang membutuhkan akses selektif untuk jenis pengguna ataupun aplikasi yang berbeda-beda.

Solusi baru yang diajukan yaitu dengan menekankan enkripsi data dengan melibatkan autorisasi pada basisdata terenkripsi itu sendiri. Jika pada prinsipnya dikatakan bahwa sebaiknya memisahkan antara pengawasan akses berbasiskan autorisasi dengan proteksi dengan kriptografi, namun pada skenario DAS kombinasi seperti ini dibuktikan dapat berhasil dengan sukses. Ide ini kemudian dikembangkan dengan menggunakan kunci enkripsi yang berbeda-beda untuk data yang berbeda pula. Untuk mengakses data terenkripsi seperti itu, pengguna harus mendekripnya yang hanya dapat dilakukan bila mengetahui algoritma enkripsi dan kunci dekripsi spesifik yang digunakan. Jika akses terhadap kunci dekripsi dibatasi untuk pengguna tertentu dari sistem, maka tiap pengguna dapat memiliki hak akses yang berbeda pula.

Dalam pengertian klasik, hak akses tiap pengguna yang berbeda tersebut dapat ditampilkan dalam bentuk suatu matriks pengaksesan A, dimana baris berkoresponden dengan subjek, kolom berkoresponden dengan obje, dan *field* A[s,o] diberi nilai 1 bila s dapat membaca o, dan bernilai 0 bila sebaliknya. Bila diberikan sebuah matriks pengaksesan A, ACLi memberikan vektor yang berkoresponden dengan kolom ke-i (yaitu daftar kontrol akses yang mengindikasi subjek yang dapat membaca tuple ti), dan CAPj menggambarkan vektor yang berkoresponden dengan baris ke-j (yaitu daftar kemampuan yang mengindikasi objek yang dapat dibaca oleh pengguna). Sebagai contoh pada situasi dengan 4 orang pengguna bernama Alice, Bob, Carol, dan David, yang butuh untuk membaca tuple dari relasi Patients. Gambar 4 mengilustrasikan contoh dari matriks pengaksesan kasus tersebut.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
Alice	1	1	0	0	0	0	1	0	0	1
Bob	0	1	0	1	1	1	1	1	0	0
Carol	1	0	1	1	0	1	0	1	1	1
David	0	0	0	1	1	1	0	0	0	1

Gambar 4. Contoh matriks pengaksesan

Pendekatan yang berbeda dapat diambil untuk memungkinkan hak akses dilaporkan pada matriks pengaksesan. Salah satu solusi yang ada yaitu dengan mengenkripsi setiap tuple dengan sebuah kunci yang berbeda-beda dan

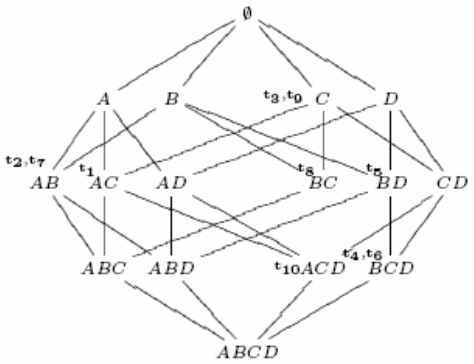
memberikan pengguna kunci untuk tuple yang dapat mereka akses. Sebagai contoh, dengan melihat pada matriks pengaksesan pada Gambar 4, maka pengguna dengan nama Carol menerima kunci-kunci untuk mengenkripsi tuple t1, t3, t4, t6, t8, t9, dan t10. Tentunya solusi ini sangat tidak efisien dan membutuhkan manajemen dari begitu banyak kunci.

Solusi lain yang diajukan yaitu dengan mengumpulkan pengguna menjadi beberapa grup dengan kewenangan yang sama dan mengenkripsi setiap tuple dengan kunci yang berasosiasi dengan set dari penguna yang dapat mengakses data-data tersebut. Untuk tujuan ini, pendekatan pada definisi dan penggunaan dari hirarki pengguna yang semua elemennya dapat merupakan set dari pengguna dan yang urutannya didefinisikan berkoresponden pada himpunan bagian dari diantaranya. Secara formal, hirarki pengguna didefinisikan sebagai berikut:

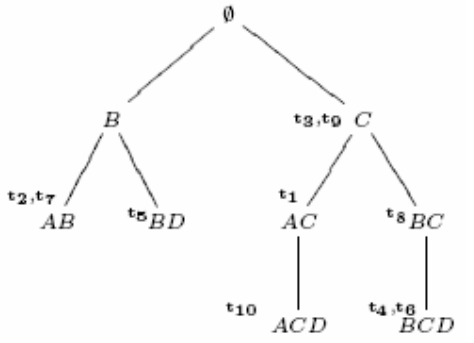
Definisi 1 (Hirarki pengguna) Diberikan sebuah set U yang berisi nama-nama pengguna, sebuah hirarki pengguna yang dinotasikan dengan UH, adalah pasangan (P(U),) dimana P(U) adalah *power set* dari U dan _ adalah urutan parsial pada P(U) sehingga $\forall X, Y \in P(U)$, X_Y jika dan hanya jika $Y \subseteq X$.

Himpunan bagian antara set dari pengguna memenuhi hubungan dengan hak-haknya. Maka bila diberikan 2 set pengguna X dan Y, jika Y adalah himpunan bagian dari X (yaitu X _ Y) maka pengguna di Y dapat mengakses seluruh tuple yang juga dapat diakses oleh pengguna X, namun tidak berlaku untuk sebaliknya. Dengan mengacu pada Gambar 3, BCD adalah set pengguna Bob, Carol, David, dan BC adalah set pengguna Bob dan Carol. Pengguna di BCD dapat mengakses tuple t4 dan t6 dan pengguna di BC dapat mengakses tuple t4, t6, dan t8. Sesuai dengan definisi, hirarki pengguna juga mengikutkan seluruh set pengguna yang berkoresponden dengan ACLi. Hirarki pengguna A dapat direpresentasikan melalui graf non-siklik (DAG) yang memiliki sebuah simpul berkoresponden dengan setiap set pengguna dan sebuah busur dari simpul X ke simpul Y jika dan hanya jika Y _ X. Gambar 5 menggambarkan hirarki pengguna yang berkoresponden dengan matriks pengaksesan pada Gambar 4. Disini setiap simpul diberi label dengan set dari huruf inisial nama pengguna yang memiliki simpul tersebut, dan setiap tuple ti digambarkan di dekat simpul ACLi. Solusi untuk determinasi dan pemilihan kunci membawa hirarki pengguna bersama dengan kunci dan pemilihan skema berdasarkan pada ide dari kunci asal.

