Criptografia assimétrica – Multi-Primos

1. RSA com múltiplos primos

Este texto destina-se a explorar o uso de multi-primos no algoritmo RSA para processamento um pouco diferente do padrão tradicional.

Obs. Neste contexto definimos como 'RSA Tradicional' o uso das expressões conhecidas:
Cifração = m^e mod n ; Decifração = m^d mod n

O objetivo de explorar o emprego de múltiplos números primos é a obtenção de **chaves maiores** e o **aumento da segurança.**

Os aspectos que se destaca são relacionados ao:

- Tamanho das chaves
- Aumento da segurança
- Tempo de processamento compatível com o nível tecnológico atual

No RSA tradicional usam-se dois primos e o tamanho da chave determinado pelo 'n', e este é calculado pelo produto desses primos. Com multi-primos, o número 'n' é calculado pelo produto de todos os primos escolhidos. A quantidade de primos deve ser definida de acordo com o tamanho desejado para a chave, de forma que esta não se torne vulnerável.

Como exemplo apresenta-se um raciocínio lógico citando chaves de 2048, 4096 e 8192 bits. O objetivo deste exemplo é a comparação de chaves de 4096 (RSA tradicional) com as de 8192 bits (RSA mult-primos), onde além do aspecto segurança é fundamental o tempo de processamento da decifração. Observa-se que o tempo de processamento para a geração das chaves não é determinantemente significativo, desde que viável, por ser necessário realizá-lo apenas uma vez para cada usuário (entidade qualquer: um usuário comum, um computador, um sistema, ...), ou na ocasião que haja necessidade de troca de chave, porém não se deve ignorar o citado tempo de criação de chaves.

A cifração não é considerada nesta comparação por se tratar de processos idênticos para o uso de dois ou mais primos.

a) Tempo de processamento e tamanho de chaves

No processamento tradicional para a decifração com chave 4096 bits, proveniente de dois primos, P_1 (2006 bits) e p_2(2090 bits) o tempo de processamento é **0,20 seg**

No processamento para a decifração com chave 8192 bits, proveniente de cinco primos, p_i.: P_1 (2006 bits), P_2 (1027 bits), P_3 (1023bits), P_4 (2046 bits) e P_5 (2090 bits) é **0,09 seg**

Portanto, o tempo de decifração com uma chave de 8192 bits (multi-primos) é de cerca de 50% do tempo gasto na decifração (tradicional) com uma chave de 4096 bits.

b) Segurança

Sabe-se que a segurança consiste na dificuldade de fatorar o número '**n**', produtos de primos. No RSA tradicional, basta encontrar um dos primos o outro é calculado e assim o '**n**' estará fatorado.

Para a chave de 4096 citada, se encontrado de alguma forma o p_1 (2006 bits), a chave estará quebrada pois o $p_2 = n / p_1$.

Num sistema multi-primos, com 'k' primos, necessita-se encontrar 'k – 1' primos, pois apenas o último pode ser calculado.

No caso da chave acima com 8192 bits, se encontrar p_1 (1021 bits), ainda não se tem indicativos sobre os demais primos. Apenas pode-se reduzir o valor de 'n' computando, $n1 = n / p_1$, para procurar o p_2.

Se eventualmente encontrar também p_2, novamente pode-se reduzir o valor 'n1' computando, $n2 = n1 / p2$.

Esse processo precisa ter continuidade até encontrar o penúltimo p_i e tornar-se possível o cálculo do último primo.

Considerando as chaves citadas e os primos componentes das mesmas, a quebra da chave de 8192 bits requer um esforço computacional equivalente à quebra de duas chaves de 2048 bits e mais duas chaves de 4096 bits, tendo em vista não se conhecer métodos eficientes para determinação em paralelo de primos grandes.

Isto é, o esforço computacional para:

- encontrar p_1 (1023 bits) é o equivalente ao esforço para quebrar uma chave com '**n'** bits, onde $\mathbf{n = p_1 \times p_x}$ e $p_1 < \mathbf{p_x}$. Logo, '**n**' poderá seá um número da ordem de 2048 ou superior;
- encontrar p_2 (1027 bits) também é equivalente a quebra de outra chave da mesma ordem da anterior;
- de forma similar encontra p3 e p4 equivalem a quebra de duas chaves de 4096 e
- cálculo do p5: $p5 = n / (p_1 \times p_2 \times p_3 \times p_4)$

Obs. A condição acima "$p_1 < px$" deve-se a uma estratégia adequada para procurar os divisores de 'n', quando este possui apenas dois fatores primos. A estratégia consiste em procurar a partir da raiz de 'n' o fator menor que a raiz. Evidentemente, o outro que está mais distante da raiz será calculado.

c) Considerações

Em vista do exposto, com o exemplo do uso de chaves com 4096 e 8192 bits, pode-se concluir que a utilização de mult-primos possibilita o emprego de chaves maiores, melhor nível de segurança e menor tempo de processamento na decifração.

