Redes de múltiplas camadas na geração de chaves criptográficas de sessão

Eduardo Gomes de Barros, José Demísio Simões da Silva

Instituto Nacional de Pesquisas Espaciais
eduardogbarros@hotmail.com, demisio@lac.inpe.br

Resumo

Segurança da informação está se tornando uma das maiores preocupações no mundo conectado de hoje. Existem diferentes abordagens em desenvolvimento. Este artigo mostra alguns resultados do uso de redes neurais artificiais, mais especificamente uma rede de múltiplas camadas para obtenção de chaves de sessão para uso em criptografia buscando confidencialidade, integridade de dados, autenticação e não-rejeição com base no princípio de Diffie-Hellman. Os resultados mostram a adequação das redes neurais na geração de chaves públicas.

1. Segurança e Criptografia

Criptografia é o estudo de métodos, ou sistemas, que permitem converter uma mensagem escrita em um dado alfabeto, normalmente com um significado associado – o texto às claras – em outra mensagem escrita em um alfabeto que pode, ou não, ser o mesmo da mensagem original, sem um significado associado – a mensagem cifrada – e vice-versa.

A história da criptografia possui registros com mais de 4000 anos. Entretanto, foi a proliferação do uso de computadores e de sistemas de comunicação, a partir da década de 60, que demandou a necessidade de proteger a informação na sua forma digital. Os primeiros trabalhos realizados para tentar atender esta demanda datam da década de 70.

O ano de 1976 [1] é particularmente importante pois Diffie e Hellman publicam o artigo *New Directions in Cryptography* que introduz o conceito de criptografia por chave pública e apresenta um novo método para troca de chaves.

Criptografia e segurança estão inter-relacionadas. Ela é uma ferramenta, das mais importantes, que permite que se alcancem os objetivos da segurança da informação. Criptografia, porém, atende somente aos seguintes objetivos:

- confidencialidade: mantém o conteúdo da informação a salvo de todos aqueles que não tem autorização para vê-lo;
- integridade dos dados: impede a alteração não autorizada dos dados;
- autenticação: identifica as entidades participantes da comunicação e/ou a origem dos dados; e,
- não-rejeição: previne que uma entidade negue ações anteriores que tenha efetivamente realizado.

1.1. Criptografia por Chave Pública

Seja $\{E_e: e \in K\}$ um conjunto das possíveis funções de cifragem e $\{D_d: d \in K\}$ o conjunto correspondente das funções de decifragem, onde: K é o conjunto de símbolos de um alfabeto onde cada elemento deste conjunto é uma chave; e é uma chave de cifragem; e, d é uma chave de decifragem.

Um sistema criptográfico por chave pública é aquele em que qualquer par de funções de cifragem/decifragem associadas (E_e, D_d) têm a seguinte propriedade: conhecido E_e é computacionalmente impossível, dado um texto cifrado c, achar a mensagem m tal que $E_e(m) = c$.

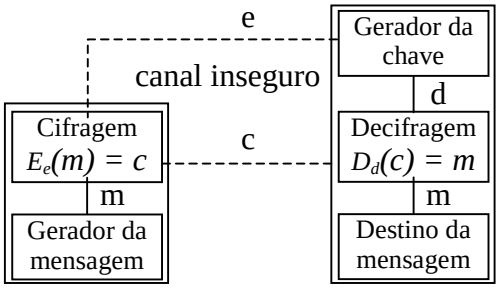

Figura 1. Criptosistema por chave pública

Corolário: uma vez conhecida a chave e é computacionalmente impossível determinar a chave de decifragem correspondente d.

Na Figura 1 ilustra-se o funcionamento da troca de chaves utilizando o conceito de criptografia por chave pública. Nela o usuário da direita seleciona um par de chaves (e, d); ele envia a chave de cifragem e, a chave

pública, sobre qualquer canal mantendo a chave de decifragem *d*, a chave privada, segura e secreta. O usuário da esquerda pode, então, mandar uma mensagem *m* aplicando a função de cifragem com a chave pública obtendo $c = E_e(m)$. O receptor é o único capaz de decifrar o texto cifrado *c* por ser o único conhecedor da chave de decifragem *d*. Para tal ele aplica a função D_d.

1.2. Algoritmo de Diffie-Hellman

Foi o primeiro algoritmo de chave pública proposto e foi desenvolvido para ser utilizado na troca de chaves.