Skema penentuan kunci dioperasikan pada hirarki yang menghitung kunci-kunci pada level bawahnya berdasarkan pada kunci-kunci dari *predecessors*-nya. Dengan kata lain, setiap simpul X dari hirarki berasosiasi dengan sebuah kunci yang dapat digunakan untuk



Gambar 5. Contoh hirarki pengguna



Gambar 6. Contoh pohon yang berkoresponden dengan gambar 5

memperoleh kunci yang berasosiasi dengan seluruh simpul Y, dimana Y _ X, namun tidak berlaku untuk sebaliknya. Maka, dalam penggunaan hirarki pengguna untuk pemilihan kunci, setiap pengguna hanya harus mengetahui kunci yang berasosiasi dengan simpul miliknya dan setiap tuple t harus dienkripsi dengan kunci yang berasosiasi dengan simpul yang merepresentasikan ACL-nya.

Sebagai contoh, tuple t1 dienkripsi dengan kunci yang berasosiasi dengan simpul C (lihat Gambar 5). Dalam hal ini, Carol dapat memperoleh kunci yang berasosiasi dengan simpul AC dan dapat mengakses tuple t1. Namun, skema pemerolehan kunci pada DAG sangat kompleks dan membutuhkan penyimpanan banyak kunci (yang ukurannya dapat meningkat secara eksponensial dengan

simpul yang ada pada hirarki). Untuk mencegah hal ini terjadi, maka diterapkan teknik yang lebih simpel untuk diterapkan kepada pohon tersebut. Dikembangkan sebuah algoritma transformasi *greedy* dari DAG menjadi pohon yang mengubah sebuah hirarki dalam pohon yang berkoresponden. Transformasi ini dilakukan dengan pertama-tama mengidentifikasi simpul-simpul pada pohon yang berkoresponden dengan ACLs, dan kemudian dengan memilih dari setiap simpul siapa "simpul orangtua terbaiknya". Pemilihan ini dilakukan dengan mengadopsi set dari kriteria yang dapat mereduksi jumlah kunci pada sistem.

Sebagai contoh, suatu ukuran membutuhkan pilihan kandidat orangtua terendah pada hirarki, yaitu simpul orangtua yang berkoresponden dengan set terbesar dari pengguna. Ukuran lainnya dinyatakan lebih baik untuk memilih sebuah orangtua yaitu simpul yang berkoresponden dengan sebuah ACL. Pada akhir proses transformasi, algoritma ini membuang simpul-simpul dalam struktur yang tidak dibutuhkan baik oleh proses enkripsi ataupun dalam proses pemerolehan kunci. Hasil dari hirarki pohon pengguna didefinisikan sebaga berikut:

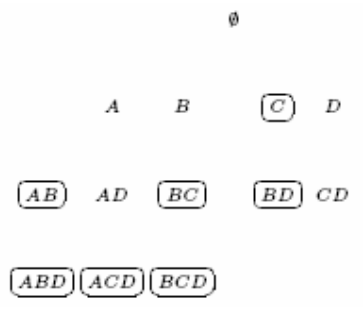
Definisi 2 (Hirarki Pohon Pengguna) Diberikan set U dari pengguna, set T dari tuple dan matriks pengaksesan A, hirarki pohon pengguna dinotasikan dengan A, yaitu pasangan (N, _) dengan

- $N \subseteq P(U)$;
- $\forall t \in T$, $ACLt \in N$;
- $\forall X, Y \in N$, $X \_ Y$ jika dan hanya jika $Y \subseteq X$;
- $\forall X, Y, Z \in N$, $X \_ Y$ dan $X \_ Z \Rightarrow Z \_ Y$ atau $Y \_ Z$.

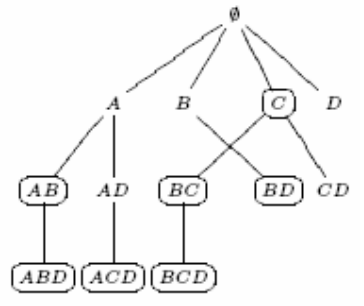
Hirarki pohon pengguna menggunakan urutan relasi parsial yang sama dengan yang didefinisikan oleh hirarki pengguna. Pemilik data harus mengkomunikasikan kepada setiap pengguna $u \in U$ atas kunci yang berasosiasi dengan elemen $V \in N$ sehingga $u \in V$ dan $u \notin W$, dimana W adalah orangtua dari V pada UTH. Sebagai catatan, hindari pengaksesan dari pengguna yang tidak dapat dipercaya, pemilik data harus mengecek identitas pengguna sebelum memberikan pengguna tersebut kunci. Gambar 6 menggambarkan UTH yang berkoresponden dengan hirarki pengguna pada Gambar 5. Sebagai contoh, pada pengguna dengan nama Carol, ia mengetahui kunci {kC} dan dapat langsung memperoleh {kAC, kBC} yang juga memungkinkannya untuk memperoleh kunci {kACD, kBCD}. Dengan menggunakan kunci tersebut Carol dapat mendekrip set tuple {t1, t3, t4, t6, t8, t9, t10} yang berkoresponden dengan CAPCarol.

