Estudo dos Ataques Matemáticos ao RSA e Hipótese de Modificação do Algoritmo

Javier García López, Lúcio Adolfo Meurer

Curso de Ciência da Computação - Centro Universitário Lasalle (Unilasalle)
Canoas – RS – Brasil

javier@lasalle.tche.br, l.meurer@terra.com.br

Abstract. *This article presents a brief study on the public key algorithm RSA and its vulnerabilities, setting some changes in it, taking advantage on certain well known numeric properties. This new version of the algorithm is going to be tested, compared to the original RSA, on the key generation, cipher, decipher, attacks and the final results are presented and evaluated, focusing the impact generated by the algorithm change.*

Resumo. *Este apresenta um estudo do algoritmo de chave pública RSA e suas vulnerabilidade e propor mudanças, aproveitando propriedades numéricas, tentando resolver os problemas de vulnerabilidade apresentados pela literatura de segurança. A partir da nova versão do RSA, este será avaliado, com respeito ao original nos seguintes quesitos, custo computacional de geração das chaves, cifragem decifragem e finalmente quebra. O artigo apresenta os resultados e a avaliação dos mesmos, focando no impacto gerado pela alteração do algoritmo*

Palavras-chave: ataques matemáticos, criptografia de chave pública, Função de Euler, RSA.

Workshop: Seguridad de Datos

1. Contextualização do Problema

Pela sua ampla utilização em sistemas de comércio eletrônico, sistemas de comunicação assíncrona, como correio eletrônico dentre vários outros, o RSA vem sendo testado sob diversos aspectos a fim de garantir-se a sua segurança.

O criptossistema RSA, apresentado pela primeira vez em 1978, por Rivest, Shamir e Adelman [RSA1978] faz parte de um conjunto de algoritmos criptográficos ditos de chave pública, onde a chave de cifragem é diferente da chave usada para decifrar.

O presente artigo visa estudar os ataques matemáticos ao RSA e sugerir uma alteração no algoritmo, visando um aumento de esforço exigido para a quebra por meios de ataques matemáticos.

Para tanto, a seção 2 trata de trabalhos relacionados onde se apresentam resultados de vários trabalhos já publicados sobre o RSA. A seção 3 traz uma descrição do método de experimentação levada a cabo e dos controles usados. A geração de chaves para o algoritmo e operações de cifragem e decifragem é tratada na seção 4, e os ataques ao RSA na seção 5. Finalmente as seções 6 e 7 tratam dos resultados obtidos e conclusões do artigo.

2. Trabalhos Relacionados

O presente artigo usa como base de conhecimento diversas fontes, iniciando pelo artigo da publicação do RSA [RSA1978]. As bases matemáticas, os métodos de geração de chaves, cifragem, decifragem e uma primeira abordagem de criptoanálise são citadas nesse artigo.

Conforme o tempo comprovava as qualidades do RSA, várias novas técnicas de criptoanálise e ataques foram sendo desenvolvidos. O artigo de Dan Boneh, [BONEH1999], "*Twenty years of attacks on the RSA cryptosystem*", faz um estudo de vinte anos de ataques sofridos pelo RSA e termina por concluir que, passadas duas décadas, ainda não se havia desenvolvido um ataque efetivo para uma implementação do RSA que levasse em conta alguns fatores críticos básicos na implementação do RSA.

William Stallings, [STALLINGS2003], também discorre sobre os ataques de implementação aos quais o RSA pode estar exposto, bem como os ataques de "*timing*", ou de temporização, porém não apresenta nenhum método efetivo de ataque ao RSA que seja universalmente aplicável ou que não seja evitável através de cuidados durante a implementação.

O presente artigo difere dos anteriores por enfocar uma característica inerente ao RSA, onde um dos fatores básicos do funcionamento do algoritmo pode vir a ser usado como meio de sua quebra e sugere uma pequena alteração de implementação a fim de diminuir a possibilidade de exploração dessa característica.

3. Descrição do Método

O presente artigo visa estudar os principais ataques matemáticos ao RSA, propondo dois modelos de ataques e apresentando o desempenho, em termos de operações computacionais da comparação entre o algoritmo RSA tradicional e sua versão alterada.