Para que funcione os elementos da comunicação, A e B, executam os seguintes passos:

- A e B, utilizando qualquer canal de comunicação concordam em utilizar um único número primo muito grande *n* e outro número, *g*, tal que este último é uma primitiva a módulo *n*;
- A escolhe um número inteiro grande, *x*, e o remete para B $X = g^x \bmod n$
- B escolhe um número inteiro grande, *y*, e o remete para A $Y = g^y \bmod n$
- A calcula $k = Y^x \bmod n$
- B calcula $k' = X^y \bmod n$

Tanto *k* como *k'* são iguais a $g^{xy} \bmod n$. Ninguém que esteja escutando o canal pode computar este valor conhecendo somente *n*, *g*, *X* e *Y* a menos que se consiga determinar o logaritmo discreto e descobrir *x* ou *y*. O valor *k* é a chave secreta, ou de sessão, compartilhada por A e B e determinada independentemente.

A segurança, ou robustez do algoritmo, advém da dificuldade em se calcular logaritmos discretos.

1.3. Chave Criptográfica de Sessão

Uma técnica criptográfica muito utilizada é aquela em que cada conversação utiliza uma chave única – a chave de sessão.

É muito comum o uso de algoritmos da chave pública para estabelecer a chave de sessão que pode, então, ser utilizada em algoritmos criptográficos simétricos.

Alguns aspectos das chaves criptográficas são muito importantes e devem ser cuidadosamente analisados. Dentre eles o tamanho da chave o uso de funções pseudo-aleatórias.

Tamanho da chave. A determinação do melhor tamanho para uma chave criptográfica depende de vários aspectos:

- quanto vale a informação a ser protegida?
- por quanto tempo a informação deve ser mantida em segredo?
- quais os recursos que o adversário dispõe?

Algoritmos de chave pública, cuja robustez depende exclusivamente da dificuldade computacional em se realizar uma determinada operação matemática, necessitam de chaves maiores. Atualmente, chaves com 2048 bits, pelo menos, estão a salvo de tentativas de quebra.

Algoritmos simétricos cuja robustez depende não somente da chave como do algoritmo sendo utilizado necessitam de chaves cujos tamanhos dependem da aplicação, da freqüência de uso dentre outros parâmetros. Atualmente, chaves com 128 bits, pelo menos, mantém a informação a salvo por muito tempo mesmo contra inimigos que detenham muitos recursos.

Funções Pseudo-Aleatórias. Os algoritmos criptográficos dependem muito destas funções. Entretanto, computacionalmente, não é possível implementar um gerador de números aleatórios que produza uma seqüência realmente aleatória. Num computador todo gerador de seqüências aleatórias é periódico e, por definição, tudo que periódico, é predizível e não aleatório.

O melhor que se pode fazer, computacionalmente falando, é gerar uma seqüência de números pseudo-aleatória, isto é, uma seqüência que parece aleatória mas com uma periodicidade muito grande.

2. Uso de um Perceptron de Múltiplas Camadas na Geração de Chaves Criptográficas de Sessão

Em [2] tem-se a seguinte proposta de um algoritmo de geração de chave de sessão:

a) Os partícipes, A e B concordam em usar uma matriz 64x64 como pesos para uma rede MLP com 64 neurônios na camada escondida, uma camada de entrada e outra de saída, ambas com 64 nós, totalmente conectada. A chave pública;
b) A e B escolhem suas próprias chaves privadas, isto é, o conjunto de valores de entrada e de valores desejados;
c) Cada usuário treina sua rede com sua chave privada e a chave pública com o número de épocas variando entre 1 e 5;
d) A e B trocam a matriz peso de suas redes treinadas através de qualquer meio;
e) B treina sua rede com a matriz de pesos que recebeu de A utilizando sua chave privada e seu número de épocas;
f) A realiza o treinamento de sua rede com a matriz recebida de B e seu número de épocas;
g) A e B passam a ter uma rede MLP onde os pesos são iguais nas 3 primeiras casas decimais.

Em [2] encontram-se resultados experimentais que demonstram que é factível o uso deste algoritmo para geração de chaves, dentro de determinados parâmetros.

3. Algoritmo, Configurações e Parâmetros

Neste artigo tentou-se reproduzir o experimento descrito em [2]. Porém, os resultados obtidos foram diferentes, particularmente no número de casas decimais, no número de épocas e no tamanho da rede.

