Reconstrução da Chave Privada RSA Multi-primo

Reynaldo C. Villena
(reynaldo@ime.usp.br)

Orientador: Routo Terada

Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

Setembro de 2012

Agenda

1 - Conceitos Básicos

PKCS - Public Key Cryptography Standards

- O PKCS é grupo de padrões devenvolvido pelos *laboratorios RSA*[1]
- O PKCS é grupo de padrões devenvolvido que contem especificações para acelerar a implementação e desenvolvimento dos algoritmos dos criptossistemas de chave pública.

Onde

O PKCS #1 é o padrão e contém definições básicas e recomendações para a implementação do criptossistema RSA.

- O RSA é o criptossistema de chave pública mais usado e implementado até a data.
- Seu nome é derivado das iniciais dos seus criadores: Ron (R)ivest, Adi (S)hamir e Len (A)dleman.
- Foi criado em agosto de 1977 no MIT e publicado em fevereiro de 1978.

[1] Empresa dedicada à criptografia e ao software de segurança

PCKS # 1- RSA (Geração da equação RSA)

1.- Geração das chaves

Usando algum algoritmo determinamos u primos distintos($u \geq 2$) e calculamos $N = \prod_{i=1}^{u} r_i$. A seguir, escolhemos um e co-primo a $\phi(N) = \prod_{i=1}^{u} (r_i - 1)$ e calculamos o valor de d tal que

$$e.d = 1 \mod \phi(N)$$

Dados extras

- Se o valor u for 2 ($u = 2$) então é conhecido como **Criptossistema RSA básico ou standart** e se for maior igual a 3 ($u \geq 3$) então é conhecido como **Criptossistema RSA multi-primo**

PKCS #1 - RSA (Chave Pública)

$$e.d = 1 \mod \phi(N)$$

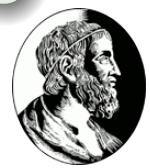
Chave Pública RSA

Está composta por duas variáveis $pk\langle N, e\rangle$.

Algoritmo de encriptação

Para criptografar uma mensagem $M \in \mathbb{Z}_N$ debemos utilizar a chave pública $pk\langle N, e\rangle$ e calculamos o criptograma

$$C \leftarrow M^e \mod N.$$



PKCS #1 - RSA (Chave Privada)

- O padrão PKCS # 1 especifica duas representações para a Chave Privada.

$$e.d = 1 \mod \phi(N)$$

1° Representação da Chave Privada RSA

Está composta por duas variáveis $sk\langle N, d\rangle$.

Algoritmo de decifração

Para decifrar o criptograma $C \in \mathbb{Z}_N$ utilizamos a chave privada $sk\langle N, d\rangle$ e calculamos a mensagem

$$M \leftarrow C^d \mod N$$

PKCS #1 - RSA (Chave Privada)

$$N = \prod_{i=1}^{u} r_i \qquad e.d = 1 \mod \phi(N)$$

2° Representação da Chave Pública RSA

Está composta pelas seguintes variáveis $sk\langle p, q, d_p, d_q, q^{-1}, \langle r_3, d_3, t_3\rangle, .., \langle r_u, d_u, t_u\rangle\rangle$.

$p = r_1$ $\quad e.d_p = 1 \mod (r_1 - 1)$ $\quad q.q^{-1} = 1 \mod p$

$q = r_2$ $\quad e.d_q = 1 \mod (r_2 - 1)$

Para $i = 3, ..., u$

$e.d_i = 1 \mod (r_i - 1)$

$t_i.R_i = 1 \mod r_i$, onde $R_i = r_1.r_2....r_{i-1}$

PKCS #1 - RSA (Chave Privada)

Algoritmo de decifração

Para decifrar o criptograma $C \in \mathbb{Z}_N$ utilizamos a chave privada $sk\langle p, q, d_p, d_q, q^{-1}, \langle r_3, d_3, t_3\rangle, .., \langle r_u, d_u, t_u\rangle\rangle$ e executamos o seguinte algoritmo.

