Pencarian Kata Kunci pada Teks Terenkripsi dengan Teknik *Inverted Indexing*

Pencarian Kata Kunci Pada Pesan Aplikasi *Instant Messaging*

Garmastewira (13514068)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
ranggarmaste@gmail.com

***Abstrak*—Makalah ini memperkenalkan sebuah teknik pencarian kata kunci pada teks terenkripsi dengan teknik *inverted indexing*. Penggunaan pencarian kata kunci berbasis *inverted index* sudah digunakan pada banyak aplikasi *instant messaging* dan *email* client. Metode yang digunakan adalah dengan menyimpan pemetaan kata kunci yang telah dienkripsi terhadap pesan-pesan yang mengandung kata kunci tersebut. Hasil eksperimen dan analisis menunjukkan bahwa pencarian kata dengan memanfaatkan *indexing* memiliki kinerja yang jauh lebih baik dengan pencarian dengan melakukan dekripsi keseluruhan pesan.**

Kata kunci—AES; ECDH; enkripsi; indeks; kata kunci; penyimpanan; pencarian.

I. PENDAHULUAN

Masyarakat pada saat ini memilih untuk menggunakan aplikasi *instan messaging* untuk berkomunikasi. Komunikasi melalui aplikasi seperti WhatsApp sudah menjadi bagian dari kehidupan sehari-hari. Tentunya, salah satu aspek yang harus diperhatikan pada aplikasi *instant messaging* adalah aspek keamanan. Pesan yang dikirimkan oleh pengguna seharusnya tidak dapat diretas oleh orang lain. Solusi untuk menangani hal tersebut adalah dengan enkripsi.

Pada *instant messaging*, salah satu operasi yang sering dilakukan pengguna adalah pencarian pesan yang sudah dikirimkan dahulu berdasarkan kata kunci. Jika aplikasi menggunakan enkripsi untuk mengamankan pesan, maka konten pesan di basis data pasti berupa cipherteks. Cara paling naif untuk melakukan pencarian adalah dengan cara mendekripsi seluruh berkas yang ada terlebih dahulu, kemudian melakukan *string matching* pada setiap dokumen. Namun, proses ini akan memakan waktu yang sangat lama, terutama jika jumlah berkas sangat banyak.

Solusi yang umum digunakan oleh banyak layanan internet adalah tanpa melakukan dekripsi data. Pencarian murni dilakukan pada berkas terenkripsi. Penelitian pencarian pada teks terenkripsi sendiri sudah banyak dan memiliki banyak pendekatan, seperti *functional encryption* [1], *fully homomorphic encryption* [2]. dan *ranked keyword search* [3]. Namun dari seluruh pendekatan, pendekatan yang paling terkenal adalah *index-based* [4]. Pendekatan *index-based* diadaptasi oleh aplikasi *instant messaging* seperti Slack dan WhatsApp.

Makalah ini akan memperlihatkan hasil implementasi sebuah aplikasi *instant messaging* web yang memanfaatkan pencarian kata kunci dengan teknik *inverted indexing* . Enkripsi pesan dilakukan dengan kunci simetris dengan algoritma AES. Kunci simetris dibangkitkan dengan algoritma ECDH. Kata kunci dapat dicari dari pemetaan kata kunci yang terenkripsi terhadap pesan-pesan yang berkoresponden. Eksperimen akan membandingkan kinerja pencarian kata kunci pada pesan dengan teknik solusi ajuan dan teknik dekripsi secara keseluruhan. Selain itu, teknik solusi akan dianalisis dari berbagai aspek, seperti aspek keamanan dan kapasitas penyimpanan daftar indeks kata kunci.

II. DASAR TEORI

A. Kriptografi

Ilmu dan keahlian yang mempelajari tentang teknik menyandikan pesan disebut dengan kriptografi. Kriptografi sudah dikenal sejak zaman dahulu. Bangsa Yunani menggunakan alat *scytale* pada tahun 400 SM, dan Nazi Jerman menggunakan cipher Enigma untuk mengenkripsi pesan pada perang dunia II [5]. Pendekatan ini dinamakan kriptografi klasik karena algoritma beroperasi pada tingkat karakter.

Setelah era komputer dimulai, kriptografi kemudian beroperasi pada tingkat bit-bit data, atau disebut juga kriptografi modern. Algoritma kriptografi modern dapat beroperasi pada tingkat bit (*stream cipher*) atau pada tingkat blok bit (*block cipher*). Banyak sekali algoritma *block cipher* terkenal, seperti DES (Data Encryption Standard) dan AES (Advanced Encryption Standard).

B. AES

AES (*Advanced Encryption Standard*) merupakan standar algoritma enkripsi data biner menggunakan algoritma Rinjdael yang dipublikasikan oleh NIST [6]. Algoritma ini dibuat dan diajukan oleh Vincent Rijnen dan Joan Daemen. AES merupakan algoritma kriptografi kunci simetris yang paling populer. AES menggantikan algoritma sebelumnya yang pernah diadaptasi oleh pemerintahan Amerika Serikat, yaitu DES (*Data Encryption Standard*).