4. DAG untuk Algoritma Transformasi Pohon

Algorima ini dibangun dengan tujuan untuk meminimalkan jumlah kunci yang akan dikomunikasikan kepada pengguna. Lebih jelasnya, ukuran yang ingin diminimalkan adalah rata-rata jumlah kunci yang harus diketahui oleh pengguna untuk dapat mengakses tuple yang merupakan haknya. Berikut adalah langkah yang harus dilakukan:



Gambar 7. Simpul material dan non-material yang berkoresponden dengan DAG pada Gambar 5



Gambar 8. Pohon yang berkoresponden dengan DAG pada Gambar 5 sebelum fase pemangkasan

Langkah 1: Identifikasi atas simpul-simpul kandidat

Pilih himpunan bagian dari simpul DAG yang dinamakan simpul *material* (M) dan *non material* (NM). Simpul material berkoresponden dengan ACLs dan harus merupakan bagian dari pohon karena kunci tersebut akan digunakan untuk mengenkripsi tuple yang bersesuaian. Sebagai contoh, dengan

melihat pada hirarki pengguna pada Gambar 5, set M dari simpul material adalah {C, AB, BC, BD, ABD, ACD, BCD}. Simpul non-material adalah simpul yang dapat digunakan untuk mereduksi jumlah dari kunci yang harus diberikan kepada pengguna. Dalam contoh ini, set NM dari simpul non-material adalah {A, B, D, AD, CD}. Simpul non-material dapat dikomputasi secara rekursif dengan memperhatikan *fix-point* dari komputasi orangtua dari 2 simpul material atau non-material. Sebagai catatan bahwa set final dari simpul material dan non-material ditutup dengan dilakukannya operasi *intersection*. Gambar 7 mengilustrasikan set M dan NM dari simpul material dan non-material yang berkoresponden dengan DAG pada Gambar 5. Untuk membedakan antara 2 jenis simpul ini, maka simpul material dibedakan dengan adanya lambang lingkaran.

Langkah 2: Identifikasi busur

Langkah kedua dari algoritma ini terdiri dari pemilihan orangtua untuk setiap simpul M dan NM dengan tujuan untuk membentuk struktur pohon secara lengkap. Langkah ini dilakukan dengan memperhatikan bahwa setiap simpul pada DAG muncul pada level tertentu dan setiap busur menghubungkan simpul-simpul dari level yang berdekatan. Sebuah level terdiri dari semua simpul yang berkoresponden dengan set dari pengguna yang memiliki kardinalitas yang sama. Yaitu, level 0 mengandung set kosong, level 1 terdiri dari simpul yang berkoresponden dengan pengguna tunggal, level 2 mengandung simpul yang berkoresponden dengan pasangan pengguna, dan seterusnya. Sebagai contoh, hirarki pengguna pada Gambar 5 memiliki 5 level (0 .. 4) dan level 3 terdiri dari simpul ABC, ABD, ACD, dan BCD. Pemilihan dari simpul orangtua dilakukan dari satu level ke level lainnya (dari kiri ke kanan), dimulai dari level tertinggi ke level 2 (dengan definisi orangtua dari simpul pada level 1 adalah set kosong). Pemilihan dari simpul orangtua dapat dilakukan tergantung pada kriteria yang berbeda. Digunakan suatu metode untuk mengevaluasi kontribusi setiap kriteria apakah memberikan kualitas terhadap solusi yang dihasilkan, dibandingkan dengan peningkatan waktu eksekusi yang dibutuhkan untuk aplikasi. Kriteria tersebut yaitu sebagai berikut:

C1. Pilih secara acak sebuah kandidat simpul orangtua yang berlokasi pada level terbawah di atas simpul yang memberikan leluhur material atau non-material dari simpul tersebut.

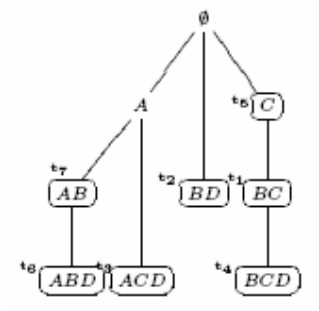
C2. Pilih sebuah kandidat simpul orangtua yang berlokasi pada level terbawah di atas simpul yang memberikan leluhur material atau non-material dari simpul tersebut, lalu berikan prioritas terhadap simpul material. Yaitu, bila terdapat leluhur kandidat material dan non-material, pilih secara acak diantara simpul material, sebaliknya pilih secara acak dari simpul non-material.

C3. Pilih sebuah kandidat simpul orangtua yang berlokasi pada level terbawah di atas simpul yang memberikan leluhur material atau non-material dari simpul tersebut, lalu berikan prioritas terhadap simpul material dan berikan prioritas antara simpul non-material pada simpul yang memiliki tepat 1 anak dan kemudian kepada simpul dengan jumlah anak terbesar. Yaitu, jika terdapat leluhur kandidat material dan non-material, pilih secara acak antara simpul material, sebaliknya pilih secara acak dari simpul non-material yang memiliki 1 anak dan jika tidak terdapat simpul yang memenuhi kriteria tersebut, pilih secara acak dari simpul non-material yang memiliki jumlah anak terbesar.

Sebagai contoh, simpul BCD pada Gambar 7 memiliki set simpul orangtua kandidat sebagai berikut: {B, C, D, BC, BD, CD}. Simpul orangtua kandidat pada level terendah yaitu BC dan mudah untuk dilihat bahwa simpul material ini lebih dipilih dibandingkan simpul material lainnya. Sebagai contoh, jika simpul material C dipilih sebagai orangtua dari simpul BCD, Bob secara langsung hanya mengetahui kunci KBC. Sebagai contoh lainnya, pada simpul AD yang simpul orangtua kandidatnya adalah simpul non-material A dan D. Simpul A memiliki 1 anak dan karena itu, sesuai dengan kriteria 3, akan dipilih sebagai orangtua dari simpul AD. Secara intuitif, alasan rasional dibalik pemilihan kriteria 3 untuk simpul non-material dengan anak yang sifatnya simpul non-material adalah hanya berguna jika diperbolehkan adanya "*factorize*" terhadap kunci, yaitu jika diperbolehkan untuk memberikan kepada pengguna kunci yang lebih tinggi yang memungkinkan untuk memperoleh lebih dari 1 kunci level dibawahnya, maka akan menghemat jumlah kunci yang pengguna harus terima. Dalam contoh ini, Alice dengan pengetahuan atas kunci KA mampu untuk memperoleh pasangan kunci KAB dan KAD. Gambar 8 mengilustrasikan pohon yang diperoleh setelah akhir dari langkah kedua ini.

Langkah 3: Pemangkasan pohon

Langkah ketiga ini terdiri dari pemangkasan pohon dimana:



Gambar 9. Pohon yang berkoresponden dengan DAG pada Gambar 5 setelah langkah pemangkasan

1) simpul pohon non-material yang hanya memiliki 1 anak, dan
2) daun pohon non-material

dibuang dari pohon. Pada kasus pertama, sebuah simpul orangtua alternatif perlu dipilih sebagai anak. Pemilihan ini dilakukan dengan menerapkan 2 kriteria berikut:

P1. Pilih dalam level dari simpul yang dibuang sebuah simpul non-material dengan setidaknya 1 anak.