$M_p \leftarrow C^{d_p} \mod p$
$M_q \leftarrow C^{d_q} \mod q$
$M \leftarrow ((M_p - M_q)q^{-1} \mod p)q + M_q$

$R_i \leftarrow r_1$
Para $i = 3, ..., u$ calculamos
  $M_i \leftarrow C^{d_i} \mod r_i$
  $R \leftarrow R * r_{i-1}$
  $M \leftarrow ((M_i - M)t_i \mod r_i)R + M$
Retornar M

- O algoritmo está baseado no TCR[2].
- Foi proposto por J-J. Quisquater and C. Couvreur em 1982.
- A utilização desse método foi muito popular já é muito mais rápido.

[2] Teorema Chines do Resto

PKCS #1 - RSA (Recomendação para implementação do RSA)

Representação ANS.1[3] das chaves RSA segundo ao padrão PKCS #1.

```
RSAPublicKey ::= SEQUENCE {
    modulus           INTEGER,  -- n
    publicExponent    INTEGER   -- e
}
RSAPrivateKey ::= SEQUENCE {
    version           Version,
    modulus           INTEGER,  -- n
    publicExponent    INTEGER,  -- e
    privateExponent   INTEGER,  -- d
    prime1            INTEGER,  -- p
    prime2            INTEGER,  -- q
    exponent1         INTEGER,  -- d mod (p-1)
    exponent2         INTEGER,  -- d mod (q-1)
    coefficient       INTEGER,  -- (inverse of q) mod p
    otherPrimeInfos   OtherPrimeInfos OPTIONAL
}
Version ::= INTEGER { two-prime(0), multi(1) }
    (CONSTRAINED BY {-- version must be multi if otherPrimeInfos present --})
OtherPrimeInfos ::= SEQUENCE SIZE(1..MAX) OF OtherPrimeInfo
OtherPrimeInfo ::= SEQUENCE {
    prime             INTEGER,  -- ri
    exponent          INTEGER,  -- di
    coefficient       INTEGER   -- ti
}
```

- Podemos observar que a chave privada é altamente redundante.

[3] Notação Sintatica Abstrata 1

Segurança do RSA

Segurança do RSA

A segurança do RSA está baseado no problema de fatoração de inteiros (N), o qual é um problema NP.

$$N \rightarrow \phi(N) \rightarrow d$$

- A recuperação da chave privada sk a partir da chave pública pk é um problema computacionalmente inviável.
- O algoritmo mais rápido para fatorar inteiros é conhecido como NFS[4].

[4] Number Field Sieve

2 - Objetivo

- Em 2008, J. A. Halderman publicou um artigo onde mostra que é possível a recuperação da informação (bits) graças à propriedade de remanência da memória DRAM (*ataques cold boot*).
- N. Heninger e H. Shacham apresentam um algoritmo de reconstrução (focado ao RSA básico) que faz uso da redundância da chave secreta RSA básico segundo ao padrão PCKS #1 para corrigir a chave privada, e assim, obter o seu valor correto.

Objetivos

- Estudar o comportamento do algoritmo de Heninger e Shacham no RSA multi-primo.

3 - Algoritmo de Heninger-Shacham

Algoritmo de reconstrução da chave privada

Do ataque *cold boot* podemos obter:

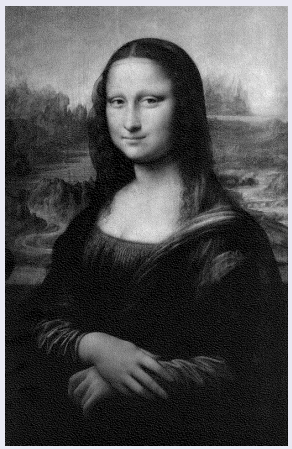
$$\tilde{sk}\ \langle \tilde{d}, \tilde{p}, \tilde{q}, \tilde{d}_p, \tilde{d}_q, \tilde{q}^{-1}, \langle \tilde{r_3}, \tilde{d_3}, \tilde{t_3} \rangle, .., \langle \tilde{r_u}, \tilde{d_u}, \tilde{t_u} \rangle \rangle \text{ com um } \delta^5.$$

