Penggunaan *Artificial Neural Network* pada Kriptografi Kunci Simetri

Novan Parmonangan Simanjuntak (13509034)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13509034@stei.itb.ac.id

***Abstract*— Algoritma kriptografi sudah berkembang pesat saat ini, secara umum algoritma enkripsi dibagi menjadi dua yaitu enkripsi simetri dan asimetri. Adapun ilmu yang digunakan dalam merancang kebanyakan algoritma enkripsi/dekripsi atau *cryptosystem* yang ada adalah dengan menggunakan teori bilangan. Di sini penulis mengajukan metode lain dalam membentuk *cryptosystem*, yaitu dengan menggunakan Intelejensia Buatan. Beberapa penelitian sudah memanfaatkan ilmu Intelejensia Buatan antara lain algoritma genetik dan *Artificial Neural Network* (ANN). Pendekatan pembuatan *cryptosystem* dengan menggunakan ANN merupakan pendekatan yang baru dipakai. Penelitian membuktikan bahwa *cryptosystem* yang dibuat dengan menggunakan ANN memberikan hasil yang baik dibandingkan metode yang lainnya, baik dari segi kompleksitas dan ketahanan *cryptosystem*. Hal ini dikarenakan ANN mampu memberikan model yang dapat memetakan secara non-linear dari beberapa variabel input ke beberapa variabel output dengan kompleksitas yang rendah (sesuai dengan jumlah node awal). Penyerang sangat kesulitan dalam melakukan dekripsi karena harus tahu dulu mengenai topologi ANN yang ada (struktur jaringan saraf, bobot serta fungsi-fungsi yang digunakan). Pada makalah ini penulis akan membentuk**

***Index Terms*—ANN, simetri, asimetri, Intelejensia Buatan *cryptosystem*.**

I. PENDAHULUAN

Kriptografi merupakan seni mengamankan dengan membuat pesan menjadi tidak bisa dibaca. Penggunaan kriptografi pertama kali adalah pada 1900 B.C ketika orang Mesir menggunakan huruf hieroglyphs tidak standar. Sampai saat ini pun kriptografi masih berkembang serta diaplikasikan pada kehidupan sehari-hari seperti pada kartu ATM, *password* komputer dan *electronic commerce*. Di era modern ini kriptografi dipertimbamgkan sebagai bagian cabang dari matematika dan teknik komputer.

Secara umum ada tiga tipe skema kriptografi, yaitu kriptografi kunci simetri (disebut juga kunci rahasia),kriptografi kunci asimetri (disebut juga kunci publik), dan fungsi hash. Secara singkat kriptografi kunci simetri menggunakan satu kunci untuk proses enkripsi dan dekripsi, sedangkan kriptografi kunci public menggunakan dua kunci, yaitu kunci public yang bisa diketahui oleh pihak lain sedangkan kunci privat tidak diketahui pihak lain. Fungsi hash disebut juga *message* digest atau enrkipsi satu arah adalah algoritma yang tidak menggunakan kunci sama sekali,tetapi menggunakan nilai hash dengan panjang tetap yang dihitung berdasarkan plainteks sehingga memungkinkan untuk memperoleh kembali isi atau panjang dari plainteks[5]. Adapun pendekatan yang ada saat sekarang memanfaatkan teori bilangan. Terdapat metode lain yang sifatnya atau fenomenanya dapat digunakan untuk kriptografi, yaitu *Artificial Neural Network*. Pada makalah ini penulis membahasa mengenai penggunaan *Artificial Neural Network* pada kriptografi kunci simetri. Berbagai settingan pada jaringan saraf dapat mempengaruhi kinerja ditinjau dari segi kompleksitas dan ketahanan.

II. DASAR TEORI

A. Algoritma kunci simetri

Pada algoritma kunci simetri, kedua pihak mempunyai kunci *k* yang sama baik untuk enkripsi dan dekripsi. Seperti yang ditunjukkan pada Gambar 1, pengirim pesan menggunakan kunci untuk mengenkripsi pesan dan mengirimkan enrkipsi yang sama dengan kunci dekripsi Untuk menjaga privasi, kunci *k* harus dijaga secara rahasia. Ketika seseorang mengetahui kunci, *k* maka algoritma ini sudah tidak aman lagi. Hal ini berbeda dengan algoritma kunci asimetri, di mana digunakan kunci publik dan kunci privat.

