Rancangan Skema Pengiriman Pesan Berbasis Gabungan *One Time Pad* dan Kriptografi Kunci Publik

Cendhika Imantoro - 13514037
Program Studi Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514037@std.stei.itb.ac.id

***Abstract*—One Time Pad merupakan teknologi kriptografi yang sudah cukup tua. Teknologi ini sangat aman namun dirasa tidak *feasible*. Saat ini, teknologi kriptografi yang sering digunakan adalah kriptografi kunci publik. Kedua teknologi yang baru disebutkan memiliki kelemahan masing-masing. Penulis merasa menggabungkan kedua teknologi tersebut memungkinkan untuk menutupi kelemahan keduanya. Makalah ini membahas tentang skema pertukaran pesan yang mengoptimasi keamanan dengan menggabungkan kedua teknologi tersebut.**

***Keywords*—One Time Pad, Public Key Cryptography, Brute Force Attack, Secret Sharing**

I. PENDAHULUAN

Dengan berkembangnya teknologi informasi, jenis data yang dikirim dalam jaringan pun bertambah. Di antara data-data yang dikirim melalui jaringan, ada beberapa jenis data yang bersifat sensitif. Data yang bersifat sensitif tersebut dapat merugikan berbagai pihak jika jatuh ke tangan orang yang tidak bertanggung jawab. Padahal, data yang dikirim melalui jaringan cenderung mudah untuk ditangkap oleh pihak ketiga. Oleh karena itu, teknologi kriptografi untuk jaringan terus dikembangkan. Teknologi kriptografi mencegah terbacanya isi dari data yang dikirim dalam jaringan, meskipun pihak ketiga tersebut berhasil menangkapnya.

Berbagai pengembangan teknologi kriptografi sudah dibentuk. Salah satunya adalah *One Time Pad*. Teknologi ini berbasiskan penggunaan kunci yang bersifat acak. Dari segi keamanan, teknologi ini dinilai cukup aman karena pada teknologi ini, penggunaan kunci berbeda dapat menghasilkan dua pesan berbeda yang sama-sama memiliki makna. Hal ini menyebabkan penyerang tidak dapat mengetahui hasil dekripsi mana yang benar. Namun, teknologi ini tidak cukup *feasible* untuk digunakan karena semua pihak yang saling pesan harus sepakat dengan kunci yang sama, padahal kunci tersebut bersifat acak.

Teknologi kriptografi lain yang telah dibentuk adalah kriptografi kunci publik. Pada teknologi ini, semua pihak yang berkomunikasi bersifat memiliki dua kunci, yaitu kunci privat yang hanya diketahui pemilik dan kunci publik yang boleh diketahui umum. Kunci tersebut dibangun sedemikian sehingga sangat sulit bagi pemilik kunci publik untuk menurunkan nilai kunci privat dari kunci publik tersebut. Namun, dengan perangkat dan waktu yang cukup, hal ini memungkinkan untuk dilakukan. Masalah dari kriptografi kunci publik adalah sering kali, pesan menghasilkan makna hanya jia didekripsi dengan kunci yang tepat. Hal ini menyebabkan kriptografi kunci publik masih rentan terhadap *brute force attack*. Meskipun demikian, saat ini teknologi kriptografi kunci publik sering digunakan karena *brute force attack* pada pesan membutuhkan waktu dan *resource* yang signifikan.

Setelah mempelajari sifat dari *One Time Pad* dan kriptografi kunci publik, penulis merasa keduanya dapat menutupi kelemahan satu sama lain. Dalam makalah ini akan dibahas rancangan skema pertukaran pesan yang memanfaatkan kedua teknologi tersebut untuk menutupi kelemahan satu sama lain.

II. DASAR TEORI

A. One Time Pad

One Time Pad merupakan metode enkripsi yang menggunakan rangkaian karakter random dengan panjang

sama atau lebih dari panjang pesan. Untuk penggunaannya. One Time Pad harus dibagi dengan partisipan komunikasi lainnya. Algoritma untuk enkripsi tidak perlu kompleks. Yang paling penting adalah isi dari One Time Pad bersifat acak.