O desempenho do RSA tradicional e do RSA alterado é medido em vários aspectos, a saber:

Geração de chaves: tanto o RSA tradicional quanto o RSA alterado terão o número de operações iterativas necessárias à geração das chaves pública e privada. Para cada um dos algoritmos de geração de chave, implementa-se uma variável contadora de iterações, de modo que, ao final do algoritmo, a mesma seja fornecida juntamente com a chave gerada.

Para os ataques matemáticos, a lógica apresentada baseia-se na técnica de ataque a texto conhecido conforme descrito em Lucchesi [LUCCHESI1986].

Para fins de controle de desempenho todos os algoritmos serão implementados na mesma linguagem de programação, usando o maior número possível de linhas idênticas.

O presente artigo não tratará da comparação de desempenho em termos de tempo, entre as versões tradicional e alterada do RSA; isso somente poderá ser feito, quando já implementado o método alterado.

4 Geração de Chaves

A geração das chaves no RSA e em sua versão alterada baseia-se na escolha de um número relativamente primo ao produto de dois outros números e no cálculo do complemento multiplicativo desse número escolhido.

A fim de que o RSA funcione adequadamente, as chaves precisam manter a característica de serem complemento multiplicativo módulo 1 de $\phi(n)$, ou seja:

$$e \times d \bmod \phi(n) = 1$$

Apesar do RSA alterado não obedecer à premissa básica exposta acima, o que inviabiliza sua utilização prática como criptossistema, o impacto observado da alteração justifica o estudo dessa alteração.

4.1 Geração das Chaves Pública e Privada no RSA Tradicional

A geração das chaves pública (identificada como a variável "e") e privada ("d") dá-se exatamente da mesma maneira em ambos os algoritmos.

A geração das chaves do RSA inicia da escolha de dois números primos, "p" e "q", da maneira que segue [RSAES-OAEP2000]:

$[2(L-1)/2] + 1 \leq p \leq [2(L/2)] - 1$

$[2(L-1)/2] + 1 \leq q \leq [2(L/2)] - 1$

onde o valor de L é o comprimento desejado da chave, em bits.

Os números primos passíveis de serem usados encontram-se no quadro abaixo, extraído de [STALLINGS2003] e alterado pelo autor:

4 bits	11	13																			
5 bits	17	19	23	29	31																
6 bits	37	41	43	47	53	59	61														
7 bits	67	71	73	79	83	89	97	101	103	107	109	113	127								
8 bits	131	137	139	149	157	163	167	173	179	181	191	193	197	199	211	223	227	229	233	241	251

Quadro 1 - Primos entre 4 e 8 bits. Fonte: adaptado de Stallings [STALLINGS2003], com base no trabalho do autor

Partindo-se dos já citados dois números primos relativamente grandes1 ("p" e "q"), obtém-se sua multiplicação:

$$n = p \times q$$

Inicia-se com o cálculo da totiente de Euler para "n" no RSA tradicional.

A função totiente de Euler indica o número de primos relativos de um determinado número, assumindo a seguinte forma:

$$\phi(p) = p^k - p^{k-1}$$

$$\phi(q) = q^k - q^{k-1}$$

$$\phi(n) = \phi(p \times q) = (p^k - p^{k-1}) \times (q^k - q^{k-1})$$

Como "p" e "q" são números primos, as equações podem ser simplificadas da seguinte maneira:

$$\phi(p) = p^1 - p^0 = p-1$$

$$\phi(q) = q^1 - q^0 = q-1$$

$$\phi(n) = (p^1 - p^0) \times (q^1 - q^0) = (p-1)x(q-1)$$

Após isso se escolhe o número "e", tal que o mesmo seja relativamente primo a $\phi(n)$, ou seja:

$$MDC(e, \phi(n)) = 1,$$

onde MDC significa maior divisor comum de ambos os números.