P2. Jika tidak terdapat simpul orangtua kandidat alternatif dalam level dari simpul yang dibuang, hubungkan anak dengan orangtua dari simpul yang dibuang tersebut.

Sebagai contoh pada Gambar 8, simpul AD dibuang dan simpul A dipilih sebagai orangtua dari simpul ACD karena tidak terdapat simpul orangtua kandidat dalam level 2. Penting untuk diperhatikan bahwa pemangkasan dilakukan dengan pertama-tama membuang simpul non-material yang hanya memiliki 1 anak dan kemudian membuang daun dari simpul non-material apapun. Gambar 9 mengilustrasikan pohon yang diperoleh setelah pemangkasan. Pohon hasil tersebut kemudian digunakan untuk memaksakan aturan pengawasan akses yang dispesifikasikan pada matriks pengaksesan sebelumnya. Hal tersebut berarti bahwa setiap tuple dienkripsi dengan kunci yang berasosiasi dengan simpul pada pohon yang berkoresponden dengan ACL. Pemilik data mengkomunikasikan dengan setiap user u kunci yang berasosiasi dengan simpul V dalam pohon sehingga pengguna $u \in V$ dan $u \notin W$, dimana W adalah orangtua dari V dalam pohon. Sebagai contoh, dengan melihat pada Gambar 9, Carol seharusnya mengetahui kunci {KC,KBC,KACD,KBCD}. Bagaimanapun kunci yang seharusnya dikomunikasikan kepada Carol hanyalah kunci {KC, KACD} karena kunci KC dapat digunakan untuk memperoleh kunci KBCD. Biaya dari algoritma transformasi ini yaitu:

$O(|NM \& M|^2)$

dimana:

- NM: set simpul non-material yang pada awal dimasukkan ke dalam graph,
- M: set dari simpul material

Biaya ini sendiri maksimal adalah kuadrat dari jumlah simpul yang dikomputasi pada langkah pertama dari algoritma ini. Karena masalah minimasi pohon ini cukup kompleks, algoritma yang dapat menjamin untuk mengidentifikasi pohon yang optimal ini memiliki kompleksitas yang eksponensial (yaitu $P \neq NP$)

5. Manajemen Metadata pada Skenario DAS

Untuk dapat mengakses dan mengatur basisdata *outsourced*, maka pengguna, pemilik data, dan jika memungkinkan, *server*, harus menyimpan beberapa informasi tambahan yang disebut metadata. *Client* dan *server* menggunakan metadata ini untuk menginterpretasikan dan mengeksekusi pernyataan SQL, dan untuk mengatur data yang disimpan. Distribusi dari metadata harus mengikuti 2 prinsip berikut:

i) Pengguna harus mengetahui metadata tambahan apapun yang dibutuhkan untuk mengakses data yang merupakan hak mereka.
ii) Pengguna harus dapat melakukan pencarian dan query secara efisien terhadap metadata dengan menghemat pada biaya bandwidth.

Untuk tujuan ini, metadata disimpan pada tabel relasional yang dapat diakses menggunakan query SQL seperti tipe data lainnya. Metadata dapat sesederhana seperti 1 kata kunci, atau dapat pula menjadi sangat kompleks seperti pada proses pengambilan jalur untuk mengkomputasi kunci. Terdapat 3 tipe utama dari metadata: metadata autorisasi, metadata deskriptif, dan metadata manajemen kunci. Metadata autorisasi meliputi informasi mengenai aturan pengawasan akses yang didefinisikan oeh pemilik data (yaitu berupa matriks pengaksesan). Pada dasarnya, metadata

autorisasi terdiri dari tabel berikut (*primary key* ditandai dengan garis bawah):

- **TabUser(IdUser, Surname, Name),**
  yang mengurus informasi mengenai setiap pengguna yang ada pada sistem. Skema pada tabel ini tergantung pada informasi yang dibutuhkan oleh pemilik data. Untuk sederhananya maka diasumsikan

  bahwa setiap pengguna diidentifikasi oleh identitas yang unik (atribut IdUser) dan memiliki sebuah nama (atribut Name) dan nama keluarga (atribut Surname).
- **AccessMatrix(ERelation, Counter, IdUser),**
  Yang mengurus informasi mengenai 'siapa' (atribut IdUser) yang dapat mengakses 'apa' (atribut Erelation, dan Counter).

Tabel-tabel ini sangat sensitif dan karenanya harus disimpan pada situs pemilik data. Sebagai contoh pada matriks pengaksesan pada Gambar 4, metadata autorisasi yang berkorespondensi digambarkan pada Gambar 10. Metadata deskriptif adalah deskripsi data dan sama dengan katalog sistem yang secara otomatis dikelola oleh sistem basisdata relasional. Pada dasarnya, metadata deskriptif menggambarkan struktur dari basisdata terenkripsi. Tabel utama pada metadata deskriptif adalah sebagai berikut:

- **TabRelation(Relation, EncryptedRel),**
  mengelola korespondensi antara nama dari relasi *plaintext* (atribut Relation) dan nama dari relasi enkripsi yang berkoresponden dengannya (atribut EncryptedRel).

- **TabIndex(Relation, Attribute, Index, IdMethod),**
  mengelola korespondensi antara nama dari sebuah atribut (atribut Attribute) pada relasi *plaintext* (atribut Relation) dan nama dari indeks yang berkoresponden dengannya (atribut IdMethod).

- **TabMethod(IdMethod, Function, IdParameter, Value),**
  mengelola informasi mengenai fungsi hash (atribut Function) yang digunakan dengan metode indeks spesifik (atribut IdMethod) bersama dengan nilai (atribut Value) dari

  parameter yang berkoresponden (atribut IdParameter).

- **EncryptAlgo(Algorithm, IdParameter, Value),**
  mengelola informasi mengenai fungsi enkripsi (atribut Algorithm) yang digunakan untuk mengenkripsi data bersama dengan nilai (atribut Valued) dari parameter yang berkoresponden (atribut IdParameter).

