Segurança do algoritmo RSA em novas aplicações

Samáris Ramiro Pereira - junho/2009
Faculdade de Tecnologia de São Bernardo - FATEC SB, SBC, SP, Brasil
profa_samaris@yahoo.com.br

Resumo. *O algoritmo criptográfico de chave pública RSA é um padrão mundial tanto para confidencialidade como para assinatura digital, certificando usuários e servidores de rede. Ele utiliza conceitos fundamentais para a compreensão da criptografia moderna, sendo um alicerce para pesquisas na área. Este artigo, elaborado após a análise dos mais de 30 anos de pesquisa do RSA (desde 1977), cita as principais variações e ataques apresentados para o algoritmo e concentra em um único texto as complexidades computacionais e os cuidados envolvidos na sua segurança.*

Palavras-chave. *Criptografia, Complexidade Computacional, Algoritmo RSA.*

Abstract. *The cryptographic algorithm of public key RSA is a world pattern for confidentiality and for digital signature, certifying users and net servants. It uses fundamental concepts for the understanding of the modern cryptography, being a foundation for researches in the area. This article, elaborated after the analysis of the more than 30 years of research of RSA (since 1977), it mentions the main variations and attacks presented for the algorithm and it concentrates in a single text the computacional complexities and the cares involved in his safety.*

Key Words. *Cryptography, Computational Complexity, RSA algorithm.*

1. **Introdução**

Métodos criptográficos é a forma mais adequada de fornecer integridade, confidencialidade e autenticação de dados e entidades sobre informações digitais, pois independem da natureza dos dados e dos meios físicos onde eles são armazenados. Neste contexto, crescem as pesquisas na área de segurança da informação e entre elas as de criptologia.

A metodologia [8] utilizada para a elaboração desse artigo foi a de pesquisas bibliográficas no tema, constituída de livros, artigos de periódicos e de materiais disponibilizados na internet, em páginas idôneas. Com base na análise do material pesquisado (o qual foi tema de dissertação de mestrado da autora [9]), abrangendo os mais de 30 anos de pesquisa do RSA (desde 1977), relacionaram-se as principais variações e ataques apresentados para o algoritmo e concentraram-se em um único texto os cuidados e as complexidades computacionais (as quais foram obtidas através de análises, cálculos e verificação quando existentes, e todas foram convertidas para a unidade de medida "operações binárias").

Complexidade computacional [9] é o estudo dos recursos necessários durante o processamento de um algoritmo, a fim de se determinar o custo computacional de execução deste algoritmo. Este estudo deve ser independente do equipamento utilizado para o processamento. Os recursos estudados são principalmente de tempo de processamento e de espaço em memória. No contexto deste artigo, o estudo em questão é o tempo de processamento.

Para mensurar o custo computacional de execução de um algoritmo define-se uma função custo ou função de complexidade *f*, onde *f(n)* é a medida de tempo, por exemplo, necessária para executar um algoritmo, considerando-se uma entrada de tamanho *n*. Observa-se que a complexidade de tempo (neste caso) não representa tempo diretamente, mas o número de vezes que determinada operação considerada relevante é executada.

Existem três medidas de custo computacional com base nos valores de entrada do algoritmo que está sendo analisado: para o melhor caso (menor custo de execução sobre todas as entradas possíveis de tamanho *n*), para o caso médio (média do custo de execução sobre todas as entradas possíveis de tamanho *n*) ou para o pior caso (maior custo de execução sobre todas as entradas possíveis de tamanho *n*).

O estudo do custo computacional dedica-se a valores grandes de *n*, isto é, estuda-se o comportamento assintótico[1] da função de complexidade *f(n)* e usualmente, apresenta-se a complexidade de pior caso, denotado por *O(f(n)).*

Um algoritmo apresenta custo computacional tratável, quando é possível a sua execução em tempo polinomial no tamanho dos dados de entrada. Um algoritmo apresenta custo computacional intratável, quando não se sabe se existe uma solução para executá-lo em tempo polinomial para todas as instâncias deste algoritmo.