A escolha de "e", embora seja fácil definir um número relativamente primo a $\phi(n)$, deve recair sobre um número razoavelmente grande [RSAES-OAEP2000]. Para fins desse trabalho, teremos um "e" como a metade do menor dos números entre "p" e "q", incrementados em um:

$$e \geq (\min(p+1, q+1))/2$$

A fim de selecionar a variável "e", pode-se lançar mão do algoritmo abaixo, usando o

[1] Atualmente são usados números primos com 100 dígitos decimais.

teorema de Euclides conforme [STALLINGS2003] e [LUCCHESI1986]:

```
função Escolhe_e (p, q)
// Escolhe o valor inicial de "e" conforme critério de início descrito acima
e:= (min(p+1,q+1))/2
contador_e :=1
phi(n)= (p-1)x(q-1)
enquanto Euclides_MDC (e, phi(n) ) <> 1 faca
      {
      e:= e+1
      contador_e := contador_e + 1
      }
retorna e
retorna contador_e

função Euclides_MDC (e, phi(n))
// Baseado em [LUCCHESI1986] página 50
x:= e;
y:= phi(n)
Repita
    resto := x mod y
    x := y
    y := resto
até que resto = 0
retorna x
```

Ato contínuo calcula-se o complemento multiplicativo de e, tal que:

$$e \times d \bmod \phi(n) = 1$$

O seguinte algoritmo permite calcular o valor de "d"

```
função calcula_d (e, ϕ(n))
phi(n):= ϕ(n)
d := 0
repita
   d = d+1
   contador_d = contador_d +1
até que ((e x d) mod phi(n) = 1)
retorna d
retorna contador
```

Obtidos esses dois valores ("d" e "e"), os mesmos serão chave pública e privada, sendo um deles tornado público juntamente com o valor de "n" e o outro mantido secreto para fins de decifragem.

Na tabela abaixo estão calculados os valores de "e" e "d", bem como os custos individuais e total (contador_e + contador_d) em termos de iterações de sua computação:

p	*q*	*n*	*ϕ(n)*	*e*	*d*	*contador_e*	*contador_d*	*total*
11	13	143	120	7	103	2	102	104
19	23	437	396	13	61	4	60	64
47	53	2491	2392	25	2105	2	2104	2106
103	107	11021	10812	55	2359	4	2358	2362
181	199	36019	35640	91	14491	1	14491	14492
173	211	36503	36120	89	28409	3	28408	28411
191	223	42593	42180	97	5653	2	5652	5654
		Total	127660				Total	53193

Tabela 1 - Cálculo de "d" e "e"

4.2 Geração das Chaves Pública e Privada no RSA Alterado

O processo de geração de chaves no RSA alterado segue exatamente a mesma dinâmica, com a multiplicação de dois primos, obtendo-se o mesmo número "n", e a seguir substituindo o valor de ϕ(n) por uma variável de teste ("z") que vai conter o valor para a função de Euler alterada.

Assim temos:

$$z = (p-2)x(q-2)$$

A escolha de "e" e o cálculo de "d" seguem o mesmo racional, ou seja:

$$MDC(e, z) = 1$$

$$e \times d \bmod z = 1$$

O algoritmo da escolha de "e" no algoritmo alterado segue a mesma estrutura algorítmica, apenas substituindo a linha:

phi(n)= (p-1)x(q-1)

pela linha:

z:= (p-2)x(q-2)

O algoritmo que permite calcular o valor de "d" segue exatamente a mesma lógica do RSA tradicional, sendo usado o mesmo algoritmo apresentado na seção anterior.

Na tabela abaixo estão calculados os valores de "e" e "d", bem como os custos individuais e total (contador_e + contador_d) em termos de iterações de sua computação:

p	*q*	*n*	*z*	*e*	*d*	*contador_e_alt*	*contador_d_alt*	*total*
11	13	143	99	7	85	2	84	86
19	23	437	357	10	250	1	249	250
47	53	2491	2295	26	971	3	970	973
103	107	11021	10605	52	7138	1	7137	7138
181	199	36019	35263	91	34488	1	34487	34488
173	211	36503	35739	89	6425	3	6424	6427
191	223	42593	41769	97	9904	2	9903	9905
		Total	126127				Total	59267

Tabela 2 - Cálculo de "d" e "e" na versão alterada

4.3 Comparação na Eficiência da Geração de Chaves

Da análise dos dados preliminares, fruto de testes de mesa dos respectivos algoritmos, lançados nas tabelas 1 (Tabela 1 - Cálculo de "d" e "e") e 2 (Tabela 2 - Cálculo de "d" e "e") para uma amostra de sete conjuntos de "p" e "q", verificou-se uma pequena diferença em termos de desempenho na geração de chaves entre o algoritmo RSA tradicional e sua versão alterada. O RSA tradicional apresentou um somatório que representa 41,67% de iterações em relação ao somatório dos ϕ(n) enquanto o RSA alterado apresentou um percentual de 46,99%.