Algoritma Rinjdael menggunakan 10 putaran dengan operasi yang sama pada setiap putarannya, kecuali putaran teakhir. Operasi yang digunakan pada algoritma ini terdiri dari 4 transformasi yang diaplikasikan terhadap *state byte* yang ditulis dalam bentuk heksadesimal. Keempat transformasi tersebut adalah transformasi *sub-bytes, shift rows, mix columns,* dan *add round key*. Proses dekripsi dilakukan dengan menggunakan algoritma yang sama namun melakukan *round* secara terbalik.

C. ECDH

ECDH (*Elliptic Curve Diffie-Helman*) terdiri atas dua buah konsep penting, yaitu ECC (*Elliptic Curve Cryptography*) dan pertukaran kunci Diffie-Helman [7]. ECC merupakan kriptografi kunci publik di mana operasi untuk membangkitkan kunci publik dan privat menggunakan konsep struktur aljabar kurva eliptik [8], sedangkan pertukaran kunci Diffie-Helman merupakan metode untuk melakukan pertukaran kunci kriptografi melalui jaringan publik secara aman [9]. Jadi, ECDH merupakan protokol pertukaran kunci yang memanfaatkan pasangan kunci publik-privat hasil pembangkitan ECC untuk menghasilkan kunci simetris untuk dua pihak. Kunci simetris ini sering juga disebut sebagai *shared secret*.

Misalkan dua orang pengguna bernama Alice dan Bob ingin melakukan pertukaran kunci yang dilakukan. Maka, Alice akan membangkitkan pasangan kunci (pub_A, pri_A) dan Bob akan membangkitkan pasangan kunci (pub_B, pri_B). Kedua pengguna tidak perlu berbagi kunci privat. Setelah mengetahui kunci publik dari masing-masing pihak, maka sesuai dengan konsep ECDH, keduanya dapat berkomunikasi dengan *shared secret* yang sama sesuai dengan persamaan (1).

$$secret = pri_A pub_B = pri_B pub_A$$

D. End-to-end Encryption

End-to-end encryption, seringkali disingkat E2EE, merupakan sistem komunikasi di mana hanya pengguna yang berkomunikasi yang dapat membaca pesan yang telah dikirim [10]. Pada dasarnya, konsep ini mencegah peretas dari segala kalangan, termasuk penyedia layanan internet, pemerintah, dan bahkan penyedia layanan komunikasi tersebut sendiri. Umumnya, aplikasi yang memanfaatkan E2EE menggunakan konsep pertukaran kunci Diffie-Hellman untuk membangkitkan kunci simetris antara dua pengguna.

Kunci privat dan publik masing-masing pengguna dibangkitkan pada perangkat pengguna, dan kunci privat tidak pernah dikirim melewati jaringan. Untuk membangkitkan kunci simetris antar pengguna, pengguna hanya membutuhkan kunci publik dari lawan bicara. Konsep ini sudah diterapkan pada banyak layanan aplikasi *instant messaging*, seperti WhatsApp dan LINE.

E. Searchable Encryption

Searchable Encryption merupakan skema kriptografi yang mampu melakukan pencarian informasi spesifik pada konten terenkripsi tanpa melakukan dekripsi terhadap konten tersebut [11]. Teknik-teknik yang diajukan sudah cukup banyak *functional encryption* [1], *fully homomorphic encryption* [2]. dan *ranked keyword search* [3], dan *index-based* [4]. Pendekatan *index-based* memanfaatkan pemetaan antara kata kunci terenkripsi ke pesan-pesan yang mengandung kata kunci tersebut.

III. Rancangan Solusi

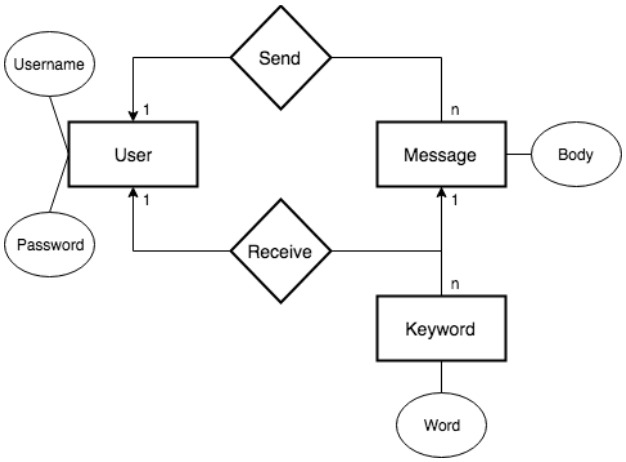
Rancangan yang dipilih untuk pencarian kata kunci pada teks terenkripsi yang efisien adalah dengan menggunakan teknik *inverted index*. Tabel 1 memperlihatkan contoh tabel *inverted index* yang dimaksud. Ketiks pengguna ingin mencari sebuah kata kunci, maka kata kunci tersebut dienkripsi terlebih dahulu, kemudian *server* mencari baris pada tabel *inverted index* yang memiliki entri kata kunci terenkripsi tersebut. Pada implementasinya, tabel pemetaan kata kunci terenkripsi ke pesan bersangkutan akan disimpan pada basis data.

Untuk demonstrasi teknik pencarian ini, akan dibuat sebuah aplikasi *instant messaging* sederhana. Pada aplikasi ini, pengguna dapat bertukar pesan dengan pengguna lain. Fitur yang paling penting tentunya adalah fitur pencarian pesan-pesan dahulu yang mengandung suatu kata kunci untuk menguji kebenaran rancangan solusi.