Um algoritmo apresenta execução em tempo polinomial quando seu custo computacional de tempo de execução é dado pela função *O(p(n))* onde *p* denota uma função polinomial e *n* denota o tamanho da entrada. Um algoritmo apresenta solução em tempo exponencial quando não é possível se encontrar solução em tempo polinomial, por exemplo, $O(n^{lg\ n})$.

---

[1] Comportamento assintótico é o limite da função de complexidade computacional f(n), quando n cresce arbitrariamente.

A definição de um algoritmo com custo computacional tratável ou intratável independe dos recursos computacionais específicos para um determinado caso. Um algoritmo com tempo polinomial pode ser ineficiente para algum caso da mesma forma que um algoritmo com tempo exponencial pode ser eficiente para um determinado caso. Um algoritmo com custo computacional exponencial $O(2^n)$ é mais eficiente que um algoritmo com custo computacional polinomial $O(n^5)$ para uma entrada n menor ou igual a 20.

2. Sistema criptográfico de chave pública

Um sistema criptográfico de chave pública, conceito apresentado em 1976 [1], se baseia em uma função unidirecional com segredo[2], a qual permite que, para cada par de chaves, pública e privada, a chave pública seja de conhecimento geral, porém, a chave privada seja apenas de conhecimento do proprietário do par de chaves. O proprietário da chave secreta pode calcular eficientemente a função em ambas as direções, cifrando e decifrando, enquanto que os possuidores apenas da chave pública, só terão tratabilidade para calcular a função de ciframento. Três tipos de sistemas criptográficos de chave pública predominam no mercado, os quais se baseiam nos problemas que seguem [10]: problema da fatoração de inteiros (PFI), problema do logaritmo discreto sobre corpos finitos (PLD) e problema do logaritmo discreto sobre curvas elípticas (PLDCE). Apesar do RSA estar relacionado com mais de um problema, ele é o exemplo mais conhecido que utiliza PFI. São exemplos de aplicação do PLD o ElGamal, o protocolo para troca de chaves Diffie-Hellman e ainda o algoritmo de assinatura digital DSA (parte do padrão de assinaturas digitais DSS). Por último, surgiram sistemas baseados no PLDCE [2], análogos aos já existentes, os quais não apresentavam vantagens apresentando apenas contribuição acadêmica. Outros algoritmos PLDCE análogos aos PLD podem apresentar vantagens por possuírem complexidade exponencial e os análogos PLD possuírem complexidade subexponencial. Assim, tais sistemas apresentam um mesmo nível de segurança com comprimentos de chave menores, indicados para aplicações onde há limitação, visto que uma chave RSA de 1024 bits apresenta o mesmo nível de segurança que uma de 160 bits de um algoritmo de curvas elípticas equivalente.

O algoritmo RSA permite troca de chaves, criptografia e assinatura digital, sendo mais completo do que o algoritmo DSA [12] (para assinatura digital) e que o algoritmo Diffie-Hellman (para troca de chaves). Apesar do ElGamal permitir troca de chaves, criptografia e assinatura digital, ele é bem menos utilizado do que o RSA. Comparando o RSA com algoritmos de curvas elípticas, observa-se que existem patentes relacionadas a cálculos específicos para algoritmos com curvas elípticas e a patente do RSA original está expirada desde 2000. Algoritmos com curvas elípticas podem prover o nível de segurança do RSA com chaves menores, sendo mais eficientes do que o RSA na assinatura e deciframento, porém, são mais lentos para cifrar e verificar assinaturas. Outro aspecto é que, devido à criação dos algoritmos com curvas elípticas serem recentes em relação ao RSA, a segurança do RSA foi muito mais testada. Devido a estas constatações, o RSA é muito utilizado atualmente e sua utilização tem crescido ainda mais desde que expirou a validade da sua patente, por exemplo, para a

---

[2] Função na qual é tratável de se calcular $y=f(x)$, mas é intratável, o cálculo de $x=f^1(y)$, a menos que se tenha conhecimento de determinado parâmetro (segredo).

segurança em aplicações Java, em softwares como Skype [13], PGP [14], SSL e OpenSSL, em Cartões Inteligentes e na ICP-Brasil [15, 16].