Interpreta-se a diferença nos resultados como sendo resultado de diferenças na interpretação dos parâmetros utilizados. Este artigo trata os seguintes parâmetros:

- Conversão das chaves;
- Intervalo de tratamento dos dados;
- Número de camadas na MLP (*hidden layers*);
- Número de neurônios em cada camada;
- Taxa de aprendizagem; e,
- Função de ativação;

Para tratá-los implementou-se um algoritmo que executa os seguintes passos:

- Cada partícipe estabelece o número de camadas, o tamanho da rede, a taxa de aprendizagem, o intervalo dos dados a função de ativação e o número de dígitos;
- É gerada uma matriz bi-dimensional com o tamanho da rede neural, que representa a chave pública, única para ambos partícipes, chamada de W;
- O partícipe *A* gera um vetor de entrada e um vetor de valores desejados, X_A e D_A, respectivamente, que representam a chave privada, com o tamanho da rede neural;
- *A* treina sua rede neural utilizando a chave pública e sua chave privada resultando na matriz de pesos W_A;
- O partícipe *A* gera um vetor de entrada e um vetor de valores desejados, X_B e D_B, respectivamente, que representam a chave privada, com o tamanho da rede neural;
- *B* treina sua rede neural utilizando a chave pública e sua chave privada resultando na matriz de pesos W_B;
- *A* envia a matriz de pesos W_A para *B* sobre um canal público e *B* envia para *A* a matriz de pesos W_B;
- *A* treina sua rede neural utilizando a matriz de pesos W_B, e sua chave privada resultando na matriz de pesos W_{AB}; e,
- *B* treina sua rede neural utilizando a matriz de pesos W_A, e sua chave privada resultando na matriz de pesos W_{BA}.

As matrizes W_{AB} e W_{BA} representam a chave única de sessão determinada por cada um dos partícipes utilizando uma informação pública e uma informação privada.

3.1. Parametrização para treinamento

Vários parâmetros devem ser corretamente interpretados e coerentemente sintonizados para permitir o treinamento das redes.

Conversão das Chaves. Na execução dos cálculos da MLP são utilizados números em ponto flutuante. Assim, a matriz de pesos gerada ao final do processamento é inteiramente constituída de valores de ponto flutuante.

Algoritmos criptográficos geralmente utilizam chaves com representação em inteiro e/ou caracteres. Assim, para compatibilizar a representação inteira e a de ponto flutuante adotou-se o critério de representar a chave como uma aproximação da matriz peso resultante da MLP. Para tal cada elemento da matriz é substituído por uma aproximação inteira a módulo 256.

Intervalo de Tratamento dos Dados. Geralmente a amplitude normalizada dos valores de entrada/saída de um neurônio são escritos no intervalo unitário fechado [0, 1] ou, ainda, em [-1, 1].

Este artigo utiliza os dois intervalos independentemente da função de ativação.

Número de Camadas na MLP. Utilizaram-se redes totalmente conectadas, com uma ou duas camadas escondidas, e com o mesmo número de neurônios em cada uma. As camadas de entrada e de saída têm o mesmo número de neurônios da camada escondida.

Número de Neurônios por Camada. Uma MLP com 4 neurônios, gera uma matriz resposta de dimensão 4x4 em que cada uma das 16 posições da matriz é substituída por um octeto. Assim, neste caso, a chave tem 4 x 4 x 8 = 128 bits.

Por não haver impedimento técnico redes com 4, 8, 16, 32, 64, 128 e 256 neurônios que geram chaves com 128, 512, 2048, 8192, 32768, 131072 e 524288 bits, respectivamente foram testadas.

Taxa de Aprendizagem. De [2] sabe-se que:

$$w_{AB} - w_{BA} = \eta^2 ab(\beta a - \alpha b) \quad (1)$$

onde $w_{AB} - w_{BA}$ representa a diferença entre as chaves geradas em A e em B; η é a taxa de aprendizagem; α o conjunto de valores desejados de A; a o conjunto de valores de entrada de A; β o conjunto de valores desejados de B; e, B o conjunto de valores de entrada de B.