Tabel 1 Contoh Inverted Index

No	Kata Kunci (Encrypted)	Pesan
1	tomat	ID 1, ID 2
2	penting	ID 2
3	presiden	ID 1, ID 3, ID 6

A. Rancangan Aplikasi Instant Messaging



Gambar 1 Diagram *Entity Relationship* Aplikasi *Instant Messaging*

Rancangan aplikasi *instant messaging* terdiri atas dua aplikasi utama, yaitu aplikasi sisi *backend* (*server*) dan aplikasi sisi *front-end (client)*. Berikut adalah fitur-fitur yang dimiliki aplikasi *client*.

1. Melakukan registrasi
2. Melakukan *login*
3. Melakukan pengiriman pesan antar pengguna
4. Mencari pesan-pesan sebelumnya yang relevan dengan suatu kata kunci

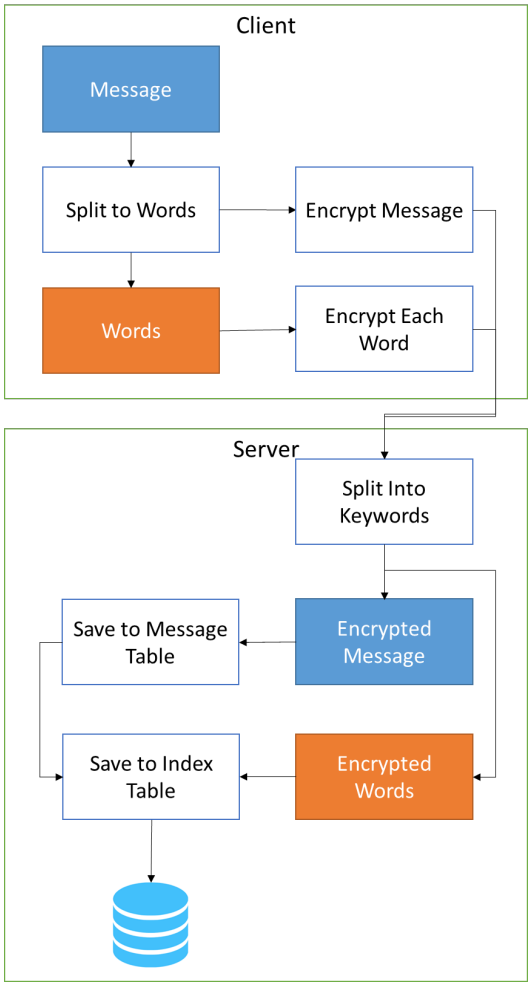
Aplikasi *server* sendiri memiliki dua tugas utama, yaitu meneruskan pesan dari satu pengguna ke pengguna lainnya dan

menyimpan data pada basis data. Gambar 1 memperlihatkan rancangan diagram *entity relationship* pada sisterm *instant messaging* untuk memungkinkan *searchable encryption*. Entitas paling penting dari rancangan ini adalah entitas kata kunci.

B. Skema Penyimpanan Pesan

Gambar 2 memperlihatkan skema penyimpanan pesan beserta proses pengindeksan masing-masing kata kunci. Berikut adalan rincian skema.

1. Membagi pesan menjadi kata-kata kunci
2. Mengenkripsi pesan, kemudian menyimpan pesan tersebut pada basis data
3. Mengenkripsi setiap kata dengan kunci simetris unik yang berkoresponden dengan seorang pengguna
4. Menyimpan *record* (ID Pesan, Kata Kunci Terenkripsi) pada basis data



Gambar 2 Skema Penyimpanan Pesan

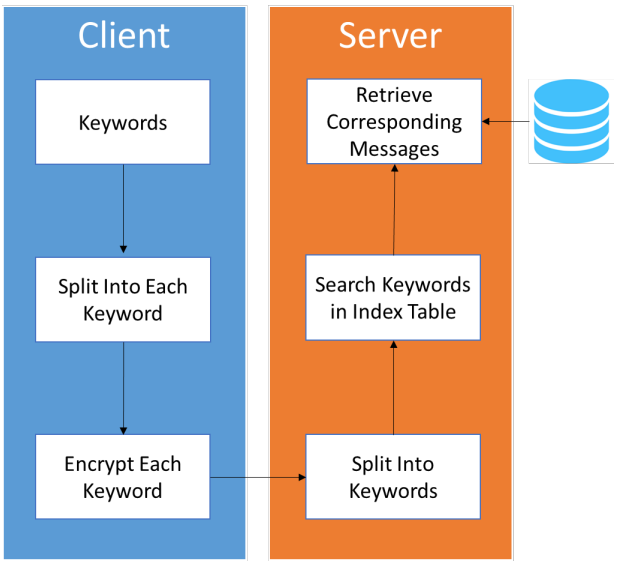
Berikut adalah kode semu untuk skema penyimpanan pesan.

```
/**
Deskripsi
Menyimpan pesan dan indeks kata kunci pada
basis data

Parameter
1. message: pesan dalam plainteks
2. secret: shared secret untuk enkripsi
**/

procedure save(String message, String secret)
begin
    String[] words = splitToKeywords(message)
    String encryptedMessage = encrypt(message,
        secret)
    for i in [0 … words.length-1] do
        word[i] = encrypt(word[i], secret)
    end
    messageID = saveMessage(encryptedMessage)
    saveToIndexTable(words, messageID)
end
```

C. Skema Pencarian Kata Kunci

Gambar 3 memperlihatkan skema pencarian kata kunci terhadap pesan-pesan yang dimiliki oleh pengguna. Berikut adalah rincian skema yang dibutuhkan.