4.4 Cifragem e Decifragem

As operações de cifrar e decifrar são exatamente as mesmas em ambas as versões do algoritmo. Por envolverem as mesmas operações de exponenciação e módulo, tanto o RSA tradicional quanto o RSA alterado apresentam o mesmo desempenho, considerando-se somente a variação dos valores do expoente.

As operações de cifragem se resumem a [RSA1978]:

função cifra_RSA (e, n, M)
// M é a mensagem às claras que se quer cifrar
// C designa a mesma mensagem após aplicada a função de cifra
C := M^e mod n
retorna C

função decifra_RSA (C, d, n)
// C é a mensagem cifrada
// M é a mensagem originalmente cifrada
M := C^e mod n
retorna M

5 Ataques ao RSA

Os ataques ao RSA possuem três abordagens principais, conforme pode ser verificado em [STALLINGS2003].

A primeira abordagem, chamado de força bruta, concentra-se em tentar descobrir todas as chaves privadas possíveis.

A defesa contra os ataques de força bruta reside exatamente em gerar chaves num intervalo tão grande que seja computacionalmente muito caro, em termos de tempo e / ou operações, definir qual a chave a ser usada.

A segunda abordagem, inicialmente apresentada por Paul Kocher ([STALLINGS2003] *apud* KOCHER, Paul, "*Timing attacks on Implementations of Diffie Hellman, RSA, DSS, and Other Systems*") baseia-se em técnicas de determinação de uma chave privada baseando-se somente no tempo levado por um computador para decifrar uma mensagem.

A terceira abordagem, objeto de estudo do presente artigo, baseia-se em técnicas matemáticas e já é descrita em [RSA1978]. William Stallings, em [STALLINGS2003] classifica em três os ataques desse tipo:

- Fatoração de "n" em seus dois fatores primos.
- Determinação direta de $\phi(n)$ sem primeiro determinar "p" e "q".
- Determinar "d" diretamente, sem determinar $\phi(n)$

5.1 Fatoração de n

A fatoração de "n" deve ser razoavelmente difícil, a fim de não apresentar uma solução trivial para a segurança do RSA.

Inicialmente os autores do RSA estimaram o tempo necessário para fatorar n, usando um algoritmo não publicado de Richard Schroeppel, sendo definido como:

$$tf_{(RSA)}=\ln(n)^{\sqrt{(\ln(n)/\ln\ln(n))}}$$

Segundo essa estimativa, um "n" de 200 dígitos (10200) consumiria um total aproximado de 1,2 x 1023 operações para ser faturado, levando 3,8 x 109 anos.

Os dados inicialmente estimados pelos autores do RSA [RSA1978] são apresentados no quadro abaixo:

Tamanho de n	Número de operações	Tempo Estimado
50	$1,4 \times 10^{10}$	3.9 horas
75	$9,0 \times 10^{12}$	104 dias
100	$2,3 \times 10^{15}$	74 anos
200	$1,2 \times 10^{23}$	$3,8 \times 10^{9}$ anos
300	$1,5 \times 10^{29}$	$4,9 \times 10^{15}$ anos
500	$1,3 \times 10^{39}$	$4,2 \times 10^{25}$ anos

Quadro 2 - Tempo Estimado de Fatoração de "n" , fonte [RSA1978]

Pela natureza dessa técnica de ataque, ao longo do tempo, vêm crescendo as chances de quebra devido a dois fatores, o aumento da capacidade computacional dos computadores e o desenvolvimento de melhores algoritmos de fatoração.