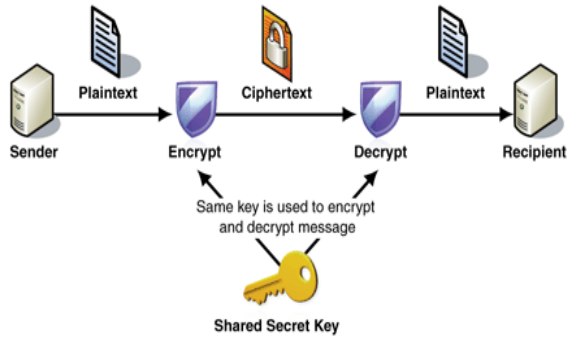
Meskipun begitu algoritma kunci simetri mempunyai keuntungan jika dibandingkan dengan algoritma kunci asimetri. Keuntungan tersebut antara lain:

1. Mengkonsumsi sedikit memori.
2. Menggunakan waktu sedikit untuk enkripsi dan dekripsi.

Pada dunia nyata kedua skema ini digunakan secara bersamaan. Karena mahal, skema kunci asimetri hanya digunakan untuk bertukar kunci simetri, sehingga kunci

simetri tidak diketahui dan proses yang lama hanya terjadi sekali, yaitu pada waktu bertukar kunci simetri.

Beberapa contoh algoritma kunci simetri adalah DES, 3DES, IDEA, BLOWFISH, TWOFISH dan CAST5.



Gambar 1. Skema kriptografi kunci simetri

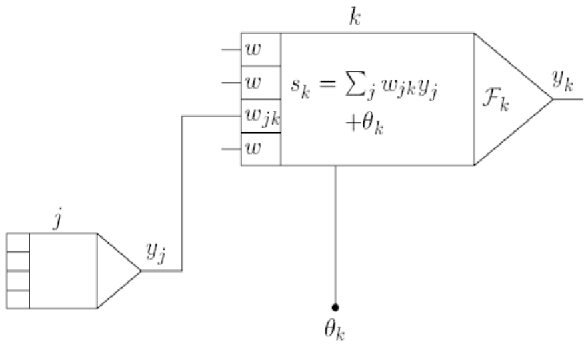
B. Artificial Neural Network

Artificial Neural Network merupakan model dari simulasi otak manusia, dalam hal ini neuron. Otak menggunakan *neuron* untuk melakukan perhitungan yang jauh lebih cepat daripada komputer digital tercepat saat ini. Selain itu otak sangatlah kompleks, bersifat *nonlinear* dan mampu memproses informasi secara pararel. Jaringan saraf tiruan dibuat untuk memodelkan otak melakukan suatu tugas.

Jaringan saraf tiruan sendiri terdiri dari banyak *neuron* yang saling berkomunikasi dengan mengirimkan sinyal satu denga lainnya melalui *link*/koneksi. Komponen dari jaringan saraf tiruan antara lain:

1. Sekumpulan neuron.
2. Nilai aktivasi y_k untuk setiap neuron, y_k adalah output dari neuron.
3. Koneksi atau *link* antar unit. Setiap koneksi mempunyai nilai bobot w_{jk} yang menentukan efek dari sinyal yang dikirimkan neuron j ke neuron k.
4. *Propagation rule*, yang menentukan nilai input total s_k dari semua input eksternal.
5. Fungsi aktivasi F_k yang ditentukan berdasarkan nilai sk(t) dan yk(t).
6. Offset untuk input eksternal, θk untuk setiap neuron.
7. Aturan learning untuk setiap pasangan input dan output. Aturan ini digunakan untuk menentukan nilai bobot pada *link*.

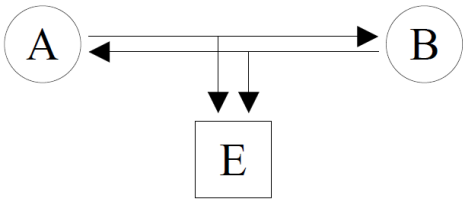
Gambar 2 menjelaskan komponen dari jaringan saraf tiruan.