Hasil enkripsi menggunakan One Time Pad termasuk jenis enkripsi yang tidak bisa dipecahkan. Hal ini disebabkan karena tanpa mengetahui One Time Pad asli, penyerang harus mencoba berbagai One Time Pad. Cara penyerang mengetahui dekripsi berhasil atau tidak adalah dengan melihat apakah hasil dekripsi bisa dipahami atau tidak. One Time Pad yang salah dapat menghasilkan plainteks yang bermakna. Hal ini menyebabkan penyerang tidak mempunyai cara untuk mengetahui mana plainteks yang benar dan mana yang salah. Hal inilah yang menyebabkan One Time Pad tidak bisa dipecahkan.

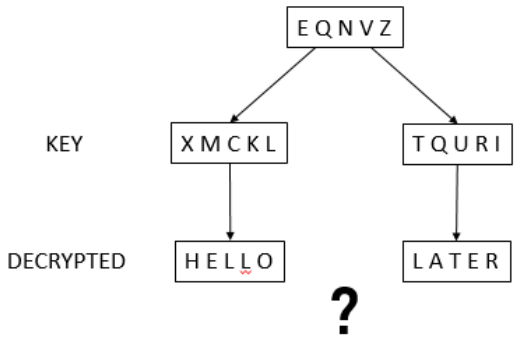

Figure 1. Confusion in OTP attack

Meskipun tidak bisa dipecahkan, One Time Pad dirasa tidak *feasible* untuk digunakan karena pembagian One Time Pad dengan peserta komunikasi lain sulit.

B. Public Key Cryptography

Public Key Cryptography merupakan skema kriptografi yang melibatkan dua buah kunci untuk tiap entitas. Kedua kunci tersebut disebut kunci privat dan kunci publik. Kunci privat bersifat rahasia dan hanya diketahui pemilik. Kunci publik bersifat tidak rahasia dan dapat diketahui siapapun. Pasangan kunci tersebut dibangun dalam satu proses komputasi. Kedua kunci ini dibangun sedemikian sehingga kunci privat sulit dibangun dari kunci publik. 'Sulit' yang dimaksud disini adalah bisa, namun membutuhkan waktu dan resource yang besar.

Dalam pertukaran pesan, plainteks dienkripsi dengan kunci publik. Algoritma enkripsi didesain sedemikian sehingga melakukan dekripsi menggunakan kunci publik membutuhkan waktu dan resource yang besar, sehingga satu-satunya cara untuk dekripsi dengan cepat adalah menggunakan kunci privat. Pesan yang telah dienkripsi lalu dikirim ke pemilik kunci publik (yang juga pemilik kunci privat). Pemilik tersebut menggunakan kunci privatnya untuk mendekripsi pesan dan membaca isinya.