As pesquisas sobre o RSA se encontram dispersas, incluindo uma época sem divulgação via Internet, o que gerou pesquisas redundantes. Isto contribui com a maior vulnerabilidade do RSA, a falta de atenção quanto aos cuidados ao se desenvolver uma aplicação. E os cuidados crescem gradativamente com o desenvolvimento computacional e matemático.

3. O algoritmo RSA

Desenvolvido em 04/1977 [3], pelos professores **R**ivest, **S**hamir e **A**dleman, batizado com as iniciais de seus nomes. A patente 4.405.829 norte-americana do RSA emitida em 20/09/1983, com licença exclusiva para a RSA Security Inc. por meio do Massachusetts Institute of Technology, com expiração para 20/09/2000, incluindo confecção de produtos com o algoritmo, sua utilização e sua venda [17]. Mas a RSA Security permitiu durante este período, a utilização não comercial gratuita do algoritmo, principalmente para uso acadêmico. No dia 06/09/2000, a RSA Security renunciou a patente e tornou público o algoritmo, porém, existem variações patenteadas por outros proprietários e com licença em vigor.

O RSA permite: **(1).** Ciframento e deciframento de mensagem; **(2).** Geração e verificação de assinatura digital; **(3).** A combinação dos itens anteriores.

Em 1999, Boneh [4] afirmou que a estrutura do RSA nunca precisou ser alterada, os ataques são utilizações inadequadas, evitadas com o uso dos parâmetros apropriados. Afirmação válida até hoje, sendo que a lista atualizada dos cuidados necessários para a sua correta implementação pode ser vista ao longo deste artigo.

O RSA gera a chave pública/privada a partir de primos grandes. O tamanho dos primos varia conforme o caso: maior os primos, maior segurança e menor desempenho.

O RSA utiliza três funções unidirecionais com segredo:

(1). Dificuldade da fatoração de números grandes, função exposta a ataques por fatoração: dado dois primos grandes, p e q, é tratável se calcular $n=pq$, porém, fatorar n a fim de descobrir p e q é computacionalmente intratável.

(2). Dificuldade de se encontrar a raiz e-ésima discreta, função exposta a ataques por criptograma conhecido: calcular uma exponenciação em módulo n é tratável: $c=m^e \bmod n$, mas determinar m sendo a raiz *e-ézima* de *mod n,* é intratável, a menos que se tenha conhecimento do inverso multiplicativo de $e \bmod \varphi(n)$ ou da fatoração de n.

(3). Dificuldade de se calcular logaritmos discretos módulo n, função exposta a ataques por mensagem conhecida: dado $m = c^d \bmod n$, tem-se que $\log_c m = \log_c c^d = d$, *então* $d=\log_c m$. Na década de 90, o problema do logaritmo discreto foi muito pesquisado, porém os mais eficientes algoritmos para cálculo de logaritmos discretos possuem complexidade computacional de tempo de processamento semelhante aos algoritmos de fatoração mais eficientes no momento. Geralmente, o logaritmo discreto em um grupo arbitrário de tamanho n possui complexidade $O(\sqrt{n})$ [5].

3.1. Processo de criação das chaves

Para a criação das chaves RSA [11] são escolhidos dois primos grandes distintos: p e q, os quais devem ser mantidos em segredo. Computa-se o módulo RSA, também chamado módulo n ou módulo: $n=pq$. Calcula-se $\varphi(n)=(p-1)(q-1)$.

Escolhe-se um inteiro positivo *e*, chamado de expoente público, tal que 1< e< $\varphi(n)$, de forma que mdc$(e, \varphi(n)) = 1$. Determina-se *d,* chamado de expoente privado, tal que $ed-1$ seja divisível por $\varphi(n)$. O par (e, n) é a chave pública *e* o par (d, n) é a privada. O ciframento é dado por $c = m^e \bmod n$, sendo *m* representação numérica (em inteiros positivos) do texto legível, dividido em blocos menores que *n*. O deciframento é dado por $m = c^d \bmod n$. A escolha de *p, q* e *e*, deve ser cuidadosa.