Gambar 2. Komponen dari Jaringan Saraf Tiruan

C. Kriptografi kunci simetri dengan Artificial Neural Network

Permasalahan yang akan diselesaikan akan dimisalkan sebagai berikut. **A** dan **B** adalah teman dan ingin saling bertukar pesan melalui *public channel*. Karena melalui *public channel* dan **A** tidak ingin musuh (**E**) mengetahui isi pesan, maka **A** mengenkripsi pesan dengan algoritma enkripsi simetri kemudian memberikan pesan yang sudah dienkripsi ke **B**. Permasalahannya adalah bagaimana caranya **B** mengetahui kunci **A** agar dapat mendekripsi pesan. Hal ini sulit dilakukan dengan asumsi **E** adalah musuh pasif (maksud pasif dalam hal ini hanya mendengar pesan yang lewat antara **A** dan **B,** bukan mengubahnya). Permasalahan ini dijelaskan secara ringkas pada gambar 3.



Gambar 3. Situasi di mana A dan B ingin berbagi kunci rahasia di mana E merupakan penyerang pasif

Pada permasalahan ini, ada tiga solusi [6] yang bisa ditawarkan, antara lain :

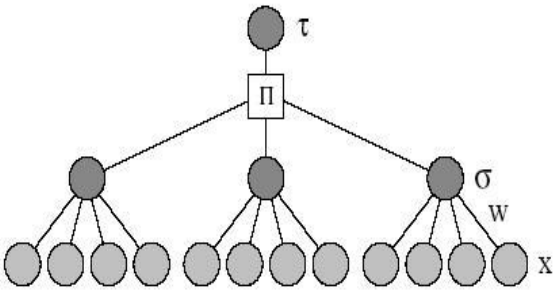
1. Menggunakan sebuah *private channel* untuk menstransimisikan kunci. Contohnya seperti **A** dan **B** mereka saling bertemu secara rahasia.
2. Menggunakan skema kriptografi kunci publik, di mana kunci publik **A** dan **B** bisa ditukar tanpa perlu khawatir penyerang pasif **C** mampu mengetahui isi pesan yang sesungguhnya.
3. Solusi ketiga adalah dengan cara **A** dan **B** berkomunikasi terus-menerus melalui *public channel* kemudian dari hasil komunikasi tersebut mereka bisa menghasilkan sebuah kunci rahasia berdasarkan pesan-pesan yang sudah dikomunikasikan. Di sini **E** tidak

dapat menentukan rahasia dengan hanya melalui komunikasi antara **A** dan **B**.

Solusi pertama sulit dilakukan sedangkan solusi kedua memakan biaya yang mahal, sedangkan solusi ketiga memerlukan sebuah protokol sedemikan sehingga **A** dan **B** dapat menghasilkan sebuah kunci rahasia *k* dengan menggunakan pesan yang dikomunikasikan lewat *public channel* dan E tidak bisa menghasilkan kunci rahasia *k* melalui pesan-pesan tersebut.

Protokol ini dapat direalisasikan dengan menggunakan sinkronisasi neural[2]. Sinkronisasi neural adalah tahap di mana dua jaringan saraf tiruan yang mempunyai topologi yang sama mempunyai vektor bobot yang sama. Vektor bobot inilah yang kemudian dijadikan kunci rahasia. Kedua jaringan saraf tiruan ini berbagi vektor input satu dengan lainnya (makanya disebut sebagai *mutual learning*). Hal ini dimungkinkan karena untuk *training* sebuah jaringan saraf tiruan bisa dengan online (menggunakan satu data saja).

Misalkan diberikan jaringan saraf tirun di bawah ini.



Gambar 4. Jaringan saraf tiruan[8]

Keterangan :

τ : output
Π : Prosedur aktivasi
σ : hidden layer
W : vektor input
X : vektor bobot

Jaringan saraf tiruan di atas dapat dideskripsikan dengan 3 buah parameter, yaitu :

1. K, jumlah *hidden neurons*.
2. N, jumlah *input neurons* yang tersambung ke tiap *hidden neuron*.
3. L, nilai maksimum dari bobot {-L..+L}.