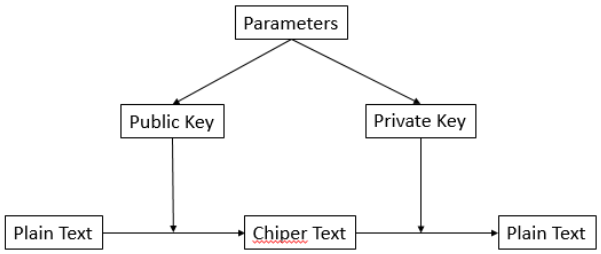

Figure 2. Public-private key encryption schema

Penggunaan kriptografi kunci publik yang lain adalah untuk pembagian rahasia. Misal Alice memiliki kunci privat K_A, dan Bob memiliki kunci privat K_B. Untuk pembagian rahasia, pertama dibutuhkan sebuah fungsi dua parameter f(a,b) sedemikian sehingga f(f(a,b), c) = f(f(a,c), b). Fungsi f(a,b) juga harus bersifat nilaib sulit diturunkan dari nilai a dan f(a,b) Selain itu diperlukan juga bilangan bebas K yang bersifat tidak rahasia. Yang dilakukan Alice adalah mengirim f(K, K_A) ke Bob. Bob mengirim f(K, K_B) ke Alice. Lalu Alice menghitung kunci rahasia K_{secret} = f(f(K, K_B), K_A) dan Bob menghitung kunci rahasia K_{secret} = f(f(K, K_A), K_B). Dengan demikian, karena f(f(K, K_B), K_A) = f(f(K, K_A), K_B), Alice dan Bob memiliki *shared secret* yang sama tanpa mengirimkan *shared secret* tersebut sama sekali.

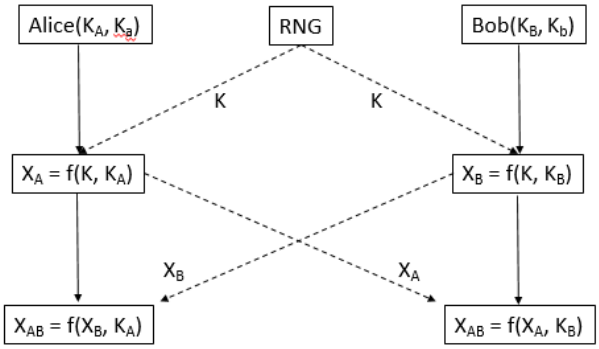

Figure 3. Public-private key secret sharing schema

Dalam prakteknya, kriptografi kunci publik jarang dipakai untuk mengenkripsi pesan karena membutuhkan waktu lama dan menghasilkan plainteks yang panjang. Biasanya, kriptografi kunci publik digunakan untuk *secret sharing* dan hasil *secret sharing* tersebut digunakan sebagai kunci dari algoritma kriptografi simetri untuk mengenkripsi pesan.

C. Cryptography Attack

Serangan terhadap kriptografi yang palin sulit dihindari adalah *brute force attack*. Serangan ini pada dasarnya adalah mencoba semua kemungkinan hingga memperoleh hasil yang benar. Hasil yang benar dinilai dari bisa dibaca

atau tidaknya hasil dekripsi.

Serangan lain yang mungkin dilakukan terhadap pesan terenkripsi adalah *repeat attack*. Serangan ini tidak berusaha membuka isi dari pesan. Melainkan mencoba untuk mengirim ulang pesan yang pernah dikirm dengan harapan akan menghasilkan dampak yang merugikan terhadap yang diserang.

III. Rancangan Skema Penggunaan OTP

A. Dasar Pemikiran

Nilai keamanan dari One Time Pad muncul karena teknik enkripsi yang dilakukan adalah *monoalpabhetic chiper* dengan kunci yang panjangnya sama dengan pesan dan bersifat acak. Hasil enkripsi dengan cara ini hanya bisa didekripsi dengan benar jika kunci yang digunakan tepat. Jika pihak yang tidak tahu kuncinya mencobamendekripsi dengan *brute force attack*, dia akan memperoleh banyak pesan yan semuanya memiliki makna. Penyerang tersebut tidak dapat mengetahui pesan mana yang benar di antara pesan-pesan bermakna tersebut.

Jenis pesan lain yang dirasa aman oleh penulis adalah pesan yang merupakan hasil enkripsi dari plainteks bersifat acak. Karena plainteksnya bersifat acak dan tidak bermakna, penyerang yang menggunakan *brute force attack* tidak dapat membedakan mana hasil dekripsi yang benar dan mana yang salah. Hal ini dapat dimanfaatkan untuk pengiriman One Time Pad, karena One Time Pad bersifat acak.

Dengan merancang skema pengiriman pesan yang hanya mengirim dua jenis pesan tersebut, nilai keamanan dari sistem pengiriman pesan akan cukup meningkat. Skema yang dirancang penulis didasarkan pada hal tersebut.