Até meados dos anos 90, a técnica mais utilizada de fatoração de "n" era baseada no algoritmo chamado de "*Quadratic Sieve*" [STALLINGS2003], ou "Crivo Quadrático" em Português

A quebra de uma chave RSA de 130 dígitos decimais foi levada a cabo por uma equipe de pesquisadores. Ao longo do ano de 1994 lançou mão de um novo algoritmo chamado de "*Generalized Number Field Sieve*". Esse novo algoritmo de fatoração permite fatorar um número da ordem de 129 dígitos decimais com apenas vinte por cento do esforço computacional do "Quadratic Sieve" [STALLINGS2003].

A fim de enfrentar possibilidades, além do natural aumento do tamanho das chaves, outras práticas recomendáveis vêm sendo sugeridas pelos pesquisadores [STALLINGS2003]:

- Os números "p" e "q" devem diferir em comprimento somente em alguns dígitos.
- Os números "p" e "q" devem conter um grande fator primo.
- O maior divisor comum de (p-1) e (q-1) deve ser pequeno.

Por não tratar diretamente da descoberta de $\phi(n)$, a presente técnica de ataque não demonstraria aumento ou decréscimo de operações computacionais entre o RSA tradicional e a versão alterada proposta no presente artigo.

Pode-se concluir também que, da dificuldade de fatorar "n", gera-se toda a segurança do algoritmo RSA. Se "n" for facilmente fatorável, a quebra da cifra torna-se trivial.

5.2 Determinação direta de $\phi(n)$ no RSA tradicional

O valor de $\phi(n)$ deve ser razoavelmente grande a fim de não se tornar um ponto de fraqueza a ser explorado por eventuais atacantes do RSA.

Por ser produto de dois números pares, $\phi(n)$ é imediatamente identificável como divisível por quatro.

Essa característica pode eventualmente ser explorada em um algoritmo de teste, que vai precisar testar somente um em cada quatro número num determinado espaço compreendido entre o ponto inicial de pesquisa e (n-1).

Considerando os dados abaixo:

p	q	n	$\phi(n)$	n/2	$\phi(n)$ - (n/2)
3	3	9	4	4,5	-0,5
3	5	15	8	7,5	0,5
3	7	21	12	11,5	0,5
5	5	25	16	12,5	3,5
3	11	33	20	16,5	3,5
5	7	35	24	17,5	6,5
7	7	49	36	24,5	11,5
5	11	55	40	22,5	17,5
7	11	77	60	38,5	21,5

Tabela 3 - Comparação entre $\phi(n)$ e n/2
- Calculado pelo autor

Pode-se facilmente perceber que, quanto maiores os fatores "p" e "q", maior a diferença entre o valor de $\phi(n)$ e n/2.

Pode-se considerar, então, o valor de n/2 como um ponto razoavelmente seguro de início de pesquisa, para todo "p" e "q" maiores que três. Com esse ponto de partida, estamos automaticamente descartando a metade inferior, entre 1 e (n/2-0,5) dos valores possíveis de serem assumidos por $\phi(n)$.

Somado ao fato de $\phi(n)$ ser múltiplo de quatro, por ser o produto de dois pares, o que nos leva a testar somente um a cada quatro números.

Os princípios acima podem ser aplicados no seguinte algoritmo, que usa um ataque de texto conhecido, para testar a validade de $\phi(n)$:

```
função testa_phi (e, n, C, M)
// Definição das variáveis
contador_phi := 0
Inicio := n/2 – 0,5
Phi(n) :=0
Finaliza := false
// Ponto de inicio - testa variável Inicio ate achar o primeiro divisível por 4
Repita
Se(Inicio mod 4) <> (0 então)
    Inicio := inicio +1
Até que Inicio mod 4 = 0
Phi(n) := Inicio
// Nesse ponto , iniciamos a varredura, para cada Phi(n) múltiplo de 4 acha-se um "d" e
//valida confrontando a decifragem com o texto claro induzido
Enquanto (finaliza = false) ou (Phi(n) < n) faca
Inicio
calcula_d (e, Phi(n))
Se C^d mod n = M
    Então
    Início
    Retorna Phi(n)
    Retorna d
    finaliza := true
    Fim
Senão
    Início
    Phi(n) := Phi(n) +4
    Contador_phi := Contador_phi +1
    Fim
Fim
Contador_ataque_phi := contador_d * contador_phi
```

O algoritmo recebe um texto às claras e sua respectiva cifração. A seguir, procedendo à descoberta de todos os possíveis valores de $\phi(n)$, iniciando em um valor aproximado da metade de n, que é comprovadamente menor que $\phi(n)$.