Esta diferença representa o foco da análise. Como se vê em (1) ela é dependente do quadrado da taxa de apren-

dizagem. Assim, para que seja possível utilizar os três primeiros dígitos da representação em ponto flutuante precisa-se de uma diferença de 10^{-5} para garantir ausência de erros de aproximação na conversão.

Função de Ativação. A função de ativação define a saída de um neurônio em termos do estímulo recebido.

As funções sigmoidais [3] são as formas mais comuns de ativação usadas na construção de redes neurais artificiais não lineares. São funções estritamente crescentes com um balanceamento entre o comportamento linear e o não linear.

A função logística, exemplo de sigmoidal, é definida por:

$$\varphi(v) = \frac{1}{1+\exp^{-av}} \quad (2)$$

onde *a* é o parâmetro de inclinação da função.

A função tangente hiperbólica, outro exemplo de função sigmoidal, é definida por:

$$\varphi(v) = \tanh(v) \quad (3)$$

Neste trabalho foram aplicadas as funções logística e tangente hiperbólica sem qualquer restrição ao intervalo dos dados.

4. Critérios Criptográficos

A geração de senhas criptográficas não considera somente os parâmetros da MLP. Tem-se de considerar, também, critérios criptográficos como a robustez do algoritmo gerador de números pseudo-aleatórios.

Optou-se por utilizar o gerador de números pseudo-aleatório desenvolvido em Bill Chanders [4] por ser um gerador próprio para máquinas que utilizam 32 bits para representar números inteiros e 8 bits para caracteres a partir de uma semente.

A semente, que na prática é a senha informada pelo usuário, pode ter até 256 bytes, nesta implementação. A versão utilizada combina dois tipos de algoritmos por fluxo. O algoritmo mais lento é utilizado para introduzir uma chave no mais rápido – um gerador Jenkins. A resposta do algoritmo gerador de números pseudo-aleatórios são as senhas públicas e/ou privadas.

Para o algoritmo deste trabalho foi necessário que algumas modificações fossem introduzidas no algoritmo proposto por [4]. A principal delas envolve a conversão da saída, em inteiros longos, do algoritmo gerador de números pseudo-aleatórios em números de ponto flutuante entre [-1, 1] ou entre [0, 1].

Outro critério criptográfico foi não aceitar a entrada do usuário como semente pura e simplesmente. Ela é combinada com outros valores levantados em tempo de execução para montar a semente do processo de geração dos números pseudo-aleatórios.

5. Exemplo de Execução

A tabela abaixo ilustra a geração da chave de sessão para uma rede MLP com 1 camada e 2 neurônios:

Tabela 1. Geração de chave de sessão numa MLP com 1 camada e 2 neurônios

	Partícipes A e B	**OBS**
W	8.81722760E-01 9.68656130E-01 4.85254440E-01 1.57954080E-01	(1)
$\mathbf{X_A}$	3.98542350E-01 5.39760330E-01	(2) A
$\mathbf{D_A}$	1.00000000E+00 0.00000000E+00	
$\mathbf{W_A}$	8.81747129E-01 9.68689134E-01 4.85198813E-01 1.57878742E-01	(1)
$\mathbf{X_B}$	3.98542350E-01 5.39760330E-01	(2) B
$\mathbf{D_B}$	1.00000000E+00 0.00000000E+00	
$\mathbf{W_B}$	8.81731572E-01 9.68700211E-01 4.85267684E-01 1.58020329E-01	(1)
$\mathbf{W_{AB}}$	8.81755941E-01 9.68733214E-01 4.85212056E-01 1.57944990E-01	(3) A
$\mathbf{W_{BA}}$	8.81755941E-01 9.68733214E-01 4.85212057E-01 1.57944993E-01	(3) B
$\mathbf{Ch_A}$	113 200 (881 mod 256 968 mod 256) 229 157 (485 mod 256 157 mod 256)	(4) A
$\mathbf{Ch_B}$	113 200 (881 mod 256 968 mod 256) 229 157 (485 mod 256 157 mod 256)	(4) B

(1) De conhecimento público
(2) De conhecimento privativo de
(3) Matriz de pesos para geração da chave de sessão
(4) Chave de sessão gerada

Neste exemplo A e B concordam, inicialmente, com a chave pública W gerada a partir de uma semente e um gerador de números pseudo-aleatórios. Esta chave é de conhecimento de ambos e pode ser enviada por qualquer canal de comunicação. Numa implementação prática este valor seria gerado por um dos partícipes e informado ao outro juntamente com os demais parâmetros.