Tahapan yang dilakukan dalam sinkronisasi neural adalah :

1. A dan B masing-masing diberikan sebuah jaringan saraf tiruan yang mempunyai topologi yang sama. Misalnya jaringannya seperti pada Gambar 4.
2. A dan B masing-masing menghasilkan bobot awal secara random. Inilah yang dirahasiakan dari E.
3. Kemudian dilakukan perulangan di mana mula-mula A dan B menghasilkan secara random dulu vektor input. Vektor ini ditukar melalui *public channel*.
4. A menggunakan vektor input dari B, begitu juga kebalikannya. Kemudian dari vektor bobot dan vektor input tersebut dihitung nilai output.
   Nilai output ditentukan oleh rumus :

$$\tau = \prod_{i=1}^{K} \text{SIGN}[\sum_{j=1}^{N} w_{i,j}\, x_{i,j}\,]$$

Gambar 5. Formula menghitung output

Pada formula di atas, fungsi SIGN diformulasikan sebagai berikut :

$$\operatorname{sgn}(x) = \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$

Gambar 5. Formula dari fungsi SIGN

5. Output ini dibagikan secara publik untuk dibandingkan. Jika tidak sama maka kembali ke langkah 3, jika sama maka *update* nilai vektor bobot.
6. Dengan sifat unik atau fenomena dari jaringan saraf tiruan ini, dijamin bahwa setelah sejumlah finite vektor input dibagikan antara kedua jaringan saraf tiruan, kedua jaringan saraf tiruan tersebut akan mempunyai vektor bobot yang sama. Vektor bobot inilah yang kemudian dijadikan kunci rahasia *k*.

Pada tahap 5 *update* nilai vektor bobot, dapat digunakan salah satu dari *3 learning rules berikut* :

- Hebbian learning rule:
$$w_i^+ = w_i + \sigma_i x_i \Theta(\sigma_i \tau) \Theta(\tau^A \tau^B)$$
- Anti-Hebbian learning rule:
$$w_i^+ = w_i - \sigma_i x_i \Theta(\sigma_i \tau) \Theta(\tau^A \tau^B)$$
- Random walk:
$$w_i^+ = w_i + x_i \Theta(\sigma_i \tau) \Theta(\tau^A \tau^B)$$

Fungsi Θ(a,b) bernilai 0 jika a ≠ b, 1 jika tidak.

D. Penggunaan struktur Tree Parity Machine

Perlu diperhatikan bahwa untuk jaringan saraf tiruan biasa, tidak ada perbedaan yang jelas antara interaksi satu arah atau dua arah. Oleh karena itulah digunakan *Tree Parity Machines* yang merupakan jaringan saraf tiruan yang lebih kompleks. Pada *Tree Parity machines* terdapat sifat khusus yang membuatnya dipakai, yaitu sinkronisasi dengan menggunakan *mutual learning* (saling learning dari jaringan saraf tiruan lainnya) jauh lebih cepat jika learning dengan hanya menggunakan examples dari jaringan lainnya (satu arah) [1-2, 7].

Hal ini diperlukan agar musuh E tidak dapat menandingi kecepatan sinkronisasi jika E ingin membentuk jaringan saraf tiruan yang sinkron dengan jaringan saraf tiruan milik A dan B. Jika E tidak dapat menandingi kecepatan sinkronisasi, maka A dan B sudah mendapatkan kunci rahasia terlebih dahulu dan bisa bertukar pesan serta menggunakan jaringan saraf tiruan baru untuk menghasilkan kunci rahasia *k* yang baru.

Perlu diperhatikan bahwa meskipun E mencoba membuat jaringan yang sama dengan A dan B, maka ia akan tetap tertinggal juga.Hal ini dibuktikan sebagai berikut. Misalkan E memperhatikan nilai output(A) dan output(B) di mana output(A) adalah nilai output yang dihitung dari jaringan saraf tiruan milik A dan output(B) adalah nilai output yang dihitung dari jaringan saraf tiruan milik B. Ada 3 kemungkinan :