3.2. Ciframento por substituição em blocos

Uma mensagem *M* é cifrada por substituição em blocos de tamanho fixo com *k* bits: $M=\{m_1,m_2,..., m_z\}$, onde m_z é um bloco de *k* bits. Cada bloco m_i deve ter um valor menor que o valor do módulo *n* e, portanto, menor ou igual a $\lg_2 n$ bits, mas não pequeno o suficiente para sofrer ataques de força bruta. Se o tamanho de *k* bits do bloco m_i fosse pequeno, um ataque de força bruta seria possível pela criação de um dicionário contendo os pares ordenados (m_i, c_i), com $c_i=(m_i^e) \bmod n$. Esta variação de ataque de força bruta não testa todas as chaves possíveis, mas verifica no dicionário criado todos os criptogramas possíveis encontrando sua respectiva mensagem. O custo computacional deste ataque é de 2^k ciframentos $c_i=(m_i^e) \bmod n$, ou seja, $O(2^k(lg\ e)(lg^2\ n))$ operações binárias. Um exemplo de bloco suscetível a este ataque seria o composto por um caractere ASCII (1 byte) ou por um caractere Unicode (2 bytes). Como o módulo *n* tem valores na ordem de *1024* bits ou maior, o tamanho do bloco tem no mínimo *1023* bits e com $|m_i| = 1023$ bits não é tratável um ataque de força bruta utilizando um dicionário "mensagem/criptograma".

Para possibilitar o deciframento correto é necessário que o tamanho em bits do bloco seja menor que o tamanho em bits do módulo *n*. Para otimizar o tempo de processamento e o espaço de armazenamento, é necessário que o tamanho do bloco em bits seja o mais próximo possível do tamanho em bits do módulo *n*, desta forma, se *k* bits for o tamanho máximo do bloco e *w* bits for o tamanho de *n*, deve-se assegurar $k=w-1$.

O módulo *n* é o produto de 2 números primos ímpares, portanto n nunca será potência de 2, o que garantiria que $n-1$ tivesse 1 bit a menos que n. Segue um exemplo de definição do tamanho do bloco, com valor teórico para o módulo *n*, que por ser pequeno possibilita melhor visualização. Para $n =35$ (5x7), o valor do bloco tem que ser menor que 35. O módulo $n=35$ é 100011_2 em binário logo $w=6$. Como um bloco pode ser no máximo 34 (100010_2), em princípio k=6, mas não haveria a possibilidade de que fosse cifrado qualquer bloco com 6 bits o que dificultaria o controle e portanto $k=w-1=5$.

Sempre que o tamanho da codificação do texto legível não proporcionar uma divisão exata pelo tamanho do bloco legível, haverá a necessidade de se completar o tamanho do último bloco, conforme um padrão a ser adotado entre as partes envolvidas, por exemplo, preenchendo-o com bits nulos.

4. Variações do RSA

Uma variação de um algoritmo criptográfico consiste em uma modificação realizada na definição original deste, a fim de trazer benefício ao processo, seja segurança, tempo de processamento, espaço de armazenamento em memória ou outro. Pode ocorrer em diferentes partes do algoritmo e atingir objetivos distintos.

Devido ao tempo de existência do RSA, surgiram variações e algumas foram adotadas como padrão em normas técnicas. Não há análise para se

determinar qual a melhor variação RSA, pois cada variação visa um objetivo. Deve-se analisar a melhor opção para a implementação em questão, levando-se em consideração fatores como necessidade de segurança, tempo de processamento, capacidade computacional, possibilidade de investimento e taxas relativas a patentes. A tabela 1 apresenta variações e finalidades. Maiores informações podem ser obtidas em [9].

