Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

Disciplina: Engenharia de Segurança (SSC-0747)

Professora Doutora Kalinka Regina Lucas Jaquie Castelo Branco

Aluno: Eduardo Lourenço Pinto Neto, n°USP: 6793121
Aluno: Emanuel Carlos de Alcântara Valente, n°USP: 7143506
Aluno: Sibelius Seraphini, n°USP: 7152340
Data: 01/07/2013

Trabalho 2 - Criptografia

1 Comparação de algoritmos de criptografia

Foram comparados os seguintes algorimtos, AES x RSA e DES x 3DES. Nas próximas seções, além da comparação entre esses algoritmos, haverá uma descrição mais detalhada sobre cada um deles.

1.1 AES x RSA

1.1.1 AES

O AES (Advanced Encryption Standard ou Padrão de Criptografia Avançado) é um algoritmo de chave simétrica. Esse algoritmo substítuiu o DES (Padrão de Criptografia de Dados) quando o governo americano em 1997 lançou um processo de seleção que definiria o novo algoritmo de chave simétrica para proteger informações do governo federal. O AES é baseado no príncipio de permutação e substituição de rede, ele usa um bloco de tamanho fixo de 128 bits com chaves de tamanho 128, 192 e 256. O tamanho da chave indica o número de repetições de ciclos: 10 ciclos para 128 bits, 12 ciclos para 192 bits e 14 ciclos para 256 bits.

De maneira geral, o algoritmo AES funciona da seguinte forma:

1. Expansão da chave - expande uma chave pequena em número de chaves cíclicas separadas
2. Ciclo inicial
   (a) AddRoundKey - cada byte do estado e combinado com um ciclo de chave usando o operação de bit XOR
3. Ciclos
   (a) SubBytes - Substituicão não linear utilizando uma lookup table
   (b) ShiftRows - na etapa de transposição cada linha é deslocada um certo número de etapas
   (c) MixColumns - Combina 4 bytes de cada coluna
4. Ciclo Final - Todos os passos do ciclo exceto o MixColumns

1.1.2 RSA

O RSA é um algoritmo de chave pública que é baseado na presumida dificuldade de de fatorar inteiros muito grandes. As chaves secretas do RSA são dois números primos muito grandes e sua chave pública é o produto desses dois números. O algortimo RSA involve três partes: geração da chave, encriptação e decriptação.

A chave é gerada da seguinte forma:

1. Escolha dois primos distintos p e q;
2. Calcule $n = p * q$;
3. Calcule $\phi(n) = \phi(p)\phi(q) = (p-1)(q-1) - (\phi$ - Função Totiente de Euller);
4. Escolha um inteiro tal que 1 ¡ e ¡ $\phi(n) emcd(e, \phi(n)) = 1$, ou seja e $\phi(n)$ sejam primos entre si;
5. Determine $d : d^{-1} \equiv e(mod\phi(n))$.

A chave pública consiste do módulo n e do expoente e. A chave privada consiste do módulo n e do expoente d. Além disso, p, q e $\phi(n)$ também devem ser mantidos em segredo.

Encriptação: Sendo m a mensagem sem criptografia a ser transmitida e c a mensagem criptografada.

$$c \equiv m^e mod(n)$$

Decriptação: A mensagem m pode ser recuperada usando a chave privada d.

$$m \equiv c^d mod(n)$$

1.1.3 RSA x AES

O RSA, embora já existam no mercado novos e sofisticados métodos e algoritmos como o AES ? Padrão Avançado de Ciframento (Advanced Encryption Standart), algoritmos de curvas elípticas até de algoritmos quânticos, o RSA é um dos mais utilizados hoje em dia, principalmente em dados enviados pela Internet. A incapacidade de se fatorar números muito grandes utilizando os

sistemas computacionais atuais torna este algoritmo muito forte. Ele é o único capaz de implementar assinatura digital e troca de chaves, entre os outros algoritmos mais comuns. O mesmo vem sendo largamente utilizado em várias aplicações de segurança envolvendo desde as camadas mais altas como certificados digitais, assinaturas digitais, S/Mime e até as camadas mais inferiores como o SSL (Secure Socket Layer) e TLS na camada de transporte e IPSEC na camada de rede Nos itens seguintes estaremos mostrando brevemente o que cada uma das aplicações de segurança se utilizam do RSA nas diversas camadas de rede. Estaremos mostrando também o formato do algoritmo bem como é realizado o gerenciamento das chaves no RSA.