1. E mengetahui Output(A) ≠ Output(B). Maka E tidak mengubah vektor bobotnya.

2. Jika Output(A) = Output(B) = Output(E), maka A, B, dan E mengubah vektor bobot mereka.

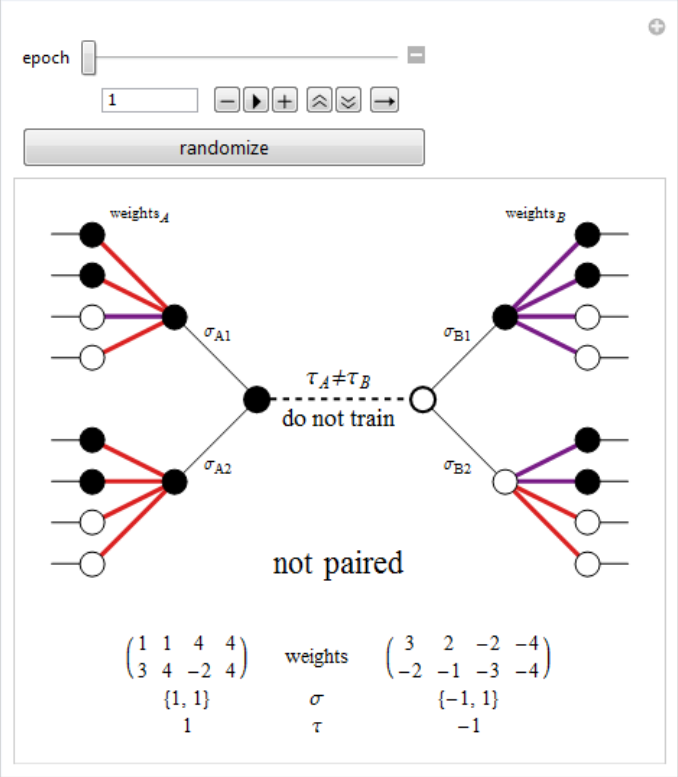
3. Jika Output(A) = Output(B) ≠ Output(E), maka A dan B mengubah vektor bobotnya, sedangkan E tidak. Karena ini maka E akan semakin tertinggal dalam learning.

Yang membedakan Tree Parity Machine dengan jaringan saraf tiruan biasa lainnya adalah strukturnya, di mana pada Tree Parity Machine terdapat K-*hidden neurons*, Y-*left dynamic hidden neurons*, dan Z-*right dynamic hidden neurons*. Makalah ini akan menggunakan Tree Parity Machine sebagai topologi dari jaringan saraf tiruan.

III. IMPLEMENTASI

A. Kakas dan File Uji

Implementasi dilakukan untuk melihat seberapa cepat sinkronisasi dilakukan untuk setiap vektor input yang random . Di sini dilakukan percobaan dengan menggunakan sebuah Tree Parity Machine sederhana. Adapun kakas dan bahasa pemrograman yang digunakan adalah Wolfram untuk membuat simulasi penggunakan jaringan saraf tiruan. Gambar 6 menunjukkan secara ringkas aplikasi yang digunakan untuk mensimulasikan. Secara sederhana, pengguna menekan tombol randomize untuk menghasilkan variable awal seperti vektor bobot, kemudian ditekan tombol play untuk memulai *mutual learning*. Epoch pada aplikasi maksudnya adalah jumlah pertukaran pesan yang sudah terjadi antara A dan B.



Gambar 6. Tampilan Aplikasi

Untuk sebuah data uji terdapat (perhatikan bahwa jumlah anggota sebuah elemen data uji ditentukan oleh topologi Tree Parity Machine yang akan dijelaskan di bagian selanjutnya):

- Pasangan vektor bobot dari A dan B. Ini adalah nilai random yang dijaga kerahasiaannya dari musuh. Nilai inilah yang turut menentukan . Perlu diperhatikan bahwa sebuah vektor bobot mempunyai 8 anggota karena terdapat 8 *hidden neurons*.
- Pasangan hidden-neurons.
- Pasangan nilai output.