Tabela 1. Variações RSA e suas finalidades

Variaçao	**Finalidade**
RSA utilizando o Expoente Universal	agilizar o decifamento
RSA com Chave Privada Particionada	compartilhar chave privada
RSA em Lote	obsoleto
RSA CRT pelo Algoritmo de Gauss	agilizar o decifamento
RSA CRT pelo Algoritmo de Garner	agilizar o decifamento
RSA CRT pelo Algoritmo ART	agilizar o decifamento
MRSA	agilizar o decifamento
RSA *PkQ*	agilizar o decifamento
DRSA	segurança
DRSA por Padhye	segurança e velocidade
RSA Redistribuído	agilizar o ciframento
MRSA Redistribuído	agilizar o ciframento
DRSA por Padhye Redistribuído	agilizar o ciframento seguro
RSA Desbalanceado	segurança por longo prazo
Twin RSA	analisa uso de primos gêmios

5. Algoritmos de fatoração – ameaça à segurança RSA

Um algoritmo apresenta custo computacional intratável, quando não se sabe se existe uma solução para executá-lo em tempo polinomial para todas as suas instâncias. Em um custo computacional preocupa-se com valores grandes de n, isto é, estuda-se o comportamento assintótico da função de complexidade *f*(n) e usualmente, apresenta-se a complexidade de pior caso, denotado por *O(f(n)).*

A segurança do RSA depende, principalmente, da não existência de algoritmo eficiente que explore uma das três fraquezas das funções unidirecionais com segredo que compõem o seu estado da arte. A mais preocupante das fraquezas é a fatoração de inteiros. Existem dois tipos de algoritmos de fatoração: Os algoritmos de propósito especial, que dependem do tamanho dos fatores desconhecidos do inteiro *n* que se deseja fatorar, de forma que, sua eficiência, depende do tamanho do menor fator primo de *n*. Ele é mais eficiente para fatorar números que possuem fatores primos pequenos do que para fatorar números que possuem fatores primos balanceados. Os algoritmos de fatoração de propósito geral dependem do tamanho do número *n* que se deseja fatorar. Os custos de tempo de processamento dos principais ataques ao RSA, incluindo os algoritmos de fatoração, são listados na tabela 2 e consideram: n, inteiro a ser fatorado; e, constante de Euler (*e=2,7182818*); p, menor fator primo de *n* e *o(1),* função assintótica que se aproxima de 0, se *n* tende ao infinito.s

Devido à importância neste contexto da análise das complexidades, a tabela 3 complementa a 2, apresentando outras complexidades envolvidas.

Tabela 2. Ataques RSA e suas complexidades de tempo de processamento

Ataque	Custo em operações binárias
Módulo n em Comum	$O(\lg^2 max(e_x, e_y))$
Módulo n com Fator Primo em Comum	$O(\lg^2 max(n_A, n_C))$
por Tabela	$O((2^{m_2})(lg\ e)(\lg^2 n))$
Falsificação de Assinatura Digital	$O(\lg^2 max(s_1, s_2))$
Erro Aleatório na Assinatura	$O((lg\ e)(\lg^2 n))$
Ocultamento I	$O(\lg^2 n)$
Ocultamento II	$O((lg\ e)(\lg^2 n))$
Mensagem que Não sofre Redução Modular	$O(e \lg\ e)$
Hastad	$O(\lg^2 N)$ com N sendo $\prod_{i=1}^{k} n_i$
Exposição LSB da Chave	$O(e \lg\ e)$
Exposição MSB da Chave	$O(e \lg\ e)$
$\|p - q\|$ pequeno	$O(\|p - q\|)$
Cíclico de Simmons e Norris	custo exponencial
Cíclico Generalizado	custo exponencial
Fatoração: $\rho - 1$	$O(p')$, onde p' é o maior fator primo de $p - 1$
Fatoração: $\rho + 1$	$O(p')$, onde p' é o maior fator primo de $p + 1$
Fatoração: ECM	$O(e^{(1+o(1))(2\ln p\ \ln\ln p)^{\frac{1}{2}}})$
Fatoração: QS	$O(e^{(1+o(1))(\ln n \ln\ln n)^{\frac{1}{2}}})$
Fatoração: NFS	$O(e^{(1,923+o(1))(\ln n)^{\frac{1}{3}}(\ln\ln n)^{\frac{2}{3}}})$