1.2 DES x 3DES

1.2.1 DES

O DES foi criado em 1977, sendo largamente utilizado desde então. Foi adotado pelo National Bureau of Standards (NIST). Seu funcionamento consiste na criptografia de blocos de 64 bits de entrada com uma chave de 56 bits, gerando blocos de 64 bits como saída. Para obter o texto original novamente, reverte-se o processo usando a mesma chave. Devido ao pequeno tamanho desta, esse algoritmo já não é considerado seguro, pois ele poderia ser quebrado com certa facilidade pela força bruta.

Até hoje alguns expecialistas são apreensivos em relação ao DES, pois existem supostos blocos internos ao DES cujo critério de arquitetura não foi explicitado, levando-os a crer que existiriam falhas internas não reveladas pelo autor.

1.2.2 3DES - Triple DES

Esse algoritmo foi criado como uma evolução do DES, visto que já se nutriam dúvidas sobre a segurança deste, devido a chave ser pequena e o seu tamanho não estar aberto a expansões.

A idéia desse novo algoritmo é fazer múltiplas cifragens com o DES, utilizando uma chave diferente para cada processo. Primeiramente, optou-se pela idéia do Double DES, que corresponde a dois processos de criptografia, sendo cada um com uma chave distinta.

Descobriu-se, no entanto, que o ataque Meet-in-the-middle era capaz de reduzir o custo da quebra desse algoritmo a 256 tentativas, o que corresponde

ao mesmo custo de se utilizar apenas uma chave.

Tendo em vista esse problema, Tuchman propôs o Triple DES, ou seja, criou-se um processo que evita o ataque descrito anteriormente, que funciona criptografando-se o texto com a chave K1, decriptando o resultado com a chave K2 e criptografando-se novamente o resultado com K1. Esse processo gera uma quebra pela força bruta da ordem de 2112 tentativas, o que, atualmente, ainda é considerado seguro.

Além disso, percebe-se que esse mesmo algoritmo pode ser utilizado para criptografar e decriptografar utilizando o DES, bastando para isso utilizar K1=k2.

Por essas razões, o Triple DES tornou-se um algoritmo popular como alternativa ao DES, sendo adotado pelo ANS X9.17, ISO 8732, e pelo Privacy-Enhanced Mail (PEM).

2 Implementação dos Algoritmos

Foram implementados 2 algoritmos. O primeiro, trata-se de um algoritmo *(chaos crypto)* baseado na teoria do caos e mapa logísticos. O segundo é uma implementação do algoritmo RC4 modificado.

2.1 Chaos Crypto

2.1.1 Introdução

Este algoritmo baseia-se na teoria do caos e como sistema caótico foi considerado o mapa logístico, que é um sistema dinâmico definido como

$$x_{n+1} = \lambda x_n(1 - x_n). \tag{1}$$

O mapa logístico foi originalmente desenvolvido para simular o crescimento populacional de insetos. Embora muito simples, para alguns valores de λ o mapa logístico entra em regime caótico e deste modo, apresenta um comportamento imprevisível e muito sensível as condições iniciais, neste caso x_0. Para o sistema implementado, foi adotado $\lambda = 4$, que é o valor com o máximo de regime caótico do sistema.