1. Menerima daftar kata kunci yang diberikan oleh pengguna
2. Mengenkripsi setiap kata kunci dengan kunci simetris yang sebelumnya digunakan terhadap seorang pengguna
3. Mencari keberadaan kata kunci yang telah dienkripsi pada basis data untuk mendapatkan ID pesan-pesan yang mengandung kata kunci tersebut
4. Mengembalikan kepada pengguna pesan-pesan yang mengandung kata kunci.

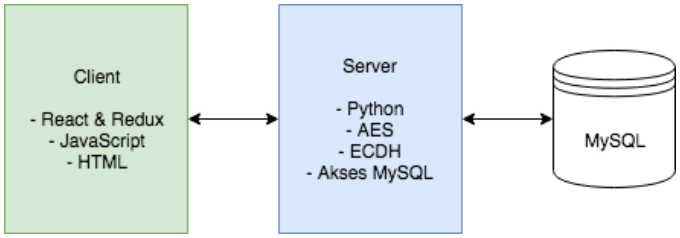


Gambar 3 Skema Pencarian Kata Kunci

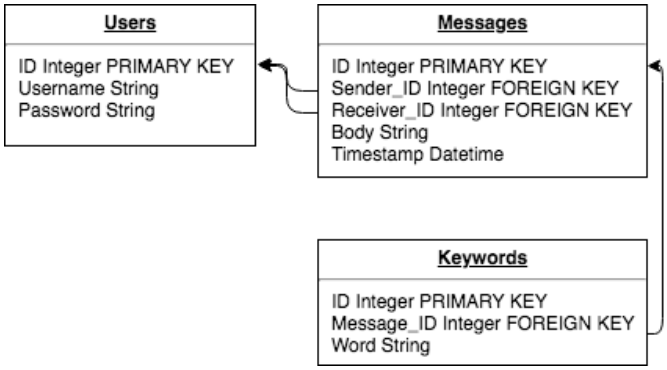
Berikut adalah kode semu untuk melakukan pencarian kata kunci pada pesan.

```
/**
Deskripsi
Mengembalikan pesan-pesan pengguna yang
memiliki suatu kata kunci

Parameter
1. words: daftar kata kunci
2. secret: shared secret untuk enkripsi
**/

function search(String[] words, String
secret): String[] messages
begin
    String[] messages
    for I in [0 … words.length-1] do
        words[i] = encrypt(words, secret)
        messages.concat(retrieveMessages(
            words[i]))
    end
    return messages
end
```

IV. Implementasi dan Pengujian



Gambar 4 Rancangan Implementasi



Gambar 5 Basis Data Relasional *Instant Messaging*

Gambar 4 memperlihatkan rancangan implementasi dari aplikasi *instant* messaging. Pada aplikasi ini, akan dibuat dua buah komponen, yaitu *server backend* yang digunakan untuk melakukan penyimpanan pesan serta aplikasi *client* yang akan digunakan untuk simulasi *chatting*. *Server backend* diimplementasikan dengan bahasa pemrograman Python. Aplikasi klien kemudian diimplementasikan pada *platform web* dengan berbagai macam *library* JavaScript. Untuk pertukaran kunci dan algoritma enkripsi, akan digunakan fungsi ECDH dan AES yang telah disediakan oleh bahasa pemrograman Python.

Gambar 5 kemudian memperlihatkan realisasi diagram *entity relationship* Gambar 1 menjadi basis data relasional. Manajemen basis data yang digunakan adalah MySQL. Terdapat tiga buah tabel yang sama jumlahnya dengan jumlah entitas pada Gambar 1. Hubungan pada diagram *entity relationship* diubah menjadi bentuk *foreign key*.

Pada tahap pengujian, terdapat dua kinerja yang akan diuji sebagai berikut.

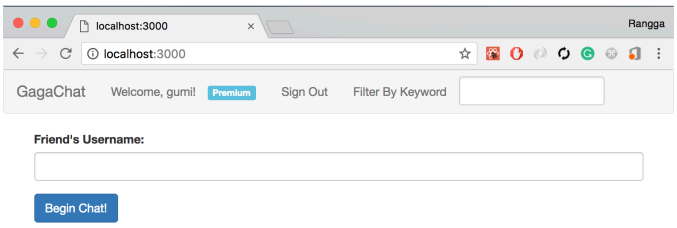
1. Kinerja kecepatan penyimpanan pesan. Pada kinerja ini, akan dibandingkan kinerja penyimpanan tidak memedulikan indeks kata dan jika menyimpan indeks kata.
2. Kinerja kecepatan pencarian kata kunci. Pada kinerja ini akan dibandingkan kinerja pencarian dengan menggunakan teknik pencarian berbasis indeks dan teknik *baseline*. Teknik *baseline* yang dimaksud adalah dengan cara meminta seluruh pesan kepada basis data, melakukan dekripsi satu per satu, dan mencoba menggunakan *string matching* untuk menemukan keberadaan kata kunci.

Khusus untuk kinerja kecepatan pencarian kata kunci, berikut adalah rincian dari aspek yang akan dilihat.