A quebra sugerida de $\phi(n)$ implica testar todos os múltiplos de quatro existentes entre o valor inicial e "n". Na prática, isso implica verificar um em cada quatro números existentes no intervalo [n/2, n], que pode ser definida como:

UT = (n – (n-1)/2)/4, onde UT significa universo do teste

5.3 Determinação Direta de "z" no RSA Alterado

Em contrapartida, em sua versão alterada, o algoritmo apresenta um universo de teste maior, visto que, ao substituir $\phi(n)$ por "z" temos que examinar todos os números impares, gerando um universo de teste aproximadamente duas vezes maior que o do algoritmo.

UT = (n – (n-1/)2)/2

Alterando o algoritmo anteriormente citado, tem-se :

```
função testa_phi_alterado (e, n, C, M)
// definição das variáveis
contador_phi_alterado := 0
Inicio := n/2 – 0,5
z :=0
Finaliza := false
// Ponto de inicio - testa variável Inicio ate achar o primeiro impar
Repita
Se(Inicio mod 2) =  0 então
    Inicio := inicio +1
Até que Inicio mod 2 = 0
z:= Inicio
// Nesse ponto , iniciamos a varredura
// Para cada Z ímpar acha-se um "d" e valida confrontando a decifragem
// com o texto claro induzido
Enquanto (finaliza = false) ou (z < n) faca
Inicio
calcula_d (e, Phi(n))
Se C^d mod n = M
    Então
    Inicio
    Retorna z
    Retorna d
    finaliza := true
    Fim
Senão
    Inicio
    z:= z+2
    Contador_phi_alterado := Contador_phi_alterado +1
    Fim
Fim
//Computa o total das iterações do ataque
Contador_ataque_phi_alterado := contador_d * contador_phi_alterado
```

5.4 Comparação da Eficiência na Determinação de ϕ(n) e "z"

Nesta seção iremos analisar comparativamente a eficiência dos dois algoritmos, iniciando pelas tabelas que mostram os resultados de testes efetuados em construtos de software desenvolvidos a partir dos algoritmos das seções 6.2 e 6.3.

Esses testes visam definir o número de iterações necessárias à quebra de determinado valor, seja ele ϕ(n) ou z, conforme a versão do algoritmo.

Importante citar que, por tratar-se de um ataque de texto conhecido, o valor final a ser atingido é critério de parada dos algoritmos e que não foram computadas as iterações relativas aos processos de cifragem ou decifragem, porque, conforme visto na seção 5, as operações têm o mesmo custo computacional, independendo da versão do algoritmo a ser testado.

p	*q*	*e*	*d*	*n*	*ϕ(n)*	*Cont Phi*	*Cont d*	*Cont Total*
11	13	7	103	143	120	13	722	9.386
19	23	13	61	437	396	45	7.156	322.020
47	53	25	2.105	2.491	2.392	287	315.688	90.602.456
103	107	55	2.359	11.021	10.812	1.326	6.881.110	9.124.351.860
181	199	91	14.491	36.019	35.640	4.408	71.474.564	315.059.878.112
173	211	89	28.409	36.503	36.120	4.468	61.431.168	274.474.458.624
191	223	97	5.653	42.593	42.180	5.221	83.670.784	436.845.163.264

Tabela 4 – Determinação de ϕ(n) no RSA - Calculado pelo autor

Analisando a tabela 5 abaixo, nota-se que a versão alterada do algoritmo tende a consumir consideravelmente mais iterações para sua quebra, em relação ao RSA, aumentado conforme aumentam o tamanho da chave e do "z" envolvido. A relação chegou a atingir a relação de 382,04 % para o sétimo conjunto de testes.