O sitema descrito na Equação 1 é um bom gerador de números aleatórios, por exemplo, a sequência de números [0;3600; 0;9216; 0;2890; 0;8219; 0;5854; 0;9708; 0;1133; 0;4020; 0;9616; 0;1478] foi gerada pelos primeiros 10 valores

de x para x0 = 0.1. ESta sequência aleatória é imprevisível, porém determinística. Entretanto, há uma ressalva, o mapa logístico é sensível a todas as casas decimais utilizadas no Cálculo. As sequências são idênticas apenas quando utilizamos a mesma arquitetura de computador, sistema operacional de linguagem de programação. Essa caracterísitca define a sensibilidade dos sistemas caóticos as condições iniciais.

Durante a implementação foram considerados os seguintes conceitos: O algoritmo usa chave simétrica. A mensagem a ser criptografada é representada pelo vetor $\vec{p}$ que deve ser codificada no vetor $\vec{c}$. A informação de $\vec{c}$ deve ser intelegível e apenas com a senha, representada pelo vetor $\vec{\pi}$. As funções de codificação e decodificação são representadas, repectivamente por $E_{\vec{\pi}}$ e $D_{\vec{\pi}}$.

Portanto, a sequência para codificação e decodificação serão as seguintes:

$$\vec{c} = E_{\vec{\pi}}(\vec{p}) \tag{2}$$

e

$$\vec{p} = D_{\vec{\pi}}(\vec{c}) \tag{3}$$

2.1.2 Convertendo a senha em condição inicial

Como segundo passso do algoritmo, a senha $\vec{\pi}$ deve ser convertida em condição inicial (x_0). Usaremos a seguinte equação para este propósito:

$$x_0 = \sum_{k=1}^{n(\vec{\pi})} \frac{\vec{\pi}(k)}{8} \tag{4}$$

Portanto, x_0 será o valor inicial da Equação 1 na obtenção do vetor de codificação $\vec{c}$.

2.1.3 Obtendo o vetor codificação

Uma vez calculado $x_0(\vec{\pi})$, o próximo passo é gerar uma sequência de números utilizando a Equação 1. O número de elementos utilizados no vetor de codificação deve ser o mesmo da mensagem. É importante deixar o sistema iterar o suficiente para que ocorram as influências da perturbação, assim serão consideradas os n elementos após a centésima iteração. Os valores de $x = 10^5 + n(\vec{p})$ obtidos pelo mapa logístico são números reais entre 0 e 1. Será

necessário convertê-los em números inteiros entre 0 e 255 obtendo o vetor $\vec{k}$. Para isso, foi utilizada a seguinte equação:

$$k_i = \lfloor 10^4 x_{10^5+i} \rfloor mod 256 \tag{5}$$

2.1.4 Codificando e Decodificando

Para estas duas operações é necessário utilizar uma operação abeliana (reversível) combinando a mensagem com os elementos do vetor $\vec{k}$. Um dos operadores que atendem esse requisito é o operador **XOR**, representado por $\oplus$.

A Equação 6 representa a codificação, nela para cada caractere da mensagem ($\vec{p}$) é realizada uma operação *xor* com o elemento correspondente do vetor $\vec{k}$, obtendo a mensagem criptografada $\vec{c}$.

$$c_i = E_{\vec{\pi}}(p(i)) = k_{vec\pi} \oplus p(i) \tag{6}$$

Da mesma forma, para operação de decodificação, relizamos a mesma operação de codificação, mudando apenas os operandos:

$$p_i = D_{\vec{\pi}}(c(i)) = k_{vec\pi} \oplus c(i) \tag{7}$$

2.2 RC4 modificado

Em criptografia, RC4 (ou ARC4) é o algoritmo de criptografia de fluxo mais usado no software e utilizado nos protocolos mais conhecidos, como Secure Socket Layers (SSL) (para proteger o tráfego Internet) e WEP (para a segurança de redes sem fios. RC4 não é considerado um dos melhores sistemas criptográficos pelos adeptos da criptografia, e em algumas aplicações podem converter-se em sistemas muito inseguros. No entanto, alguns sistemas baseados em RC4 são seguros o bastante num contexto prático.