B. Skema Pengiriman Pesan

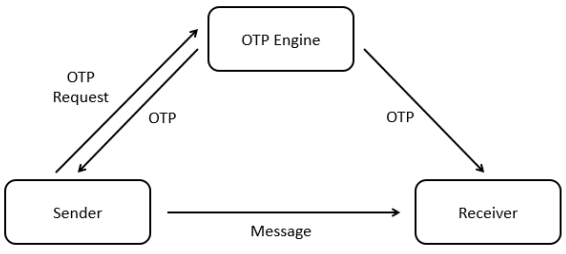

Figure 4. OTP-Public Key communication schema

Skema yang dirancang memiliki tiga aktor utama, yaitu *sender*, *receiver*, dan OTP Engine.

OTP Engine berfungsi untuk membangun OTP. Pengiriman pesan yang dilakukan OTP Engine hanya dua, yaitu pengiriman OTP ke pihak *sender* dan pengiriman OTP ke pihak *receiver*. Keduanya dilakukan dalam kondisi terenkripsi. Enkripsi dilakukan menggunakan kunci hasil *secret sharing* kriptografi kunci publik. OTP Engine hanya menerima satu jenis pesan, yaitu permohonan pembangunan OTP dari *sender*.

Sender dapat mengirim dua jenis pesan. Pesan pertama adalah permohonan pembangunan OTP. Pesan ini dikirim ke OTP Engine. Pesan kedua adalah pesan aktual yang dikirim ke *receiver*. Pesan ini dikirim setelah dienkripsi dengan OTP. *Sender* hanya menerima satu jenis pesan, yaitu OTP yang diperoleh dari OTP Engine.

Receiver dapat menerima dua jenis pesan. Pesan pertama merupakan OTP yang diperoleh dari OTP Engine. Pesan kedua merupakan pesan aktual yang diterima dari *sender*. Pesan yang diterima dari *sender* didekripsi menggunakan OTP yang diterima dari OTP Engine. Dalam satu proses pengiriman pesan pada sistem, r*eceiver* tidak melakukan pengiriman pesan apapun.

C. Alur Kerja Umum

Sistem memiliki dua tahap kerja. Tahap pertama adalah tahap *handshake*. Tahap ini cukup dilakukan sekali di awal penggunaan sistem. Pada tahap ini semua aktor membangun *shared secret* dengan tiap aktor lainnya. Pembangunan *shared secret* dilakukan dengan menggunakan skema *secret sharing* pada kriptografi kunci publik. Setelah tahap ini selesai, kondisinya adalah:

- OTP Engine dan *sender* memiliki *shared secret* K_A.
- OTP Engine dan *receiver* memiliki *shared secret* K_B.
- *Sender* dan *receiver* memiliki *shared secret* K_{AB}.

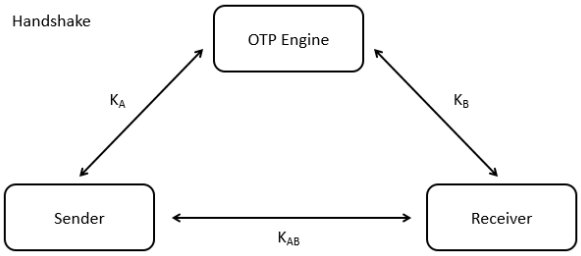

Figure 5. Handshake phase

Tahap selanjutnya adalah pengiriman pesan. Tahap ini dapat dilakukan berulang-ulang. Tahap ini terdiri dari beberapa langkah.

Langkah pertama, *sender* melakukan *request* kepada OTP Engine untuk membangun OTP. *Request* ini dienkripsi dengan menggunakan kunci K_A.

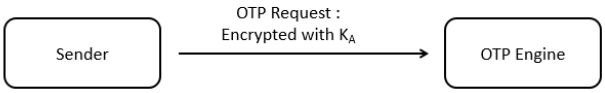

Figure 6. Requesting OTP

Langkah kedua, OTP Engine mendekripsi *request* yang diterima menggunakan kunci K_A. Dari hasil dekripsi, OTP Engine dapat menentukan panjang pesan yang dibutuhkan. OTP Engine lalu membangun rangkaian *byte pseudo-random* sesuai dengan panjang yang diminta. Rangkaian *byte* ini merupakan sebuah OTP.