Berikut 4 data uji yang nilainya degenerate secara random

id	$weights_A$	$weights_B$	σ_A	σ_B	τ_A	τ_B
1	(1 1 4 4 3 4 -2 4)	(3 2 -2 4 -2 -1 -3 -4)	{1, 1}	{-1, 1}	1	-1
2	(-2 2 0 -3	(3 -4 -1 -4	{-1, -1}	{1, -1}	1	-1

	-2 -3 3 4)	-3 1 4 4)				2	
3	(-1 -2 2 3 3 2 4 -3)	(4 -2 -3 -3 -1 4 3 -1)	{-1, 1}	{1, -1}	-1	-1	
4	(1 4 0 2 2 2 1 1)	(1 4 0 2 2 2 1 1)	{-1, -1}	{-1, -1}	1	1	

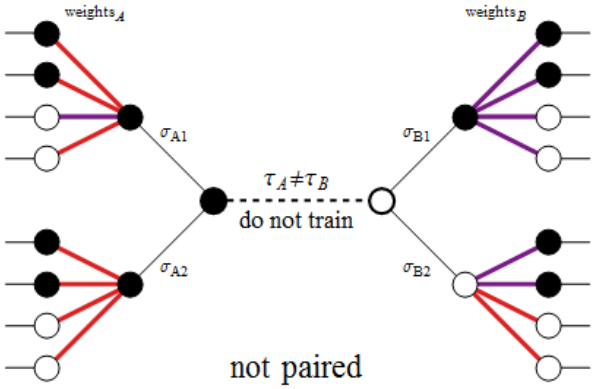
Tabel 1. Data uji

2	869	4.345
-1	99	0.495
4	283	1.415

Tabel 2. Hasil eksperimen

B. Metode Artificial Neural Network

Pada metode ini ditentukan terlebih dahulu Jaringan Saraf Tiruan yang digunakan. Pada makalah ini digunakan Tree Parity Machine yang digunakan mengandung 1-*hidden neurons*, 1-*left dynamic hidden neurons*, dan 1-*right dynamic hidden neurons*. Jumlah total *hidden neurons* ada 2, sedangkan jumlah elemen vektor input ada 8. Topologi sepenuhnya dapat dilihat di gambar 7, di mana ANN sebelah kiri adalah ANN untuk A, sedangkan yang kanan untuk B.



Gambar 7. Topologi Jaringan Saraf Tiruan yang digunakan

Setelah menentukan topologi, dijalankan algoritma sinkronisasi neural. Berikut pseudo-code dari algoritma yang digunakan dalam aplikasi :

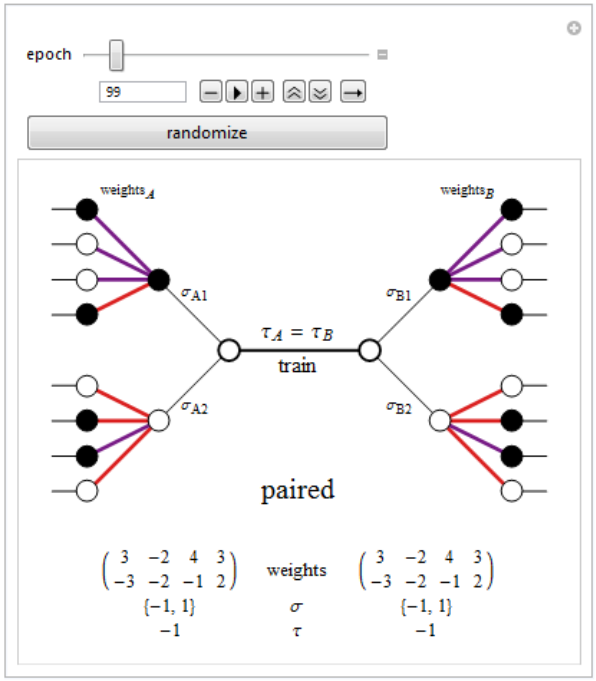
1. Mula-mula vektor input dibentuk dengan nilai $x_{i\_j} \in \{-1,1\}$.
2. Hitung nilai dari *hidden neurons* dengan rumus

   $\sigma i = sgn(\sum_{j=1}^{n} \omega i\_j \, x_{i\_j})$
3. Hitung nilai output untuk kedua TPM $\tau = \sum_{i=1}^{K} \sigma i$.
4. Bandingkan nilai output. Jika output tidak sama maka kembali ke langkah 1, jika sama maka update vektor bobot dengan memilih salah satu 3 learning rules.
5. Ulangi proses sampai vektor bobot dari kedua ANN sama. Kunci rahasia adalah vektor bobot