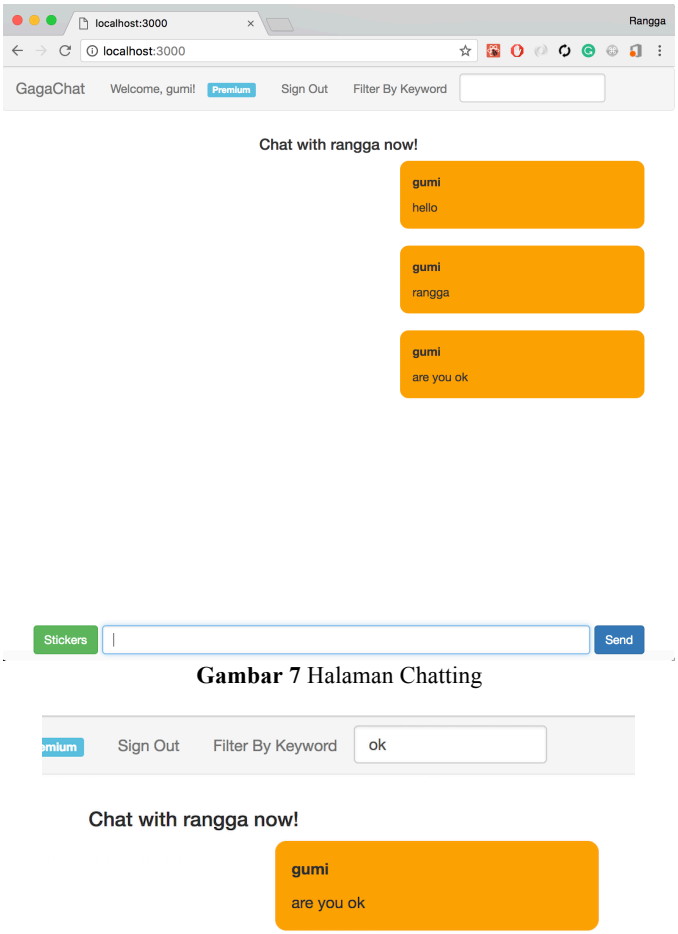
1. Ketika meninjau dari aspek jumlah pesan yang disimpan, pengujian akan melakukan 1000 percobaan. Setiap percobaan akan menyimpan **n** pesan, di mana **n** adalah urutan percobaan. Masing-masing pesan hanya akan terdiri dari 15 kata kunci saja.
2. Ketika meninjau dari aspek jumlah kata kunci yang disimpan, pengujian akan melakukan 1000 percobaan. Setiap percobaan akan menyimpan 50 pesan dan setiap pesan mengandung **n** pesan, di mana **n** adalah urutan percobaan

A. Aplikasi Klien

Gambar 6 dan Gambar 7 memperlihatkan dua halaman utama pada aplikasi. Awalnya, pengguna melakukan registrasi dan kemudian melakukan login dengan *username* dan *password* yang sesuai. Kemudian, pengguna harus memasukkan nama pengguna yang ingin dituju untuk melakukan percakapan. Gambar 8 kemudian memperlihatkan kasus ketika pengguna ingin memfilter pesan-pesan yang memiliki kata kunci "ok". Terlihat bahwa dari tiga pesan pada Gambar 7, hanya satu pesan yang ditampilkan, yaitu pesan yang mengandung kata "ok".



Gambar 6 Halaman Depan Aplikasi



Gambar 7 Halaman Chatting

Gambar 8 Contoh Pencarian Pesan Berdasarkan Kata Kunci "OK"

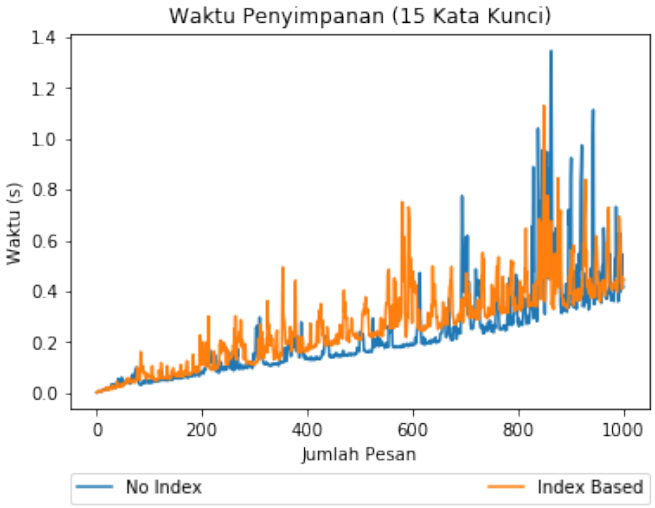
B. Perbandingan Waktu Penyimpanan Pesan

Tabel 2 memperlihatkan hasil secara rata-rata untuk perbandingan waktu penyimpanan pesan. Gambar 9 dan Gambar 10 memperlihatkan perbedaan waktu antara penyimpanan tanpa memedulikan penyimpanan indeks kata dan dengan penyimpanan indeks kata. Grafik bergerak sedikit fluktuatif dikarenakan *overhead* dari inisiasi koneksi serta penutupan koneksi terhadap basis data. Garis berwarna biru menunjukkan penyimpanan tanpa indeks, sedangkan garis berwarna jingga adalah sebaliknya. Pada Gambar 9, terlihat bahwa terdapat sedikit perbedaan antara penyimpanan menggunakan indeks kata dan tanpa indeks kata dari segi beban jumlah pesan. Gambar 10 juga memperlihatkan hal yang serupa namun dari segi jumlah kata kunci.