De uma forma geral, o algoritmo consiste em utilizar um array que a cada utilização tem os seus valores permutados, e misturados com a chave, o que provoca que seja muito dependente desta. Esta chave, utilizada na inicialização do array, pode ter até 256 bytes (2048 bits), embora o algoritmo seja mais eficiente quando é menor, pois a perturbação aleatória induzida no array é superior. Aplicação e segurança

A variação em relação ao algoritmo RC4 padrão é feito da seguinte forma: O algorimto implementado usa 2 vetores de estados S1 e S2, e 2 índices j1 e j2. A cada iteração, i é incrementado e 2 bytes são gerados:

Em relação ao algoritmo RC4 padrão, que usa S1 e j1, porém, no último passo, a soma $S1[i] + S1[j1]$ é procurada em S2. Depois, essa operação é repetida (sem incrementar i de novo) em S2 e j2 e a saída é $S1[S2[i]+S2[j2]]$. Abaixo o algoritmo em pseudo-linguagem:

```
Todos os cálculos são em módulo 256
i := 0
j1 := 0
j2 := 0
while GeraSaida:
    i := i + 1
    j1 := j1 + S1[i]
    troca valores de S1[i] e S1[j1]
    saida S2[S1[i] + S1[j1]]
    j2 := j2 + S2[i]
    troca valores de  S2[i] e S2[j2]
    saida S1[S2[i] + S2[j2]]
endwhile
```

Embora o algoritmo requeira o mesmo número de operações por byte gerado, há um aumento de paralelismo em relação ao RC4 padrão, obtendo um ganho de velocidade.

Embora, mais forte que o RC4 padrão, técnicas parecidas com este algoritmo foram exploradas por Alexander Maximov e a equipe da NEC.

2.3 Comparação dos 2 algoritmos implementados

Os dois algoritmos apresentaram bom desempenho para encriptar e decriptografar de texto pequenos e médios (até 20Mb). Porém, o algoritmo RC4 modificado é mais inseguro em relação ao primeiro (crypto chaos).

O algorimto de criptografia caótica baseado no mapa logístico, embora simples, é funcional e pode ser utilizado para realizar criptografia de arquivos de texto. Para que esse algorimto possa ser utilizado em diferentes linguagens, máquinas e arquiteturas de computadores, ele precisará ser implementado utilizando aitmética de precisão variável. Embora não seja trivial, a

criptografia deste algoritmo pode ser quebrada, sendo que a criptoanálise para tanto já foi estudada na literatura científica.

2.4 Manual de Compilação e Uso

2.4.1 Compilação

Para compilar ambos algoritmos é necessário ter um compilador da linguagem C, como o gcc; um compilador java e a máquina virtual java.

Para compilar, execute o comando: *make*

2.4.2 Encriptar e Decriptar

Considere os seguintes arquivos:

- *arquivoentrada.txt*: arquivo de entrada que será encriptado;
- *arquivosaida.enc*: arquivo de saída que será encriptado;
- *arquivosaida.txt*: arquivo de saída que contém o arquivo decriptado;
- *chave.txt*: arquivo que contém a chave.

Para gerar a chave, utilizando o número 7 como seed, digite o seguinte comando:

./t2-engseg 1 K chave.txt 7

./t2-engseg 2 K chave.txt 7

O primeiro exemplo gera a chave para o algoritmo 1 *crypto chaos* e o segundo para o algoritmo 2 *RC4*. Ambas chaves são de 2048 bits.

Para encriptar o arquivo *arquivoentrada.txt* utilizando o primeiro algoritmo com a chave anteriormente gerada, digite o seguinte comando:

./t2-engseg 1 E chave.txt arquivoentrada.txt arquivosaida.enc

E para decriptar, utilizando o algoritmo 1, digite o seguinte:

./t2-engseg 1 D chave.txt arquivosaida.enc arquivosaida.txt