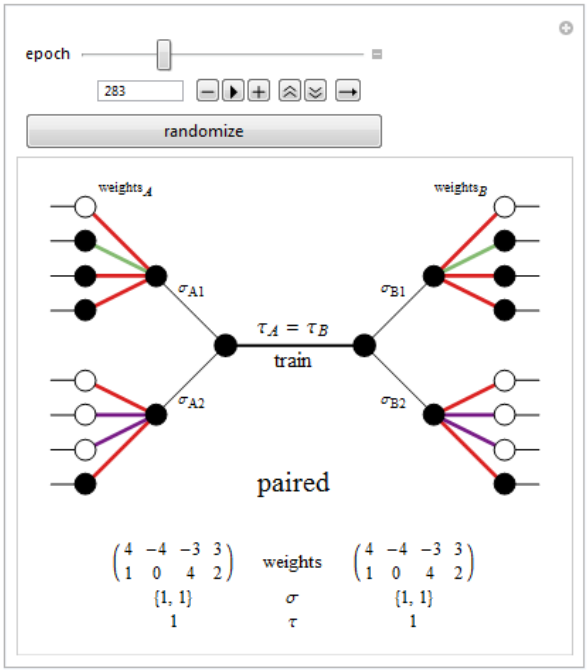
Hasil implementasi untuk setiap data uji adalah sebagai berikut :

id	epoch	Time to synchronize (second)
1	297	1.485

Berikut gambar eksperimen dari aplikasi untuk data 3 dan data 4 saja :



Gambar 8. Hasil eksperimen untuk data 3



Gambar 9. Hasil eksperimen untuk data 4

IV. ANALISIS

Hasil eksperimen menunjukkan bahwa dengan menggunakan . Baik dari data-1 sampai data ke-4 menunjukkan hanya dibutuhkan waktu yang sedikit untuk mendapatkan kunci rahasia *k*. Waktu maksimal adalah 4.345 detik. Hal ini kemungkinan besar diakibatkan karena Jaringan Saraf Tiruan yang digunakan adalah berupa Tree Parity Machine yang menyebabkan waktu yang dibutuhkan untuk sinkronisasi sangatlah cepat. Hal ini dikarenakan pada Tree Parity Machine *training* secara *mutual learning* jauh lebih cepat dibandingkan dengan *training* dari satu arah saja, dan semenjak kedua jaringan adalah Tree Parity Machine dan saling *mutual learning* satu dengan lainnya, maka efek cepatnya sinkronisasi akan bertambah.

V. KESIMPULAN

Dari percobaan yang dilakukan dapat disimpulkan bahwa metode enrkipsi kunci simetri dengan menggunakan ANN sangat efektif karena dari eksperimen didapat bahwa dibutuhkan waktu yang sedikit untuk menghasilkan kunci secret tanpa harus khawatir ada musuh yang ingin mengetahui.

VI. SARAN

Penulis mempunyai beberapa saran untuk pengembangan metode ini, antara lain :

1. Metode kunci simetri dengan algoritma ini masih rentan terhada serangan aktif atau *man-in-middle attacks*. Ini karena pesan akan diubah, yang berarti nilai vektor input dan output akan berubah yang akan menyebabkan vektor bobot menjadi sangat sulit untuk sama atau sinkron.

REFERENCES

[1] N. Prabakaran, "A New Security on Neural Cryptography with Queries" in Int. J. of Advanced Networking and Applications, 2010, pp. 437-444.

[2] A. Klimov, "Analysis of Neural Cryptography".

[3] I. DALKIRAN, "Artificial neural network based chaotic generator for cryptology" in Turk J Elec Eng & Comp Sci,, 2010.

[4] K. M. G. NOAMAN, "DATA SECURITY BASED ON NEURAL NETWORKS" , pp. 409–414.

[5] V. Gujral, "Cryptography using Artificial Neural Networks".

[6] A. Beutelspacher. *Kryptologie*. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, 2002.

[7] W. Kinzel and I. Kanter. Neural cryptography. cond-mat/0208453, 2002.

[8] http://www.codeproject.com/Articles/39067/Neural-Cryptography, diakses pada tanggal 24 Maret 2013, pukul 20.00 WIB.

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, 23 Maret 2013

Novan Parmonangan Simanjuntak
13509034