p	*q*	*e*	*d*	*n*	*z*	*Cont z*	*Cont d*	*Cont Total*
11	13	7	85	143	99	14	680	9.520
19	23	10	250	437	357	70	12.082	845.740
47	53	26	971	2.491	2.295	525	499.314	262.139.850
103	107	52	7.138	11.021	10.605	2.548	11.052.342	28.161.367.416
181	199	91	34.488	36.019	35.263	8.627	138.876.251	1.198.085.417.377
173	211	89	6.425	36.503	35.739	8.744	119.318.268	1.043.318.935.392
191	223	97	9.904	42.593	41.769	10.237	163.031.469	1.668.953.148.153

Tabela 5 – Determinação de z no RSA alterado - Calculado pelo autor

5.5 Determinação Direta de "d" no RSA Tradicional

A variável "d", que se usa como chave secreta do RSA, invariavelmente possui um valor ímpar, visto que ϕ(n) possui valor par, "e" sempre vai ser impar por não poder ter divisores comuns com ϕ(n).

Do mesmo modo, ϕ(n), por ser par, somente tem múltiplos pares pois não é possível formar um número ímpar a partir de multiplicação de inteiros, onde uma das parcelas é par.

Partindo dessa premissa, somente um "d" com valor ímpar pode satisfazer a equação:

$$e \times d \bmod \phi(n) = 1$$

Logo, uma tentativa direta de definição de "d" passa pelo teste de todos os valores ímpares iniciando em 3 e terminando em "d", ou seja:

$$UT = (d - 1)/2$$

Usando-se novamente um ataque de texto conhecido, tem-se o seguinte algoritmo:

```
função testa_d (e, n, C, M)
d := 1
cont_ataque_d := 0
finaliza := false
Enquanto (d < n) e (finaliza = false) faça
Início
Se C^d mod n <> M então
    Inicio
    d := d+2
    cont_ataque_d := cont_ataque_d+1
    Fim
    Senão finaliza = true
Fim
```

5.6 Determinação Direta de "d" no RSA Alterado

A variável "d", na versão alterada do RSA, pode assumir valores pares ou ímpares, dependendo do valor inicial de "e" e também do valor de "z".

Como se tem um "z" forçosamente ímpar, o produto "e x z" pode eventualmente assumir valores pares ou ímpares:

$$e \times d \bmod z = 1$$

Dessa maneira, a generalização de somente testar a metade dos números no intervalo entre 1 e n não pode ser aplicada à versão alterada do RSA.

$$UT = d - 1$$

Em termos de algoritmo torna-se necessário testar todos os valores de d, visto que o mesmo pode assumir valores pares ou ímpares:

```
função testa_d_alterado (e, n, C, M)
d := 1
cont_ataque_d_alt := 0
finaliza := false
Enquanto (d < n) e (finaliza = false) faca
Início
    Se C^d mod n <> M então
    d := d+1
    cont_ataque_d_alt := cont_ataque_d_alt+1
Senão finaliza = true
Fim
```

5.7 Comparação da Eficiência na Determinação direta de "d"

Os resultados das tentativas determinação direta de "d" para o RSA tradicional e alterado encontram-se na tabela abaixo:

e	*n*	*d*	*Cont ataque d*	*d alt*	*Cont ataque d alt*
7	143	103	51	85	84
13	437	61	30	250	249
25	2.491	2.105	1.052	971	970
55	11.021	2.359	1.179	7.138	7.137
91	36.019	14.491	7.245	34.488	34.487
89	36.503	28.409	14.204	6.425	6.424
97	42.593	5.653	2.826	9.904	9.903

Tabela 6 – Determinação de "d" nos RSA tradicional e alterado - Calculado pelo autor

Face o exposto, pode-se concluir dos dados preliminares calculados pelo autor, que a versão alterada do RSA demandou um número de iterações necessárias razoavelmente maior que o RSA tradicional.

6. Resultados Obtidos e Limitações do Trabalho

O presente artigo analisou as diversas tarefas necessárias ao funcionamento e ataque de um criptossistema de chave pública e privada, usando como base o RSA e sua versão alterada.

Das tarefas necessárias à geração de chaves, podemos averiguar que tanto o RSA quanto sua versão alterada tiveram uma diferença percentual, nos dados tabulados neste artigo, da ordem de 5,32 por cento a mais na versão alterada.