O emprego de mult-primos possibilita uma grande variedade de combinações de primos com diferentes tamanhos de forma que atenda a segurança almejada e um tempo de processamento adequado.

A Tabela seguinte mostra os tempos de processamento em segundo para vários tamanhos de chaves e diversas quantidades de primos.

Tam chaves / Processos	4096	6400		8192			16384		32768		65536	
	2P	2P	6P	2P	5P	13P	2P	26P	2P	40P	2P	103P
Criação de chaves	11,9	76,4	4,29	211,1	13,54	3,70	9321,75	7,15	Vários dias	21,97	vários anos	25,63
Cifração	0.0*	0.0*	0.0*	0.01	0.01	0.01	0,015	0,015	0,078	0,078	0,281	0,281
Decifração Tradicional	0,22	0,72	0,753	1,470	1,470	1,470	10,704	10,704	79,41	79,41	604,12	604,12
Decifração Mult-primos	-	-	0,031	-	0,108	0,015	-	0,062	-	0,187	-	0,453

Razão $= \frac{Decif\ Tradicional}{\text{Decif Mult}-\text{primos}}$			24,3		13,6	98,0		172,64		424,65		1333,5

Obs.

- Os tempos referem-se às médias, devido à variação na duração dos processos.
- nP → significa n números primos
- Nas colunas '2P' constam os dados do processamento do RSA Tradicional
- As cifrações foram efetuadas com expoente 'e' variando de 17 a 24 bits e msg com tamanho próximo ao de 'n'.
- Para chaves com até 4096, não se verifica a conveniência de empregar mais de dois primos. O uso de vários primos pode comprometer a segurança. O tempo da decifração, para esse processamento, está minimizado com a aplicação do TRC (Teorema do Resto Chinês).
- Referente ao tempo de processamento para o cálculo das chaves com:
  - n = 16384 e sendo '**n'** o produto de**:**
    - **dois primos → 2 horas e 35 minutos**
    - **26 primos → 7,15 segundos**
  - n = 32384 e sendo '**n'** o produto de
    - **dois primos → não foi possível calcular. Estima-se que o processamento demore cerca de 10 dias**
    - **40 primos →21,97 segundos**
  - n = 65536 e sendo '**n'** o produto de
    - **dois primos → não foi possível calcular. Estima-se que o processamento demore cerca de 5 anos**
    - **103 primos →45,63 segundos**
- A última linha da tabela (Razão $= \frac{Decif\ Tradicional}{\text{Decif Mult}-\text{primos}}$) mostra o quanto a velocidade de decifração mult-primos é maior que a tradicional.

2. Processamento com mult-primos

Criptografia assimétrica com chave grande e com boa velocidade pode ser obtida com expressão exponencial modular utilizando:

a) Implementação otimizada do algoritmo de Euclides estendido;
b) Teorema Chinês do Resto;
c) Múltiplos primos;
d) Cifração na forma do RSA original;
e) Decifração baseada:
   - Teorema do Resto Chinês (TRC)
   - Resolução do TRC pelo algoritmo de Gauss (1982) e de Garner(1984)
   - Inovação: Chave pública tratada, em parte, na própria geração.

3. Especificação do Algoritmo para geração de chaves

a) Para a geração de chaves

- Escolha da quantidade de primos.

Esta escolha deve ser adequada ao tamanho desejado para as chaves, de forma que não contemple primos pequenos e contenha dois primos grandes.

Neste contexto estão sendo considerados primos pequenos os com menos de 1024 bits e primos grandes com mais de 2048 bits.

Exemplo:

a) Para chave de 1024, 2048 ou 4096 bits, não usar mais de dois primos.
b) Para chaves de 16384 bits, usar pelo menos dois primos com cerca de 2048 bits e os demais podem ser menores desde que não sejam pequenos. Assim, a quantidade máxima de primos poderá, possivelmente, chegar entorno de 14.
c) Para chaves de 32768 bits, usar pelo menos dois primos com cerca de 2048 bits e os demais podem ser menores desde que não sejam pequenos. Assim, a quantidade máxima de primos poderá, possivelmente, chegar entorno de 30.