De posse da chave pública A e B geram suas chaves privadas, valores de entrada e desejados para a rede neural, X_A, D_A, X_B e D_B, respectivamente, utilizando sementes particulares para o gerador de números pseudo-aleatórios.

A treina a rede neural com sua chave privada e a chave pública gerando W_A. B também treina a rede neural com sua chave privada e a chave pública gerando W_B. Os valores gerados são transmitidos para o outro partícipe utilizando qualquer canal de comunicação.

A treina, novamente, a rede neural usando sua chave privada e W_B obtendo W_{AB}. B, por sua vez, também treina a rede neural usando sua chave privada e W_A obtendo W_{BA}.

As matrizes W_{AB} e W_{BA} têm valores idênticos e são as chaves de sessão.

Porém, para criptografia, o uso de números de ponto flutuante não são interessantes. Assim realiza-se uma conversão entre a representação em ponto flutuante e inteiro.

6. Fase de Testes

Durante a execução dos testes alguns parâmetros se mostraram mais determinantes na obtenção de chaves criptográficas do que outros, particularmente o relacionamento da taxa de aprendizagem e quais os dígitos decimais a serem utilizados.

Taxa de Aprendizagem e Dígitos Decimais. Os resultados obtidos indicam que para uma taxa de aprendizado de 10^{-5} e usando as três primeiras casas decimais, os valores gerados são muito próximos, senão iguais, para quase todas as chaves privadas independentemente da semente.

Sob a ótica exclusiva de redes neurais esta consideração é desnecessária e o algoritmo está absolutamente correto, porém, sob o ponto de vista criptográfico, falho. Dos valores numéricos obtidos reparou-se que a partir da quarta casa decimal começam a aparecer diferenças significativas.

Verificou-se a existência de uma relação entre a taxa de aprendizado e o dígito decimal a ser considerado. Para permitir maior flexibilidade introduziu-se mais o parâmetro DÍGITOS, na configuração da rede neural, que informa qual o último dígito decimal que o usuário deseja utilizar.

Testes indicam que as melhores combinações utilizam taxas de aprendizado entre 10^{-7} e 10^{-8} com aproveitamento das 5°, 6° e 7°, ou 6°, 7° e 8°, ou 7°, 8° e 9° casas decimais, isto é, com o parâmetro DÍGITOS valendo 7, 8 ou 9 respectivamente.

Considere, por exemplo, o valor obtido pela MLP 0,123456789123456 (*x*) e o parâmetro DÍGITOS configurado para 7. As operações de conversão do número em ponto flutuante para inteiro são:

- x = 0,123456789123456 * 10^7 = 1234567,89123456
- (aproximação para inteiro) x = 1234567
- fracao = $1234567/10^3$ = 1234,567
- x = (1234,567 – (int) 1234,567) * 10^3 = 567
- x = 55 = 567 mod 256

6.1. Resultados Obtidos

Os resultados obtidos com o programa de testes estão sumarizados na tabela a seguir:

Tabela 2. Resumo dos Resultados

			Hidden Layers							
			1	**2**	**1**	**2**	**1**	**2**	**1**	**2**
Intervalo			**[0, 1]**		**[0, 1]**		**[-1, 1]**		**[-1, 1]**	
Fç			**SIGM**		**TG HI**		**SIGM**		**TG HI**	
N	**T**	**D**	**% Coincidência**							
4	7	7	100	100	100	100	100	100	100	100
...										
8	7	7	100	100	100	100	100	100	100	100
...										
16	7	7	100	100	100	100	100	100	100	100
...										
32	7	9	100	100	100	100	100	100	**99**	100
...										
64	7	8	100	100	100	100	**99**	**99**	100	100
64	7	9	100	100	100	100	**99**	**99**	**99**	**99**
64	8	8	100	100	**99**	**99**	100	100	100	100
64	8	9	**99**	100	100	100	100	100	100	100
128	7	8	100	100	100	100	**99**	**99**	**99**	**99**
128	7	9	100	100	100	100	**99**	**99**	**99**	**99**
128	8	8	100	100	100	100	100	100	**99**	**99**
256	7	7	100	100	100	100	**99**	100	**99**	**99**
256	7	8	100	100	100	100	**99**	**99**	**99**	**99**
256	7	9	100	100	100	100	**99**	**99**	**99**	**99**
256	8	8	100	100	100	100	**99**	100	100	100
256	8	9	100	100	100	100	100	100	**99**	**99**