Selanjutnya, OTP dikirimkan ke *sender* dan *receiver*. OTP yang dikirim ke *sender* dienkripsi dengan kunci K_A. OTP yang dikirim ke *receiver* dienkripsi dengan kunci K_B.

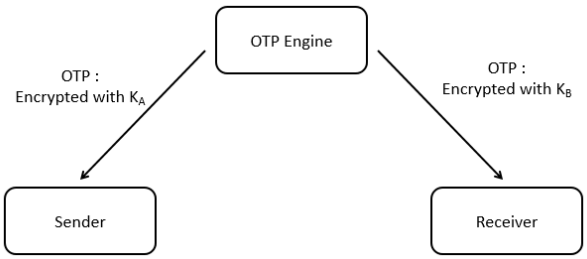

Figure 7. OTP Multicast

Setelah menerima pesan dari OTP Engine, *receiver* dan *sender* mendekripsi pesan yang diterima dengan menggunakan *shared secret* masing-masing. *Sender* mendekripsi dengan kunci K_A, *receiver* mendekripsi dengan kunci K_B. Pada titik ini, *receiver* dan *sender* memiliki OTP yang sama.

Selanjutnya, *sender* mempersiapkan pesan yang akan dikirim ke *receiver*. *Sender* mengenkripsi pesan plainteks dengan menggunakan OTP. Pesan ini lalu dikirim ke *receiver*.

Receiver menerima pesan dari *sender*. *Receiver* lalu mendekripsi pesan tersebut dengan menggunakan OTP. Pada titik ini, tujuan proses pengiriman tercapai, yaitu *receiver* berhasil menerima pesan dari *sender*. *Receiver* lalu menghapus atau meng-*invalidate* OTP yang baru digunakan

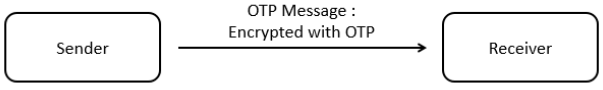

Figure 8. Sending message

D. Struktur Dasar Pesan

Pada skema ini, terdapat tiga macam pesan. Pesan yang dimaksud adalah *request* OTP dari *sender*, OTP dari OTP Engine, dan pesan sesungguhnya dari *sender* ke *receiver*.

Agar OTP Engine dapat membangun OTP dengan panjang secukupnya, maka di dalam *request* perlu ada informasi panjang OTP. Lalu, untuk mencegah *repeat attack*, perlu ada kode yang hanya dapat dipakai sekali. Kode yang tepat untuk ini adalah *hash* dari OTP terakhir yang diterima *sender*. Setelah *request* ini diproses, *sender* akan menerima OTP baru, sehingga kode ini akan berubah. Oleh karena itu, *request* OTP harus memuat panjang OTP dan *hash* OTP sebelumnya.

OTP Request	
Hash of previous OTP	For ‘repeat attack’ prevention
Requested OTP length	Minimum size of generated OTP

Table 1. OTP request format

Pada pesan OTP, tentunya harus memuat OTP. Selain itu, sama seperti *request* OTP, pesan OTP juga perlu mekanisme untuk menangani *repeat attack*. Pesan OTP juga harus memuat kode unik sekali pakai. Kode yang sama bisa digunakan, yaitu *hash* dari OTP terakhir yang telah dikirim ke *sender* dan *receiver*. Oleh karena tu, pesan OTP perlu memuat OTP dan *hash* OTP terakhir.

OTP	
Hash of previous OTP	For ‘repeat attack’ prevention
New OTP	OTP for next message

Table 2. OTP format

Pada pesan sesungguhnya, yang harus dimuat adalah isi dari pesan saja. Penanganan *repeat attack* pada bagian ini ditangani dengan penghapusan OTP yang sudah terpakai oleh *receiver*.