Tabel ini sangat mungkin untuk diakses oleh pihak yang tidak berwenang. Karenanya, metadata deskriptif tidak boleh disimpan di sisi *server*. Sebagai catatan bahwa setiap pengguna hanya boleh mengetahui metadata deskriptif sesuai dengan porsi hak membaca yang dimilikinya pada matriks pengaksesan, sedangkan pemilik data memiliki pengetahuan yang lengkap mengenai metadata ini. Sebagai contoh, Gambar 10 menggambarkan metadata deskriptif yang berasosiasi dengan pemilik data dan *client* bernama Carol. Diasumsikan bahwa metode pengindeks yang digunakan yaitu fungsi hash yang diimplementasikan melalui operator modular, yaitu

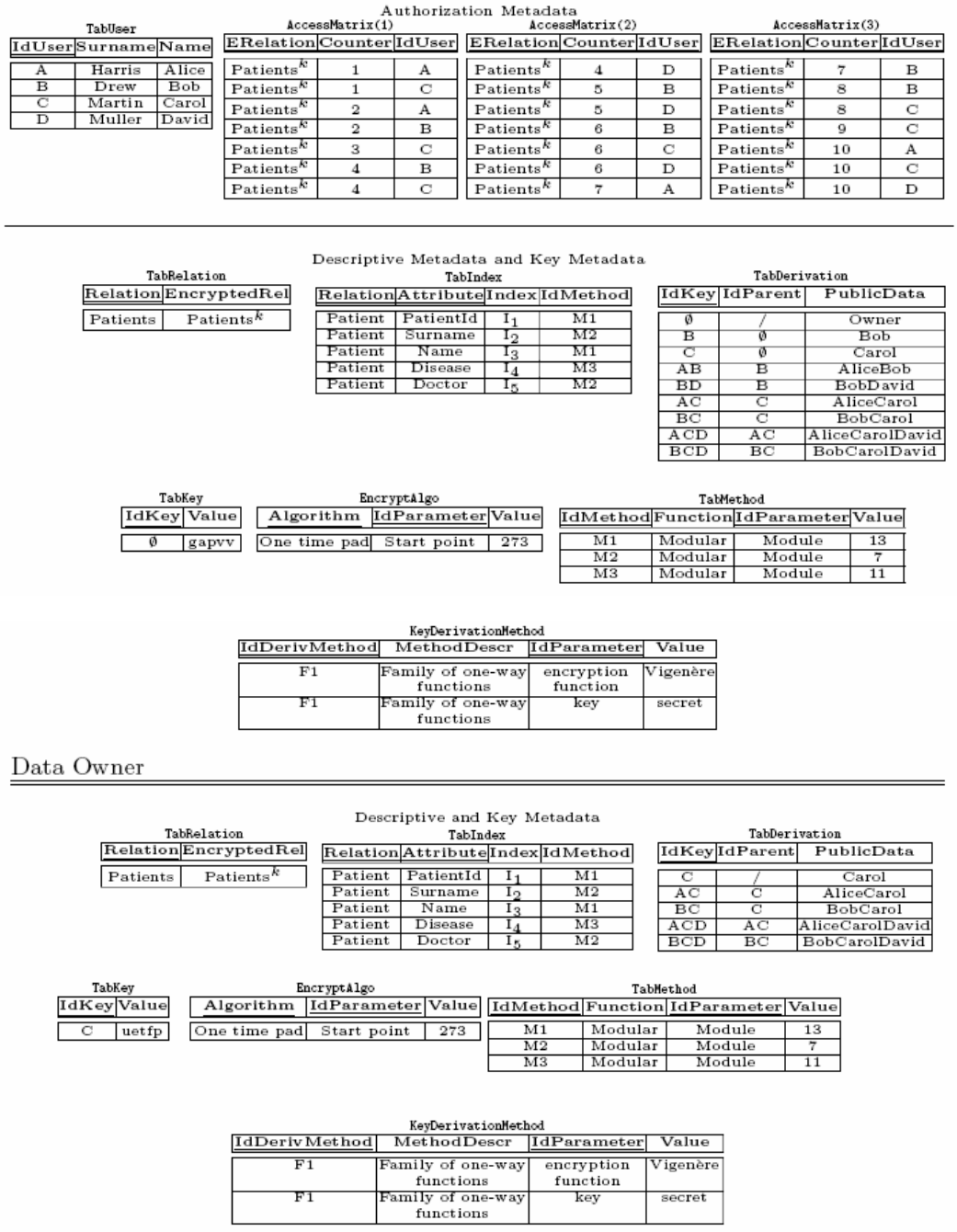
I = A mod M

dimana:

- I adalah nilai index berkoresponden dengan atribut A, dan
- M adalah angka prima yang disimpan pada tabel TabMethod

5.1. Penyimpanan Metadata Kunci pada Sisi *Client*

Metadata manajemen kunci disimpan pada masing-masing *client* termasuk informasi mengenai bagian dari hirarki pohon pengguna yang berasosiasi dengan pengguna. Untuk setiap sub-hirarki memungkinkan pengguna untuk memperoleh kunci yang dibutuhkan untuk mendekrip data yang merupakan bagian dari hak bacanya. Sebagai contoh, mengacu pada hirarki pohon pengguna pada Gambar 6, pengguna bernama Carol harus mengetahui bagian dari akar hirarki pada simpul C. Sedangkan tabel relasional yang disimpan pada sisi *client* terdiri dari tabel berikut:



Gambar 10. Metadata yang berasosiasi dengan pemilik data dan *client* Carol

- **TabKey(IdKey, Value),** mengelola nilai dari kunci (atribut Value) dan mengkomunikasikannya langsung kepada pengguna.
- **TabDerivation(IdKey, IdParent, PublicData),** mengelola untuk setiap kunci (atribut IdKey) pada sub-hirarki yang bersangkutan, *identifier* dari orangtuanya (atribut IdParent) dan informasi publik (atribut PublicData) yang dibutuhkan untuk memperoleh kunci. Jika kunci berkoresponden kepada akar dari sub-hirarki maka atribut IdParent disesuaikan dengan mengubah nilainya menjadi /.