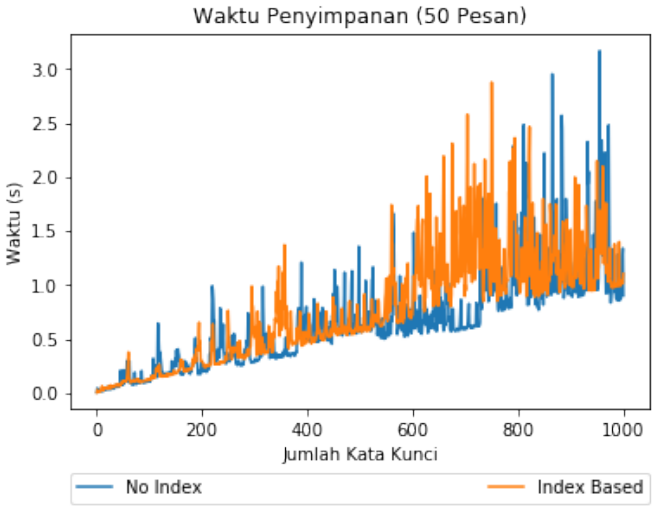
Dari kedua gambar tersebut dapat disimpulkan bahwa penyimapanan indeks kata ke basis data tidak membuat proses penyimpanan pesan menjadi jauh lebih lambat. Dengan demikian, penambahan indeks tidak menimbulkan *bottleneck* yang cukup berarti.

Tabel 2 Hasil Pengujian Waktu Penyimpanan Pesan

No	Aspek	Nilai	Waktu Rata-rata
1	Jumlah Pesan	No Index	0.223 s
2		Index Based	0.261 s
3	Jumlah Kata Kunci	No Index	0.659 s
4		Index Based	0.760 s



Gambar 9 Grafik Waktu Penyimpanan Relatif Terhadap Jumlah Pesan



Gambar 10 Grafik Waktu Penyimpanan Relatif Terhadap Jumlah Kata Kunci

C. Perbandingan Kecepatan Pencarian

Tabel 3 memperlihatkan hasil pengujian waktu pencarian pesan rata-rata ditinjau dari aspek jumlah pesan dan jumlah kata

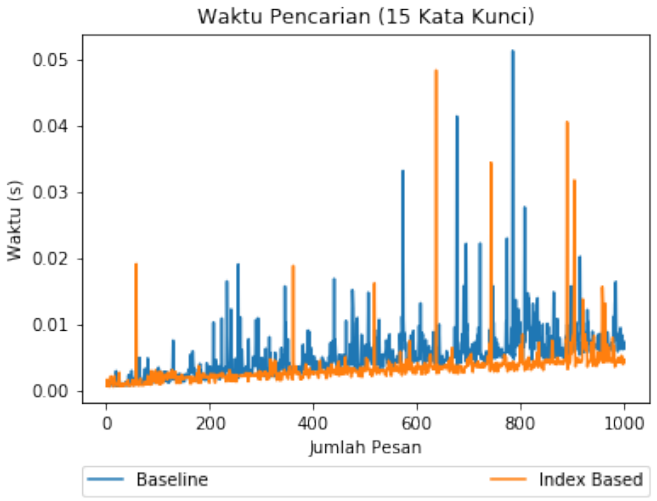
kunci. Terlihat bahwa dari aspek jumlah pesan, pencarian dengan *inverted indexing* mampu mencari dua kali lebih cepat dibandingkan teknik *baseline.* Dari aspek jamlah kata kunci, kecepatan yang dihasilkan hampir empat kali lebih cepat dibandingkan dengan teknik *baseline.*

Gambar 11 dan Gambar 12 memperlihatkan perbedaan waktu antara pencarian tanpa dengan pencarian kata berbasis indeks dan teknik *baseline*. Garis berwarna biru menunjukkan pencarian dengan teknik *baseline*, sedangkan garis berwarna jingga adalah pencarian menggunakan indeks kata. Pada Gambar 11, terlihat bahwa terdapat sedikit perbedaan antara penyimpanan menggunakan indeks kata dan tanpa indeks kata dari segi beban jumlah pesan. Namun, terlihat bahwa gradien dari garis berwarna jingga lebih rendah dibandingkan garis berwarna biru. Hal ini menunjukkan waktu pencarian ditinjau dari jumlah pesan lebih cepat ketika menggunakan teknik pencarian berbasis indeks kata.