6. Cuidados necessários em novas aplicações

A maior vulnerabilidade do RSA se encontra na falta de atenção quanto aos cuidados que seguem, a serem observados ao se desenvolver uma aplicação que utilize o algoritmo RSA, envolvendo implementação, escolha das chaves e utilização do algoritmo [6]: **(1).** Os primos *p* e *q* devem ser grandes para evitar ataques de força bruta. A ordem de grandeza deles deve ser analisada conforme o contexto da aplicação (necessidade de segurança, capacidade computacional relacionada à aplicação). O tamanho mínimo para *p* e *q* considerado seguro é de 512 bits. **(2).** Os primos *p* e *q* devem ter aproximadamente o mesmo comprimento em bits para evitar ataques com algoritmos de fatoração no primo de menor tamanho. Em aplicações atuais é recomendável a utilização de módulos com 2048 bits e nestes casos, os primos devem ter 1024 bits cada. **(3).** Os números p-*1* e *q-1* devem ter um fator primo grande, caso contrário, fica possível fatorar *n* em tempo tratável. No mínimo, *p-1=2p'* e *q-1=2q'*, com *p'* e *q'* também primos da mesma ordem de grandeza que *p* e *q*. **(4).** Os números *p+1* e *q+1* devem ter um fator primo grande, caso contrário, torna-se possível fatorar *n* em tempo computacional tratável. No mínimo, *p+1=2p''* e *q+1=2q''*, com *p''* e *q''* também primos da mesma ordem de grandeza que *p* e *q*. **(5).** A diferença |*p-q*| não pode ser pequena, pois se *p* é próximo de *q*, então *p* é aproximadamente igual à raiz quadrada de *n* e efetuando-se testes sucessivos para todos os primos próximos a raiz quadrada de *n* encontra-se *p* (ou *q*). Este ataque possibilita a fatoração de *n* em tempo *O(|p-q|),* o qual não deve ser tratável. Como referência, utiliza-se |*p-q*| aproximadamente igual a 2^x, *x* sendo o quociente de |*n*| dividido por 4. **(6).** O mdc (*e-1, p-1*) e o mdc(*e-1, q-1*) devem ser pequenos para evitar um elevado número de pontos fixos. **(7).** O tamanho do bloco em bits não pode ser pequeno demais para que possibilite a construção de uma tabela de pares ordenados *(mensagem, criptograma).* O tamanho de bloco ideal é menor inteiro maior ou igual a $\lg_2(n)$-1.

Tabela 3. Complexidades de tempo de processamento relacionadas ao RSA

Operação	Custo em operações binárias
Algoritmo AKS	$O(lg^{19}\ n)$
Algoritmo CRT por ART	$O(kt^2)$ sendo k o número de primos de n e t o tamanho dos primo
Algoritmo CRT por Garner	$O(kt^2)$ sendo k o número de primos de n e t o tamanho dos primo
Algoritmo CRT por Gauss	$O((kt)^2)$ sendo k o número de primos de n e t o tamanho dos primo
Algoritmo de Euclides	$O(lg^2\ n)$
Algoritmo de Euclides Estendido	$O(lg^2\ n)$
Algoritmo de Fatoração ρ de Pollard	$O(n^{\frac{1}{4}})$
Algoritmo de Fatoração por Fermat	$O(\frac{n}{2})$
Ataque por Força Bruta na Chave	$O(2^n)$
Ataque por força bruta no criptograma RSA	$O(2^k(lg\ e)(lg^2\ n))$
Ciframento RSA	$O((lg\ e)(lg^2\ n))$ sendo e o expoente
Crivo de Eratóstenes	$O(\sqrt{n})$
Deciframento RSA com Expoente Universal	$O((lg\ d)(lg^2\ n))$ sendo d o expoente
Deciframento RSA original	$O((lg\ d)(lg^2\ n))$ sendo d o expoente
Divisão em $\mathbb{Z}$ $(a = qb + r)$	$O((lg\ a)(lg\ b) = O(lg^2\ n))\ \ n = max\{a, b\}$
Exponenciação Modular LR	$O((lg\ e)(lg^2\ n))$ sendo e o expoente e n o módulo
Exponenciação Modular RL	$O((lg\ e)(lg^2\ n))$ sendo e o expoente e n o módulo
Gerenciamento de chaves assimétricas	$O(n)$
Gerenciamento de chaves simétricas	$O(n^2)$
Inversa Multiplicativa	$O(lg^2\ n)$
Máximo Divisor Comum entre dois números	$O(lg^2\ n)$
Multiplicação em $\mathbb{Z}$ $(a \times b)$	$O((lg\ a)(lg\ b) = O(lg^2\ n))\ \ n = max\{a, b\}$
Multiplicação em $\mathbb{Z}_n$ $(a \times b)\bmod n$	$O(lg^2\ n)$
Soma em $\mathbb{Z}$ $(a + b)$	$O(lg\ a + lg\ b) = O(lg\ n)\ \ n = max\{a, b\}$
Soma em $\mathbb{Z}_n$ $(a + b)\bmod n$	$O(lg\ n)$
Subtração em $\mathbb{Z}$ $(a - b)$	$O(lg\ a + lg\ b) = O(lg\ n)\ \ n = max\{a, b\}$
Subtração em $\mathbb{Z}_n$ $(a - b)\bmod n$	$O(lg\ n)$
Teorema Chinês de Resto	$O(lg^2\ n)$
Teste Probabilístico de Miller-Rabin	$O(\lg^3\ n)$
Teste Probabilístico de Solovay-Strassen	$O(\lg^3\ n)$