- **TabDecryption(EncryptedRelation, Counter, IdKey),** mengelola untuk setiap relasi terenkripsi (atribut EncryptedRelation) dan setiap tuple pada relasi (atribut Counter), *identifier* (atribut IdKey) dari kunci dekripsi berasosiasi dengan tuple tersebut.

- **KeyDerivationMethod(IdDerivMethod, MethodDescr, IdParameter, Value),** mengelola informasi mengenai metode pemerolehan kunci yang digunakan untuk memperoleh kunci yang berasosiasi dengan simpul pada hirarki pohon pengguna.

Sementara pengguna harus menyimpan secara lengkap sub-hirarki yang dapat mereka akses, pemilik data dapat menentukan apakah ingin menyimpan sejarah dari informasi yang berasosiasi dengan setiap simpul pada hirarki (yaitu *identifier* dan parameter publik yang digunakan oleh metode pemerolehan kunci) tanpa menyimpan hubungan orangtua-anaknya. Yaitu, pemilik data dapat memilih untuk menyimpan versi sederhana dari tabel TabDerivation yang mengandung hanya atribut IdKey dan PublicData. Walaupun solusi ini memungkinkan pemilik data untuk menghemat kapasitas penyimpanannya, hal ini membutuhkan komputasi ulang pada hirarki pohon pengguna setiapkali pemilik data ingin mengakses data. Ditambah lagi, jika matriks pengaksesan berubah (sebagai contoh bila pengguna ditetapkan tidak dapat mengakses sebuah tuple lagi) dan hirarki pohon pengguna diubah tanpa menggunakan algoritma transformasi, maka versi baru dari hirarki tersebut akan berbeda dengan yang diperoleh menggunakan algoritma. Untuk kasus ini, maka pemilik data harus menyimpan tabel TabDerivation seperti didefinisikan di atas.

5.2. Penyimpanan Metadata Kunci pada Sisi *Server*

Pendekatan sisi *client* untuk menyimpan metadata manajemen kunci memiliki keuntungan bahwa setiap pengguna dapat secara langsung mengetahui informasi yang dibutuhkan untuk dapat mengakses basisdata terenkripsi. Bagaimanapun, dengan menganalisa data tersebut lebih detail, mudah untuk dilihat bahwa pendekatan ini membutuhkan duplikasi dari informasi: asosiasi antara tuple t dan *identifier* dari kunci yang digunakan untuk mengenkripsi diduplikasi oleh setiap pengguna yang dapat mengakses t (table TabDecryption). Hal yang sama terjadi pada jalur pemerolehan kunci: 2 pengguna dengan sub-hirarki pohon yang non-*disjoint* memiliki sejumlah bagian dari jalur pemerolehan kunci replika pada tabel TabDerivation. Satu-satunya informasi sensitif yang tidak boleh disimpan pada sisi *server* yaitu tabel TabKey. Karena itu, untuk mencegah duplikasi data dan memungkinkan pertukaran informasi antara banyak pengguna, hirarki pohon pengguna dan *identifier* kunci tuple dapat disimpan pada sisi *server*. Untuk tujuan ini, tabel TabDerivation terdiri dari seluruh hirarki pohon pengguna yang dikelola sepenuhnya oleh *server* dan atribut IdKey yang didefinisikan pada skema relasional dari relasi terenkripsi digunakan untuk mengelola asosiasi *identifier* kunci tuple.

Untuk meyakinkan integritas metadata, kode autentikasi pesan yang melibatkan kunci rahasia dalam komputasi dari *digest* akan digunakan. Tentunya kunci yang digunakan harus diketahui oleh seluruh pengguna dalam sistem. Salah satu kelemahan yang ada dengan penggunaan solusi ini yaitu traversal untuk hirarki pohon pengguna hanya dapat dilakukan oleh *client*. Hal ini berarti bahwa untuk memperoleh sebuah kunci, *client* harus melakukan sejumlah query yang menghasilkan simpul pohon pada jalur pemerolehan kunci. Kelemahan lainnya dari solusi ini yaitu karena keberadaan atribut tambahan IdKey, ukuran hasil yang dikembalikan kepada *client* lebih besar daripada ukuran hasil yang diberikan dengan menggunakan strategi pada sisi *client*. Bagaimanapun, dampak dari atribut IdKey pada ukuran hasil adalah minimal dan dapat diabaikan.

5.3. Solusi Kombinasi Sisi *Client* dan Sisi *Server*

Solusi campuran untuk menyimpan metadata manajemen kunci dapat juga diadaptasi dengan menggabungkan keuntungan dari kedua strategi yang telah dijelaskan sebelumnya. Sebagai contoh, asosiasi antara *identifier* kunci tuple dapat disimpan pada sisi *server* dengan menggunakan atribut IdKey yang telah dibahas sebelumnya, dan informasi yang digunakan untuk pemerolehan kunci (yaitu hirarki pohon pengguna, metode pemerolehan kunci, dan informasi publik yang berasosiasi dengan setiap elemen yang ada pada hirarki) dapat disimpan pada sisi *client*. Dengan cara ini, dapat dihindari adanya duplikasi dari informasi dan proses pemerolehan kunci menjadi lebih

efisien karena *client* dapat mengeksekusinya tanpa melakukan query pada *server*.

Pemilihan antara strategi sisi *client*, sisi *server*, atau solusi gabungan ini tergantung pada kapasitas penyimpanan dan *bandwidth* yang tersedia untuk *client*. Sebagai contoh, jika kapasitas penyimpanan adalah sumber daya yang lebih kritis daripada kapasitas *bandwidth*, maka solusi strategi di sisi *server* lebih dipilih. Namun sebaliknya, jika kapasitas *bandwidth* lebih kritis daripada kapasitas penyimpanan, maka solusi strategi pada sisi *client* atau solusi gabungan lebih dipilih. Sebagai catatan bahwa saat metode pemerolehan kunci tertentu digunakan, ukuran dari data publik yang disimpan pada sisi *client* sangat minimal dan dampaknya pada kapasitas penyimpanan dapat diabaikan. Sebagai contoh, metode pemerolehan kunci berbasiskan fungsi *hash* membutuhkan nama unik sebagai informasi publik yang berasosiasi dengan setiap simpul dari hirarki. Ukuran dari informasi publik ini yaitu:

$O(n \log n)$

dimana:

- n: jumlah elemen dalam hirarki pohon pengguna

Metode pemerolehan kunci yang bekerja menggunakan DAGs dan berbasiskan pada teknik eksponensial modular akan menggunakan data publik yang berasosiasi dengan sebuah elemen n dari hirarki, yaitu produk dari angka prima yang berasosiasi dengan simpul pada hirarki yang tidak didominasi oleh n. Karena itu, pada kasus terburuk (yaitu untuk sebuah daun dari hirarki) ukuran dari informasi publik yaitu:

$O(n(n-1)k) = O(n^2k)$

dimana:

- n: jumlah elemen dalam hirarki
- k: jumlah bits untuk merepresen-tasikan angka prima yang diguankan

Biaya komputasi dari mekanisme pemerolehan kunci dapat direduksi jika setiap *client* menyimpan *cache* dari kunci-kunci yang sudah dikomputasi. Dengan menggunakan cara ini, jika hasil dari suatu query meliputi tuple terenkripsi dengan salah satu dari kunci yang telah dikomputasi sebelumnya, maka tidak diperlukan adanya komputasi ulang terhadap kunci dekripsi. Tentu saja mekanisme penyimpanan *cache* ini membutuhkan kapasitas penyimpanan tambahan pada *client*. Juga penting untuk diperhatikan bahwa apabila terjadi perubahan pada matriks pengaksesan, *cache* harus dihapus karena kunci-kunci dapat berubah.

6. Pemrosesan Query

Bagian ini akan membahas mengenai evaluasi query *client* pada skenario DAS dimana solusi gabungan (sisi *client* dan sisi *server*) digunakan untuk menyimpan metadata digunakan. Singkatnya, diasumsikan bahwa basisdata terenkripsi B^k terdiri dari relasi tunggal $Patients^k$ (lihat Gambar 3), dan query berupa ekspresi seleksi-proyeksi. Berdasarkan pada metadata yang disimpan, query Q pada relasi *plaintext* dibagi menjadi query Qs pada relasi enkripsi yang berkoresponden yang kemudian dieksekusi oleh *server*, dan query *client* Qc untuk hasil pasca-proses dari query *server*. Transformasi dari query Q dan query Qs diilustrasikan seperti pada Gambar 11.

Original Clause (Q)	Transformed Clause (Q_s)
SELECT $A_1, \ldots, A_n$	SELECT Counter, Etuple, IdKey
FROM $R_1, \ldots, R_m$	FROM $R_1^k, \ldots, R_m^k$
WHERE $A_j = \text{val}$	WHERE $I_{A_j} = f(\text{val})$
$A_k = A_w$	$I_{A_k} = I_{A_w}$

Gambar 11. Transformasi query

Seperti dapat dilihat dari tabel tersebut, daftar atribut dalam klausa SELECT digantikan dengan atribut Counter, Etuple, dan IdKey, sebagai dampak dari fakta bahwa relasi dienkripsi pada level tuple, maka *server* hanya dapat mengembalikan seluruh tuple terenkripsi Etuple, dan menyebabkan operasi proyeksi tidak dapat dilakukan pada sisi *server*. Atribut IdKey dibutuhkan untuk mengidentifikasi kunci dekripsi. Daftar dari relasi pada klausa FROM digantikan dengan daftar dari relasi terenkripsi yang berkoresponden (tabel TabRelation) dan kondisi pada klausa WHERE ditransformasi sesuai dengan teknik indeks yang digunakan. Lebih jelasnya, setiap atribut Aj dalam klausa WHERE digantikan dengan indeks yang berkoresponden (tabel TabIndex) dan nilai konstan ditransformasi dengan menerapkan teknik indeks yang sesuai (tabel TabMethod). Sebagai contoh, jika Carol ingin menemukan nama depan, nama keluarga, dan dokter dari pasien yang memiliki penyakit “Tosillitis”, maka query SQL yang digunakan yaitu:

```
SELECT PatientId, Surname, Name,
Doctor
FROM Patients
WHERE Disease = “Tonsillitis”
```

Modul pemrosesan query kemudian akan menerima dari metadata nama dari relasi terenkripsi yang berkoresponden dengan Patients, nama dari indeks yang berkoresponden dengan atribut Disease dan fungsi *hash* (dan parameter-parameternya) digunakan untuk membentuk indeks. Dalam fase ini juga dibutuhkan untuk mengambil baik fungsi enkripsi maupun metode pemerolehan kunci bersama-sama dengan parameternya.

Counter	Etuple	IdKey
1	*r*tso/yui+*	*AC*
2	hai4de-0q1	AB
5	3gia*ni+aL	BD
8	*/bur21/*-D*	*BC*
10	*bew0"!DE1a*	*ACD*

Gambar 12. Hasil query terenkripsi

PatientId	Surname	Name	Doctor
125YP894	Carter	Andrew	Wayne
364UK784	Rogers	Laura	Wayne

Gambar 13. Hasil final yang diberikan kepada Carol

Untuk tujuan ini, berikut adalah query SQL yang akan dieksekusi:

```
SELECT EncryptedRel INOT :R
FROM TabRelation
WHERE Relation = "Patients"

SELECT Function INTO :h
FROM TabMethod
WHERE IdMethod = :M

SELECT Algorithm, Value INTO :E,
:Pe
FROM EncryptAlgo

SELECT Index, IdMethod INTO :I, :M
FROM TabIndex
WHERE Relation = "Patients"
AND Attribute = "Disease"
SELECT Value INTO :P
FROM TabMethod
WHERE IdMethod = :M

SELECT MethodDescr, Value INTO
:DM, :Pm
FROM KeyDerivatonMethod
```

Dengan mengasumsikan bahwa nilai dari variabel R adalah Patients^k, nilai dari variabel I adalah I_4, dan nilai indeks yang berkoresponden dengan "Tosillitis" adalah π, query *plaintext* Q yang asli ditranslasi sebagai berikut:

```
SELECT Counter, Etuple, IdKey
FROM Patients^k
WHERE I4 = "π"
```

Gambar 12 mengilustrasikan hasil query yang dikembalikan kepada *client*. *Client* harus mendekrip seluruh tuple yang dapat diakses oleh Carol (sebuah tuple dapat diakses oleh pengguna saat nilai IdKey yang berkoresponden muncul pada tabel TabKey atau tabel TabDerivation) dan juga untuk menyaring yang tidak sesuai dengan predikat asli query Q. Dalam contoh ini, Carol dapat mengakses tuple t1, t8, dan t10 sedangkan tuple t2 dan t5 dapat diabaikan. Kunci yang akan digunakan untuk mendekrip tuple tersebut dihitung sebagai berikut. Pertama, untuk setiap tuple t dari hasil query, jika tabel TabKey mengandung sebuah tuple t` dimana t`[IdKey] = t[IdKey] maka kunci dekripsi telah tersedia. Sebaliknya, jika kunci dekripsi diperoleh dengan mengikuti jalur pemerolehan kunci yang disimpan pada tabel TabDerivation.