- Geração dos primos.

A geração pode ser efetuada por qualquer método. Por exemplos, usando o algoritmo de Miller-Rabin para encontrar primos estatísticos, com alta probabilidade.

- Cálculo de n (módulo)

$n = \prod_{i=1}^{j} p_i$; onde: j = quantidade de primos; p_i = primo i

- Cálculo do (FI) de n: Φ(n)

$\Phi(n) = \prod_{i=1}^{j} (p_i - 1)$; p_i = primo i ; j = quantidade de primos

- Cálculo do Lambda de n: λ(n)

$\lambda(n) = mmc\ (p_i - 1)$; p_i = primo i

O λ(n) sempre será menor que Φ(n), sendo no pior caso igual à metade deste. O mmc(p_i - 1) é fácil de ser cálculo quando se tem dois primos, mesmo que estes sejam grandes, entretanto é difícil ou inviável quando se tratam de mais de dois primos. Observa-se que se tratando de números grandes não é viável utilizar método de fatoração.

Para dois números, (p_1 -1) e (p_2 -1), aproveita-se da propriedade seguinte.

$$\lambda(n) = mmc(p1 - 1, p2 - 1) = \frac{(p1 - 1)\ x\ (p2 - 1)}{mdc(p1 - 1, p2 - 1)}$$

Nota-se que o mdc($p_1 - 1$, $p_2 - 1$) pode ser calculado de forma rápida com o algoritmo de Euclides.

- Cálculo do Alfa de n: α(n)

Se Qt_primos = 2 ; onde Qt_primos = quantidade de primos

Então $\alpha(n) = \lambda(n)$

Senão $\alpha(n) = \frac{\Phi(n)}{2^{Qt\_primos-1}}$

Neste cálculo considera-se o fato do Φ(primo) ser par.

- Escolha do expoente *e*.

Pode ser qualquer número co-primos com FI de n (os pares não atendem esse requisito)
A empresa RSA recomenda para expressão modular (cifração) o expoente 65537. Entretanto, não há inconveniência em usar expoentes maiores, desde que a implementação permita um processamento compatível com a aplicação. Sugere-se um primo estatístico de 32 bits, não divisor de FI de n.

- Cálculo da chave privada.

O expoente privado d pode ser calculado com Φ(n) ou com λ(n) ou ainda com o α(n). Em vista da relação Φ(n) > α(n) ≥ λ(n) e da dificuldade da determinação λ(n) para mais de dois primos, faz-se α(n) = λ(n) quando possível o cálculo deste último. Em geral, o menor valor de d é o calculado com o α(n). Logo, $d = e^{-1} \bmod \alpha(n$, acarretando consequentemente uma maior eficiência na decifração.

- Para a decifração tradicional:
  Chave privada = (d, n), não depende da quantidade de primos

- Para a decifração rápida:
  A decifração é bem mais lenta que a cifração devido ao tamanho do expoente. Esse processo torna-se mais rápido com o desmembrado do expoente '**d**' de forma que se tenha um d_i para cada p_i , usando o $\Phi(p_1)$ e calculando $dp_i = d \bmod (p_i - 1)$.
  Para esse processamento a chave privada é composta por '**d'** e todos o primos, $\mathbf{p_i}$.
  Na decifração pode ser usado o Teorema do Resto Chinês (CRT) com o algoritmo de Garner ou com o de Gauss. Entretanto, apesar de ser mais rápido do que a decifração tradicional, ainda necessita realizar muitos cálculos a cada decifração.
  Chave privada = $(d, p_1, p_2, \ldots , p_n)$.

  Um procedimento adotado para minimizar o tempo de decifração é antecipar para o cálculo da chave todos os procedimentos que não envolvem o texto a decifrar. A melhora deve-se ao fato de efetuar tais cálculos apenas uma vez e não a cada decifração.
  Assim, realizam-se os seguintes cálculos na definição da chave:

  $MP_k = \prod_{i=1}^{j} p_i$; para $k \neq i$ e $k = 1,2 \ldots, j$; j = quantidade de primos

  $INVp_i = MP_i * ((MP_i)^{-1} \bmod p_i)$; $i = 1,2 \ldots, j$; j = quantidade de primos

  Chave Privada = $(n, p_i, dp_i, INVp_i)$; para i = 1, 2, ... j ; j = quantidade de primos

b) Cifração

Tx : texto em claro (ex: chave simétrica)
Tc : texto cifrado

$Tc = (Tx)^e \bmod n$; *Esta expressão continua válida independentemente da quantidade de primos.*

c) Decifração

- **Decifração tradicional**

  $Tx = (TC)^d \bmod n$; Esta expressão continua válida independentemente da quantidade de primos.