Message	
Content	Content of the message

Table 3. Message format

E. Scalability

Pada skema yang telah dijelaskan, komunikasi yang dilakukan hanya melibatkan dua pihak. Tentunya akan muncul pertanyaan bagaimana satu OTP Engine dapat menangani lebih dari dua pihak. Misal ada 10 ‘client’ dan

tiap dua ‘client’ dapat melakukan komunikasi. Untuk menangani hal ini, perlu ada beberapa modifikasi, terutama di bagian struktur pesan.

Dalam kasus satu OTP menangani lebih dari dua ‘client’, tiap *request* OTP harus memuat juga tujuan dari pesan yang akan dikirim dengan OTP tersebut. Hal ini diperlukan karena ketika ada lebih dari dua ‘client’, kemungkinan tujuan lebih dari satu. Begitu juga dengan pesan OTP. Dalam pesan tersebut, perlu dimuat juga OTP itu untuk komunikasi dari siapa dan ke siapa. Dengan informasi ini, jika *sender* punya beberapa pesan yang akan dikirimkan ke tujuan berbeda, *sender* bisa menentukan pesan mana yang harus di enkripsi dengan OTP yang diterima.

OTP Request	
Hash of previous OTP	For ‘repeat attack’ prevention
Requested OTP length	Minimum size of generated OTP
Receiver	Message receiver identity

Table 4. Modified OTP request format

OTP	
Hash of previous OTP	For ‘repeat attack’ prevention
New OTP	OTP for next message
Sender	Message sender identity
Receiver	Message receiver identity

Table 5. Modified OTP format

F. Additional Security

Pada skema yang telah dijelaskan, K_{AB} belum digunakan. *Shared secret* ini dapat digunakan untuk meningkatkan keamanan. Caranya adalah, OTP yang diterima oleh *receiver* dan *sender* diacak terlebih dahulu sebelum digunakan untuk enkripsi dan dekripsi. Dengan demikian, penyerang tidak bisa melakukan *brute force attack* dengan mencoba-coba pada pesan OTP dan langsung menggunakan hasilnya pada pesan sesungguhnya. Penyerang juga harus mempertimbangkan permutasi OTP sehingga kompleksitas bertambah signifikan.

Tambahan keamanan juga dapat diimplementasikan pada panjang OTP yang dibangun OTP Engine. Jika panjang OTP Engine tepat sesuai dengan permintaan *sender* (misal panjang yang diminta N), hasil dekripsi juga akan memiliki panjang N. Maka, penyerang bisa membuat hipotesis ‘hasil dekripsi dari *request* yang dikirim *sender* mengandung bilangan N’. Hal ini akan membantu penyerang untuk mencari hasil dekripsi yang tepat.

G. Customization

Pada skema ini, ada beberapa hal yang bisa dikustomisasi sesuai keinginan pengguna, yaitu:

- Algoritma kriptografi kunci publik
- Algoritma *secret sharing*
- Algoritma kriptografi simetri
- Algoritma enkripsi OTP
- Algoritma pseudo-random OTP
- Algoritma pengacakan OTP
- Algoritma *hash* OTP

IV. Analisis Keamanan

Analisis dilakukan berdasarkan kemungkinan informasi yang diperoleh penyerang melalui jaringan dan kemungkinan tindakan yang dilakukan pada informasi tersebut.

A. Serangan 1

Penyerang memulai serangan dari pesan *request* OTP terenkripsi yang dikirim dari *sender*. pesan ini mengandung *hash* OTP sebelumnya dan panjang OTP yang diminta. *Hash* memiliki *fixed length* sehingga penyerang dapat mem-parsing bagian pangjang OTP dari hasil dekripsi *request.* Dengan membandingkan *length* pada *request* dan panjang OTP balasan dari OTP Engine, penyerang dapat memperkirakan mana hasil dekripsi yang mungkin benar.