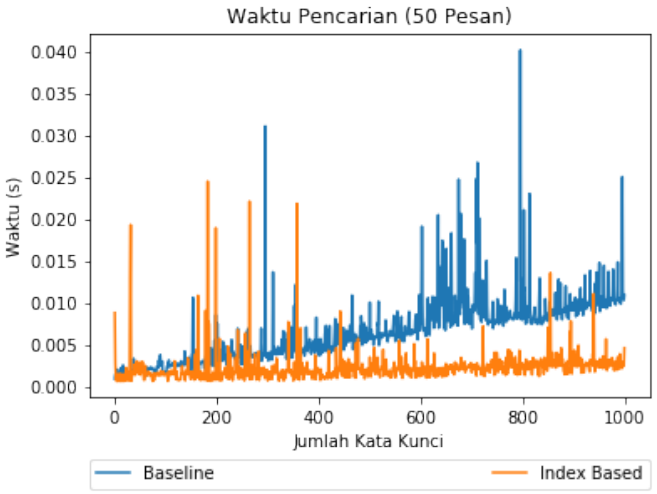
Ketika ditinjau dari jumlah kata kunci, Gambar 12 memperlihatkan hasil yang cukup berbeda dibandigkan dengan Gambar 11. Garis berwarna jingga memiliki gradien hampir bernilai nol. Hal ini menunjukkan untuk mencari pesan-pesan dengan suatu kata kunci, kecepatan tidak dipengaruhi oleh seberapa banyak sebuah pesan memiliki kata kunci. Sebaliknya, pencarian dengan menggunakan teknik *baseline* menunjukkan kenaikan yang cukup besar. Sebagai contoh, jika sebuah pesan memiliki 10.000 kata kunci, dan kata kunci yang harus dicari adalah kata kunci yang terakhir, maka *string matching* akan terus mencari hingga kasus terburuk. Waktu yang diperlukan menjadi cukup lama.

Tabel 3 Hasil Pengujian Waktu Pencarian Pesan

No	Aspek	Nilai	Waktu Rata-rata
1	Jumlah Pesan	Baseline	0.005 s
2		Index Based	0.003 s
3	Jumlah Kata Kunci	Baseline	0.007 s
4		Index Based	0.002 s



Gambar 11 Grafik Waktu Pencarian Relatif Terhadap Jumlah Pesan



Gambar 12 Grafik Waktu Pencarian Relatif Terhadap Jumlah Kata Kunci

V. ANALISIS

A. *Akses Pencarian Pesan*

Percapakan antara dua pihak memiliki kunci simetris yang unik. Kunci simetris atau *shared secret* tidak hanya digunakan untuk mendekripsi pesan agar dapat dibaca oleh pengguna. *Shared secret* juga digunakan untuk melakukan pencarian pesan. Oleh karena itu, walaupun peretas mampu mengakses basis data, peretas tidak akan mampu mengetahui maksud dari tabel indeks kata-kata yang telah dienkripsi. Dengan demikian, tidak mungkin ada peretas yang mampu mengetahui kata-kata kunci pada suatu pesan selama peretas tidak memiliki *shared secret* sebuah percakapan.

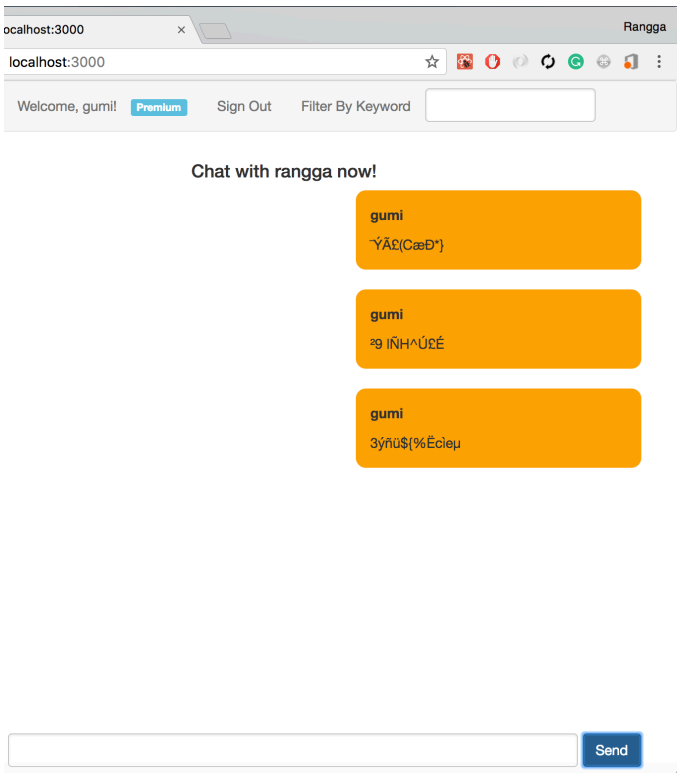
Untuk memperoleh *shared secret*, maka peretas harus memiliki kunci privat dari pengguna yang ingin diretas. Karena sistem ini menerapkan *end-to-end encryption*, kunci privat dibuat oleh pengguna pada aplikasi *client* dan tidak pernah melewati jaringan publik sama sekali. *Server backend* bahkan tidak pernah mengetahui bahkan menyimpan kunci privat pengguna. Oleh karena itu, cukup mustahil bagi peretas untuk mendapatkan kunci privat seseorang selain dengan cara membajak aplikasi *client* pengguna.

Tentunya, pengguna kemudian dapat melakukan *brute force* untuk menemukan *shared secret* yang mampu membuka pesan-pesan serta melakukan pencarian pesan. *Shared secret* yang dihasilkan nilainya bisa sangat beragam, namun untuk kunci algoritma AES, nilainya merupakan nilai pengubahan titik *shared secret* menjadi tipe string, kemudian mengambil 64 karakter pertama saja. Oleh karena itu, untuk menghasilkan *shared secret*, peretas harus menghitung 2^{64} kemungkinan kunci yang dapat dihasilkan.