- **Decifração Mult-primos**

  Para i = 1 até j ; J = quantidade de primos

  $mp_i = (Tc)^{dpi} \bmod p_i$

  $$Tx = \left(\sum_{i=1}^{j} (mp_i \text{ x } INVp_i) \right) mod\ n$$

  Decifrado: Tx

  Com este procedimento se conseguiu os menores tempos nos processamentos assimétricos e a viabilidade do uso de chaves muito superiores as tradicionalmente utilizadas.

4. Exemplo com quatro primos e números pequenos para facilitar os cálculos

a) Geração das chaves

- Primos gerados: p1 = 5 ; p2 = 23 ; p3 = 41 e p4 = 67
- Cálculo do módulo: n = 5 x 23 x 41 x 67 = 315905
- Cálculo do $\alpha(n)$ = $\Phi(n)$ / (2 ^ 3)= (4 x 22 x 40 x 66) / 8 = 29040
- Escolha do expoente: e = 17
- Cálculo do d:

  $d = e^{-1} \bmod \alpha(n) = 17^{-1} \bmod 29040 = 6833$

- Cálculo dos dp_i

  $dp_1 = 6833 \bmod (5 - 1) = 1$

  $dp_2 = 6833 \bmod (23 - 1) = 13$

  $dp_3 = 6833 \bmod (41 - 1) = 33$

  $dp_4 = 6833 \bmod (67 - 1) = 35$

- Cálculo dos MP_i

  MP1 = 23 x 41 x 67 = 63181

  MP2 = 5 x 41 x 67 = 13735

  MP3 = 5 x 23 x 67 = 7705

  MP4 = 5 x 23 x 41 = 4715

- Cálculo dos $INVp_i$

  $INVp1 = MP1 * ((MP1)^{-1} \bmod p1) = 63181 \text{ x } (63181^{-1} \bmod 5) = 63181 * 1 = 3563181$

  $INVp2 = MP2 * ((MP2)^{-1} \bmod p2) = 13735 \text{ x } (13735^{-1} \bmod 23) = 13735 \text{ x } 6 = 82410$

  $INVp3 = MP3 * ((MP3)^{-1} \bmod p3) = 7705 \text{ x } (7705^{-1} \bmod 41) = 7705 \text{ x } 27 = 208035$

  $INVp4 = MP4 * ((MP4)^{-1} \bmod p4) = 4715 \text{ x } (4715^{-1} \bmod 67) = 4715 \text{ x } 59 = 278185$

- Chaves:

  Pública = (e, n) = (17, 315905)

  Privada = {n, {p1,p2,p3,p4},{dp1,dp2,dp3,dp4 },{INVp1, INVp2, INVp3, INVp4}}

  Privada = {315905, {5,23,41,67},{1,13,33,35},{63181,82410,208035,278185}}

b) Cifração: Tx (msg a cifrar)

Tx = 1000 (Msg em claro)

$Tc = Tx^{e} \bmod n$

$Tc = 1000^{17} \bmod 315905 = 178770$

Msg cifrada = 178770

c) Decifração: Tc = 178770

$mp_i = (Tc)^{dpi}, \bmod p_i$

$mp_1 = 178770^{1}, \bmod 5 = 0$

$mp_2 = 178770^{13}, \bmod 23 = 11$

$mp_3 = 178770^{33}, \bmod 41 = 16$

$mp_4 = 178770^{35}, \bmod 67 = 62$

$Tx = (mp_1 \times INVp_1 + mp_2 \times INVp_2 + mp_3 \times INVp_3 + mp_4 \times INVp_4) \bmod n$

$Tx = (0 \times 63181 + 11 \times 82410 + 16 \times 208035 + 62 \times 278185) \bmod 315905$

Tx = 1000 (msg decifrada corretamente)

Recursos usados:

Característica do computador usado no processamento

- Processador – Intel I7, 2,5 MHz
- SO – Windows 64

Obs: números primos citados neste texto são primos estatísticos