Dengan menggunakan algoritma kunci simetris yang kuat, penyerang akan menggunakan *brute force*. Hal ini pun masih menghasilkan beberapa kemungkinan hasil dekripsi. Untuk tiap kemungkinan hasil dekripsi, yang dilakukan penyerang :

- Perkiraan kunci simetri antara *sender* dan OTP Engine diperoleh penyerang.
- Kunci tersebut dapat digunakan untuk mendekripsi OTP yang dikirim OTP Engine ke *sender*. Penyerang memperoleh OTP.
- OTP yang diperoleh dapat digunakan untuk melakukan *known plaintext attack* pada OTP

yang dikirim ke *receiver*. Kemungkinan penyerang memperoleh kunci simetri antara OTP Engine dan *receiver* meningkat.
- Jika *sender* tidak melakukan pengacakan pada OTP, pada titik ini, penyerang dapat mendekripsi pesan dari *sender* ke *receiver*.
- Jika *sender* melakukan pengacakan, untuk mendekripsi, penyerang butuh kunci simetri *sender-receiver*. Kunci ini tidak bisa dibangun dari informasi yang telah diperoleh.
- Penyerang harus melakukan *brute force* pengacakan OTP untuk memperoleh plainteks

Dari hal tersebut, disimpulkan jika menggunakan algoritma kunci simetri yang kuat, *brute force* pada *request* OTP sulit. Dan meskipun berhasil dilakukan, penyerang masih haris melakukan *brute force* terhadap permutasi OTP.

B. Serangan 2

Penyerang memperoleh pesan OTP yang dikirim OTP Engine ke *sender*. Pesan tersebut berisi *hash* OTP sebelumnya dan OTP baru, semuanya terenkripsi. Semua konten tidak memiliki makna, sehingga penyerang tidak bisa mengetahui mana hasil dekripsi yang benar dan mana yang salah. Namun, *hash* memiliki *fixed length*, sehingga untuk semua kemungkinan hasil dekripsi, penyerang bisa mengambil bagian *hash* dan bagian OTP. Dari hal ini yang mungkin dilakukan penyerang:

- Melakukan serangan *brute force* ke pesan OTP yang dikirim ke *receiver*.
- Memperoleh kemungkinan OTP yang dikirim ke *receiver.*
- Membandingkan kemungkinan OTP yang dikirim ke *sender* dan kemungkinan OTP yang dikirim ke *receiver*.
- Memperoleh OTP yang sama (mungkin ada banyak)
- Memperoleh kemungkinan kunci simetris OTP Engine-*sender*
- Memperoleh kemungkinan kunci simetris OTP Engine-*receiver*
- Karena OTP diacak di *sender* dan *receiver* menggunakan kunci lain, informasi tidak cukup untuk menentukan permutasi yang digunakan
- Melakukan *brute force* permutasi tiap kemungkinan OTP pesan *sender-receiver*.

Untuk menyerang melalui pesan OTP, perlu melakukan tiga lapis *brute force*, dan dua *brute force* pertama masih menghasilkan banyak kemungkinan.

C. Serangan 3

Penyerang memperoleh pesan yang dikirim dari *sender* ke *receiver*. Karena enkripsi dilakukan menggunakan OTP, serangan *brute force*yang dilakukan penyerang tidak dapat membantu.

V. KESIMPULAN

OTP dan Public Key Cryptography dapat menutupi kelemahan satu sama lain. Masalah terkait sulitnya membagi OTP ke partisipan komunikasi dapat ditangani dengan menggunakan *secret sharing* dan enkripsi *public key cryptography*. Masalah serangan *brute-force* pada *public key cryptography* dapat ditangani dengan hanya digunakan untuk menangani pesan tak bermakna seperti OTP, sehingga sulit di dekripsi.

REFERENCES

[1] "The only unbreakable cryptosystem known", http://www.pro-technix.com/information/crypto/pages/vernam_base.html
[2] W. Stallings, "*Cryptography and Network Security: Principle and Practices"*. Prentice Hall.

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, 18 Mei 2018

Cendhika Imantoro - 13514037