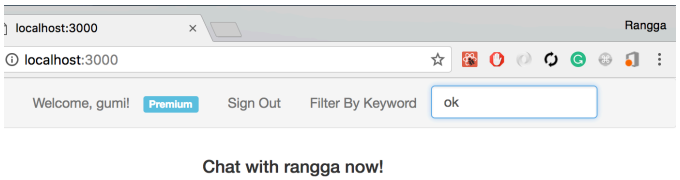
B. *Perubahan Sedikit Kunci*

Ketika kunci simetris suatu percakapan nilainya diubah, terlihat pada Gambar 13 bahwa isi pesan-pesan kemudian tidak dapat dibaca. Umumnya karakter-karakter yang dihasilkan bukan berupa huruf alfanumerik, melainkan huruf-huruf dengan nilai *encoding* yang tidak dapat dibaca oleh manusia. Kemudian, Gambar 14 memperlihatkan percobaan untuk melakukan

pencarian terhadap kata kunci "ok". Jika pada sebelumnya hasil pencarian mengembalikan pesan ketiga, operasi ini kini tidak mengembalikan pesan apapun.



Gambar 13 Hasil Dekripsi Pesan-Pesan dengan Kunci yang Tidak Tepat



Gambar 14 Pencarian Kata Dengan Kunci yang Tidak Tepat Sehingga Tidak Mengembalikan Hasil Apapun

Dari kedua hasil tersebut, dapat disimpulkan bahwa ketika kunci yang digunakan untuk mendekripsi ternyata salah, maka pesan yang dikembalikan tidak dapat dibaca dan pencarian kata kunci tidak dapat dilakukan.

C. *Penggunaan Kapasitas Memori*

Misalkan setiap kata kunci memiliki panjang 10 karakter dan setiap pesan memiliki 10 kata kunci. Dengan demikian, ketika menyimpan satu pesan, sistem membutuhkan kapasitas sebesar 100 bytes untuk menyimpan indeks kata kunci. Kapasitas tersebut tidak termasuk kapasitas dari pesan sendiri. Untuk kapasitas 1 GB, maka sistem dapat menyimpan 10 juta pesan dan 100 juta indeks kata kunci. Perbedaan 1 GB untuk jumlah pesan yang banyak tentu tidak menjadi *overhead* yang cukup besar.

VI. Simpulan dan Saran

Berdasarkan hasil eksperimen, pencarian kata dengan teknik *inverted indexing* menghasilkan kinerja yang lebih baik dari segi kecepatan pencarian kata kunci dibandingkan pencarian dengan melakukan dekripsi pesan satu per satu. Berdasarkan hasil analisis, terbukti bahwa penambahan penyimpanan tabel *indexing* tidak menambah *overhead* dan sulit diretas oleh orang tidak bertanggung jawab.

Untuk ke depannya, penelitian ini dapat dilengkapi dengan pemisahan kata kunci yang lebih baik dengan pembelajaran karena beberapa frasa mungkin juga penting untuk dijadikan kata kunci.

Ucapan Terima Kasih

Penulis memanjatkan rasa syukur kepada Tuhan YME karena atas kuasa-Nya penulis dapat menyelesaikan makalah ini. Penulis mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir selaku dosen mata kuliah Kriptografi karena atas bimbingannya penulis mendapatkan bekal ilmu yang cukup untuk menulis makalah ini. Terakhir, penulis juga mengucapkan terima kasih kepada pihak-pihak yang secara langsung maupun tidak langsung telah membantu menyelesaikan makalah ini.

Referensi

[1] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Advances in Acryptology – EURO-CRYPT'04*, volume 3027 of *Lecture Notes in Computer Science*, pages 506-522. Springer, 2004.

[2] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC `09. pages 169-178. New York, NY, USA. 2009. ACM.

[3] C. Wang, N. Cao, K. Ren, and W. Lou. Enabling secure and efficient ranked keyword search over outsourced cloud data. *IEEE Transaction on Parallel Distributed Systems*, 23(8):1467-1479, 2012.

[4] D. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *Proceeding of the IEEE Symposium on Security and Privacy `00*. 2000. pp. 44-55.

[5] Munir, Rinaldi. 2015. Slide Kuliah IF4020 Kriptografi: Pengantar Kriptografi.

[6] Daemen, Joan; Rijmen, Vincent. 2003. *AES Proposal: Rijndael*. National Institute of Standards and Technology. p. 1.

[7] Stallings, William. 2014. Cryptography and Network Security (6th ed.). Upper Saddle River, N.J.: Prentic Hall.

[8] U.S. National Security Agency. *Commercial National Security Algorithm Suite and Quantum Computing FAQ*. January 2016.

[9] Merkle, Ralph C. 1978. Secure Communications Over Insecure Channels. *Communications of the ACM*. 21 (4): 294–299.

[10] Chris Alexander, Ian Avrum Goldberg. 2007. Improved user authentication in off-the-record messaging. *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society*. New York: Association for Computing Machinery: 41–47.

[11] Marlinspike, Moxie. 2016. WhatsApp's signal protocol integration is now complete. Open Whisper Systems.

PERNYATAAN

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

Bandung, 15 Mei 2018

Garmastewira (13514068)