Para fins de cifragem e decifragem, demonstrou-se que ambos os algoritmos possuem o mesmo desempenho, pois usam as mesmas operações de exponenciação e módulo. O maior ou menor número de operações necessárias vai ser ditado somente pelos expoentes a serem usados.

No que tange ao ataque do RSA por fatoração de "n", não se observou também diferença entre os dois algoritmos visto que a fatoração de "n" se dá exatamente da mesma maneira para ambos e que a descoberta dos valores de $\phi(n)$ e z é trivial, uma vez fatorado o número "n".

Com relação ao ataque por determinação de $\phi(n)$, observou-se, nos dados preliminares, que a versão alterada do RSA apresentou uma maior quantidade de iterações com uma diferença gradativamente maior conforme valor de n aumenta, chegando a atingir um total de iterações 382,04% maior no RSA alterado.

Para o ataque por determinação direta de “d”, observou-se que o RSA tradicional precisa testar somente valores ímpares de “d” ao passo que sua versão alterada demanda da verificação de todos os valores possíveis de “d”, sejam pares ou ímpares. Dessa maneira, foram necessárias até 222,87% iterações a mais no RSA alterado.

O escopo do presente artigo limitou-se ao teste e experimentação, em ambiente altamente controlado e induzido, de uma versão alterada do RSA. Essa versão alterada não pode ser usada efetivamente como criptossistema, visto que os dados decifrados não se apresentam os mesmos quando do ciframento.

Por fim, a utilização da técnica de ataque de texto conhecido, apesar de ter sido empregada para facilitar a identificação de quebra de determinada chave, nem sempre é empregável em situações de uso, o que tem impacto direto nas medições que eventualmente feitas em ambiente de real utilização.

7. Conclusões

O presente artigo levou a efeito um breve estudo do criptossistema RSA, enfocando características básicas, geração de chaves e ataques matemáticos mais comuns.

Para fins de experimentação, foi sugerida uma alteração no algoritmo RSA, mais especificamente uma alteração no valor da totiente de Euler, o que teve conseqüências no desempenho da geração de chaves, aumentando em torno de seis por cento o total de iterações necessárias à geração de um par de chaves no algoritmo alterado.

No que tange aos ataques matemáticos, observou-se um aumento expressivo do esforço necessário ao sucesso de dois dos três ataques estudados, sendo eles a determinação direta de $\phi(n)$ da ordem de trezentos e oitenta por cento maior na versão alterada e a determinação direta de “d”, atingindo um aumento da ordem de duzentos e vinte por cento.

Demonstrou-se que tanto para o ataque por fatoração de n e quanto para as atividades de cifragem e decifragem não se observam alterações consideráveis de desempenho entre o RSA e sua versão alterada ora proposta nesse artigo.

8. Bibliografia

[BONEH1999] BONEH, Dan, “Twenty years of attacks on the RSA cryptosystem”, In Notices of the American Mathematical Society (AMS), Vol. 46, No. 2, pp. 203--213, 1999

[LUCCHESI1986] LUCCHESI, Cláudio Leonardo, “Introdução à Criptografia Computacional”, Editora da Unicamp / Editora Papirus, Campinas, 1986

[RSA1978] RIVEST, Ron, SHAMIR, Adi, ADELMAN, Leonard, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”, Commun. of the ACM, 21:120-126, 1978

[RSAES-OAEP2000] RSA Security Inc, “RSAES-OAEP Encryption Scheme Algorithm specification and supporting documentation”, RSA Laboratories, Bedford, USA, 2000

[SANTOS2000] SANTOS, José Plínio de O, “Introdução à Teoria dos Números”, Instituto de Matemática Pura e Aplicada CNPq, Rio de Janeiro, 2000

[STALLINGS2003] STALLINGS, William “Cryptography and Network Security”, Prentice Hall, New Jersey, USA, 2003

[TERADA2003] TERADA, Routo, RASMUSSEN, Mads “Vulnerabilidades do RSA”, http://www.opencs.com.br/infocenter/arquivo/Seguranca_e_Privacidade/Vulnerabilidades.pdf