Na Tabela 2 encontram-se representados os resultados obtidos em redes com 1 ou 2 camadas escondidas (Hidden Layers), com intervalos de dados [0, 1] e [-1, 1], com as funções sigmoidal e tangente hiperbólica (Fç), utilizando

4, 8, 16, 32, 64, 128 e 256 neurônios (N), com taxas de aprendizado (T) 10^{-7} e 10^{-8} representados por 7 e 8, além dos dígitos decimais aproveitados para a conversão, 7, 8 ou 9. Os valores internos representam a taxa de coincidência entre as chaves de sessão geradas por A e por B.

Em todas as situações acima descritas foram geradas 2.500 diferentes chaves de sessão.

Destacaram-se as situações em que a taxa de coincidência não foi igual a 100%. As taxas diferentes de 100% variam entre 99,999883% e 99,999999% e foram representadas, todas, por 99%.

Da análise da Tabela 2 verifica-se que:

- Redes neurais com tamanho até 16 poderão ter qualquer precisão, qualquer número de dígitos para qualquer função e qualquer intervalo;
- Redes neurais com intervalo [0, 1], para qualquer função, com 64 neurônios, não poderão ter precisão 8 com 8 ou 9 dígitos;
- Redes neurais com intervalo [-1, 1], para qualquer função, com 32 neurônios, não poderão ter precisão 7 com 9 dígitos;
- Redes neurais com intervalo [-1, 1], para qualquer função, com 64 neurônios, não poderão ter precisão 7 com 8 ou 9 dígitos;
- Redes neurais com intervalo [-1, 1], para qualquer função, com 128 neurônios, não poderão ter precisão 7 com 8 ou 9 dígitos;
- Redes neurais com intervalo [-1, 1], para qualquer função, com 128 neurônios, não poderão ter precisão 8 com 8 dígitos;
- Redes neurais com intervalo [-1, 1], para qualquer função, com 256 neurônios, não poderão ter precisão 7 com 7, 8 ou 9 dígitos; e,
- Redes neurais com intervalo [-1, 1], para qualquer função, com 256 neurônios, não poderão ter precisão 8 com 8 ou 9 dígitos.

7. Conclusões

Verifica-se que os resultados são muito bons para todos os resultados se considerado somente o aspecto de rede neural. O pior percentual de coincidência corresponde a uma rede com 32x32 = 1.024 bytes de saída. Como foram geradas 2.500 chaves diferentes tem-se 2.560.000 valores obtidos. Deste total somente 3 foram diferentes. Entretanto, sob a ótica criptográfica, esta discrepância não é aceitável.

Verifica-se que o melhor resultado prático é a combinação da função sigmoidal com o intervalo [0, 1]. Entretanto, nada, nos testes realizados, impede a utilização de qualquer outra combinação a menos das listadas anteriormente.

O que os resultados estão apontando é que a rede deve ser genérica, isto é, ela possui um aspecto de generalização perdendo em efetividade à medida que que fica mais especializada. Isto pode ser visto á medida que se aumente o número de neurônios nas camadas. O aumento de neurônios corresponde a uma especialização.

Uma possível visão para utilização prática deste algoritmo deve introduzir a transmissão, por exemplo, após a geração da chave de sessão, de um valor hash da chave gerada para comparação com o valor gerado pelo outro partícipe.

Deve ser feita, ainda, uma análise crítica da robustez das chaves geradas analisando se um invasor, de posse das informações públicas, consegue gerar as chaves privadas utilizando qualquer método, particularmente o método da força bruta dentro de um período de tempo exeqüível.

Referências

[1] B. Schneier, Applied Cryptography: protocols, algorithms and source code in C, 2nd Ed, John Wiley & Sons, Inc, 1996.

[2] L. P. Yee, L. C. de Silva, Application of Multilayer Perceptron Networks in Public Key Cryptography, World Congress on Computational Intelligence, Hawaii, 2002.

[3] S. Haykin, Neural Networks: a comprehensive foundation. 2nd Ed, Prentice Hall, Inc, 1999.

[4] B. Chambers, A cryptographic pseudo random number generator, Article: 139 of sci.crypt.research, 1995.