(8). Em uma assinatura digital, ao se particionar a mensagem *M* em blocos, não se deve permitir *mdc(m, n)* diferente de 1, evitando assim, um ataque de fatoração de *n* via *mdc (c, n).* **(9).** O algoritmo RSA não deve ser aplicado isoladamente, mas sempre dentro de uma codificação de mensagem, o que impede diversos tipos de ataques. **(10).** O par de chaves RSA do usuário deve apresentar certificação de validade de AC credenciada, evitando um ataque MITM. **(11).** Os valores relacionados à chave privada, tais como de *p*, *q*, *φ(n)* ou suas variações devem permanecer secretos assim como o expoente *d*. **(12).** O proprietário da chave privada deve ter cuidado, não só quanto a não divulgá-la, mas também a não utilizá-la sem necessidade, por exemplo, em mensagens aleatórias, as quais podem ser utilizadas como parte de algum ataque. **(13).** O expoente privado *d*, não pode ser menor que $n^{0,292}$. **(14).** O expoente público *e* deve ser no mínimo $e = 2^{16}+1=65.537$ para módulos de 1024 bits. Para módulos *n* de 2048 bits devem ser utilizados expoentes e com no mínimo 21 bits. **(15).** O módulo *n* não pode ser utilizado por mais de um usuário, assim como não deve existir na aplicação, dois módulos n_1 e n_2 tal que (n_1, n_2) seja diferente de 1. **(16).** Para utilização de uma variação do RSA, observar os cuidados específicos. **(17).**

Ao se implementar uma aplicação RSA, deve-se atentar que somente atendendo a todos os cuidados, a implementação estará segura. **(18).** Atentar a novas publicações - podem surgir ataques ou variações úteis.

7. Conclusões e pesquisas em aberto

Estão em aberto, pesquisas que podem influenciar na história futura do RSA, tais como: teste de primalidade determinístico com tempo computacional menor do que o algoritmo AKS [7], algoritmo para fatoração de propósito geral mais eficiente que o NFS [18] ou de propósito especial mais eficiente que o ECM [2], algoritmo eficiente para cálculo da raiz e-ésima módulo *n,* algoritmo para cálculo de logaritmo discreto módulo *n* mais eficiente do que os atuais, algoritmos de variação RSA com benefício novo.

O algoritmo RSA utiliza uma teoria matemática elegante, sendo que a análise dos fundamentos matemáticos, dos algoritmos envolvidos e de suas complexidades possibilita a correta implementação, confiança na sua segurança, alicerce para estudos mais sofisticados e é embasamento para a criação de novos algoritmos, sejam variações do RSA ou novas propostas. A maior vulnerabilidade do RSA se encontra na falta de atenção quanto aos cuidados que seguem, a serem observados ao se desenvolver uma aplicação que utilize o algoritmo RSA, envolvendo implementação, escolha das chaves e utilização do algoritmo.