Gambar 14 mengilustrasikan prosedur untuk mengkomputasi jalur pemerolehan kunci. Untuk setiap tuple t, dimulai dari daun (yaitu t[IdKey]) pada jalur, tabel TabDerivation diquery-kan untuk menentukan orangtua dari simpul *current* dari jalur. Prosedur ini dihentikan sat akar / ditemukan. *Array* Path menyimpan jalur pemerolehan kunci dalam urutan yang terbalik. Sebagai contoh pada tuple t10, dilakukan enkripsi dengan kunci k_{ACD} yang dapat diperoleh dengan mengikut jalur Path[3] = $k_C \rightarrow$ Path[2] = $k_{AC} \rightarrow$ Path[1] = k_{ACD}. Kunci dekripsi dikomputasi dengan melakukan metode pemerolehan kunci DM sepanjang Path. Lalu, *client* harus melakukan langkah berikut:

1) mendekrip tuple menggunakan fungsi E.
2) terapkan kondisi original untuk menghilangkan kemungkinan tuple palsu yang tidak termasuk bagian dari set hasil.
3) Mengeksekusi operasi proyeksi yang diminta.

Algorithm 1 (Key derivation path)

```
KeyDerivationPath(t[IdKey])
/* Initializes some variables */
i:=1; Path[i]:= t[IdKey]
While Path[i] ≠ / do
        i := i + 1
        SELECT IdParent INTO :Path[i]
        FROM TabDerivation
        WHERE IdKey=:Path[i − 1]
return Path
```

Gambar 14. Algoritma untuk mengkomputasi jalur pemerolehan kunci

Tuple palsu dapat dihilangkan dengan melakukan query berikut:

```
SELECT PatientId, Surname, Name,
Doctor
FROM Result
WHERE Disease= “Tonsillitis”
```

Gambar 13 melaporkan set final dari tuple yang dapat dibaca oleh pengguna dengan nama Carol. Sebagai catatan bahwa tuple-tuple ini adalah himpunan bagian dari tuple yang berhak dibaca oleh Carol (lihat kembali matriks pengaksesan pada Gambar 4).

7. Kesimpulan

Untuk mengatasi permasalahan keamanan atas basisdata yang diletakkan pada dunia Internet yaitu dengan dilakukannya enkripsi terhadap basisdata tersebut. Namun pengenkripsian basisdata juga menyebabkan masalah lain yaitu sulitnya mengatur hak pengaksesan terhadap jenis pengguna yang beragam. Selain itu, karena kerahasiaan data membutuhkan syarat bahwa dekripsi data hanya boleh dilakukan pada sisi *client*, teknik-teknik tertentu diperlukan untuk memungkinkan *server* eksternal dapat mengeksekusi query pada data terenkripsi, atau bagaimana cara agar seluruh relasi yang berhubungan dengan query tersebut dapat dikirimkan ke sisi *client* untuk melakukan eksekusi query.

Solusi yang dapat diberikan untuk menjaga keamanan atas kunci yaitu dengan melakukan manajeman metadata. Manajemen atas metadata untuk mengakses *remote* basisdata terenkripsi adalah salah satu kebutuhan penting dari skenario *database-as-a-service*. Dalam pembahasan sebelumnya, metadata dibuat untuk menyediakan deskripsi abstrak dari struktur data dan format data yang digunakan dalam sistem. *Client* dan *server* menggunakan metadata ini untuk menginterpretasikan dan mengeksekusi pernyataan SQL, dan untuk mengatur data yang disimpan. Distribusi dari metadata harus mengikuti 2 prinsip berikut:

i) Pengguna harus mengetahui metadata tambahan apapun yang dibutuhkan untuk mengakses data yang merupakan hak mereka.
ii) Pengguna harus dapat melakukan pencarian dan query secara efisien terhadap metadata dengan menghemat pada biaya bandwidth.

Metode penyimpanan metadata sendiri dapat dibagi menjadi 3 cara:

i) Penyimpanan metadata kunci pada sisi *client*
ii) Penyimpanan metadata kunci pada sisi *server*
iii) Penyimpanan metadata kunci metode gabungan yatiu pada sisi *client* dan *server*

Sedangkan masalah yang masih harus dikaji dalam *issue* atas manajemen metadata kunci ini antara lain yaitu:

i) Implementasi yang efektif atas solusi berbeda yang ada untuk mengevaluasi *trade-off* antara konsumsi penyimpanan dan *bandwidth*.
ii) Evaluasi dari strategi yang dialamatkan kepada perubahan dinamis atas hak pengaksesan

Daftar Pustaka

Damiani, E., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati P. *Key Management for Multi-User Encrypted Databases* Universita di Milano 26013 Creme. Italia.

Damiani, E., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P. *Metadata Management in Outsourced Encrypted Databases*. Universita di Milano 26013 Creme. Italia.

Thales e-Security. *Database Encryption, WebSentry Enables High Performance Database Security*. http://www.thales.e-security.com.

E. Damiani, S. De Capitani di Vimercati, M. Finetti, S. Paraboschi, P. Samarati, and S. Jajodia. *Implementation of a storage mechanism for untrusted DBMSs*. In Proc. of the Second International IEEE Security in Storage Workshop, Washington DC, USA, May 2003.

E. Damiani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, and P. Samarati. *Selective release of information in outsourced encrypted database*. Technical report, University of Milan, 2005.

E. Damiani, S. De Capitani di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. *Balancing confidentiality and efficiency in untrusted relational DBMSs*. In Proc. of the 10th ACM Conference on Computer and Communications Security, Washington, DC, USA, October 27-31 2003.