8. Referências

8.1. Artigos em revistas e anais e capítulos de livros

[1] Diffie, W., Hellman, M.E. (1976) “New Directions in Cryptography”, Cambridge University Press, 2005. IEEE International Symposium on Information Theory in Ronneby, Sweden, June.

[2] Johnson, D., Menezes, A. (1999) “The Elliptic Curve Digital Signature Algorithm (ECDSA)”, Technical Report CORR 99-31, Depto of C&O, University of Waterloo, Canada.

[3] Rivest, R.L., Shamir, A., Adelman, L.M. (1978), “A method for obtaining digital signatures and public-key Cryptosystems”, Communications of the ACM, 21, 120-126.

[4] Boneh, D. (1999) “Twenty years of attacks on the RSA cryptosystem”, journal Notices of the American Mathematical Society, volume 46, number 2.

[5] Rivest, R.L. (1992), “Response to NIST's proposal”, Communications of the ACM 35, 41-47.

[6] Pereira, S.R. (2009), “30 anos do Algoritmo Criptográfico de Chaves Públicas RSA”, 6th CONTECSI, SP, Brasil.

[7] Agrawal, M., Kayal, N., Saxena, N. (2002), “Primes is in P”.

8.2. Livros e teses

[8] Lakatos, E.M., Marconi, M.A. (1985) “Fundamentos de Metodologia Científica”, São Paulo, Ed. Atlas.

[9] Pereira, S.R. (2008) “O Sistema Criptográfico de Chave Pública RSA”, Dissertação apresentada à Universidade Católica de Santos, UNISANTOS.

[10] Camara, D.P.B.A. (2001), “Criptografia de Chave Pública baseada em Curvas Elípticas com Aplicações”, Universidade Federal de Pernambuco, Dissertação apresentada à Universidade Federal de Pernambuco.

[11] Menezes, A.J., Oorschot, P.C., Vanstone S.A. (2001), “Handbook of Applied Cryptography”, CRC Press.

8.3. *Internet*

[12] National Institute of Standards and Technology - NIST, FIPS Publication 186-2: Digital Signature Standard (DSS) (2000), Disponível em: http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf. Acesso em: 10 Junho 2009.

[13] Skype Technologies, Guide for Network Administrators - edition 1.0.1 (2009), Disponível em: http://www.skype.com/security/guide-for-network-admins.pdf. Acesso em: 10 Junho 2009.

[14] P. Zimmermann, Philip Zimmermann Home Page (2009), Disponível em: http://www.philzimmermann.com/EN/background/index.html. Acesso em: 10 Junho 2009.

[15] ICP Brasil, Infra-estrutura de Chaves Públicas Brasileira (2009), Disponível em: http://www.icpbrasil.gov.br/. Acesso em: 10 Junho 2009.

[16] Instituto Nacional de Tecnologia da Informação, Autoridade Certificado Brasileira Raiz – ITI (2009), Disponível em: HTTP://www.iti.gov.br/twiki/bin/view/ITI/Apresentacao. Acesso em: 10 Junho 2009.

[17] Laboratórios RSA, RSA Security Inc. (2009), Disponível em: http://www.rsasecurity.com/rsalabs/. Acesso em: 10 Junho 2009.

[18] NFSNET project, NFSNET Large-scale Distributed Factoring (2009), Disponível em: http://www.nfsnet.org. Acesso em: 10 Junho 2009.

9. Contato

Samáris R. Pereira é mestre em Informática com linha de pesquisa em Criptografia. Bacharel em Matemática e Tecnóloga em Técnicas Digitais. Atualmente é professora de graduação (Fatec SB, Av. Pereira Barreto, 400, SBC, SP, fone 4121-8905) e pós-graduação (Faculdade Anchieta, Rua Cáspio, 300, SBC, SP, fone 4123-3427).