Chave privada com δ

$\tilde{sk}$

$\xrightarrow{HS}$

Chave privada



sk

[5] Porcentagem de bits corretos

Redundância da chave privada

Redundância de dados que oferece a chave privada

$$sk = (N, e, p, q, d, d_p, d_q, q^{-1}, \langle r_3, d_3, t_3\rangle, .., \langle r_u, d_u, t_u\rangle\rangle)$$

segundo ao padrão PKCS # 1. Esses variáveis estão relacionadas matematicamente por:

$$N = \prod_{i=1}^{u} r_i \qquad e.d = 1 \mod \prod_{i=1}^{u}(r_i - 1)$$

$$e.d_1 = 1 \mod (r_1 - 1) \qquad r_2.r_2^{-1} = 1 \mod r_1$$

$$e.d_2 = 1 \mod (r_2 - 1) \qquad r_1.r_2.t_3 = 1 \mod r_3$$

$$\vdots \qquad\qquad \vdots$$

$$e.d_u = 1 \mod (r_u - 1) \qquad r_1.r_2...r_{u-1}.t_u = 1 \mod r_u$$

Redundância da chave privada

Redundância de dados que oferece a chave privada

$$sk = (N, e, p, q, d, d_p, d_q, q^{-1}, \langle r_3, d_3, t_3\rangle, .., \langle r_u, d_u, t_u\rangle\rangle)$$

segundo ao padrão PKCS # 1. Esses variáveis estão relacionadas matematicamente por:

$$N = \prod_{i=1}^{u} r_i \qquad e.d = 1 + k\prod_{i=1}^{u}(r_i - 1)$$

$$e.d_1 = 1 + k_1(r_1 - 1) \qquad r_2.r_2^{-1} = 1 + g.r_1$$

$$e.d_2 = 1 + k_2(r_2 - 1) \qquad r_1.r_2.t_3 = 1 + g_3.r_3$$

$$\vdots \qquad\qquad \vdots$$

$$e.d_u = 1 + k_u(r_u - 1) \qquad r_1.r_2...r_{u-1}.t_u = 1 + g_u.r_u$$

Redundância da chave privada

Redundância de dados que oferece a chave privada

$$sk = (N, e, p, q, d, d_p, d_q, q^{-1}, \langle r_3, d_3, t_3\rangle, .., \langle r_u, d_u, t_u\rangle\rangle)$$

segundo ao padrão PKCS # 1. Esses variáveis estão relacionadas matematicamente por:

$$N = \prod_{i=1}^{u} r_i \qquad e.d = 1 + k\prod_{i=1}^{u}(r_i - 1)$$

$$\begin{aligned} e.d_1 &= 1 + k_1(r_1 - 1) \\ e.d_2 &= 1 + k_2(r_2 - 1) \\ &\vdots \\ e.d_u &= 1 + k_u(r_u - 1) \end{aligned}$$

Calculando os valores de $k, k_1, ...k_u$

Analisando os valores $k, k_1, ...k_u$.

$$N = \prod_{i=1}^{u} r_i$$

$$e.d = 1 + k\prod_{i=1}^{u}(r_i - 1)$$

$$e.d_i = 1 + k_i(r_i - 1)$$

$$e, d \in \mathbb{Z}^*_{\phi(N)} \Rightarrow e, d < \phi(N) \Rightarrow k < e$$

$$e, d_i \in \mathbb{Z}^*_{\phi(r_i)} \Rightarrow e, d_i < \phi(r_i) \Rightarrow k_i < e$$

$$0 \leq k, k_1, ..., k_u < e$$

Aplicando mod e para calcular os valores $k, k_1, ...k_u$.

$$N = \prod_{i=1}^{u} r_i$$

$$0 = 1 + k\prod_{i=1}^{u}(r_i - 1)$$

$$0 = 1 + k_i(r_i - 1)$$

$$N\prod_{i=1}^{u} k_i = \prod_{i=1}^{u}(k_i - 1)$$

$$\prod_{i=1}^{u} k_i = k(-1)^{u-1}$$

Calculando os valores de $k, k_1, ...k_u$

Atribuímos valores a $\langle k_1, .., k_{u-1} \rangle$: e^{u-1} equações

$$N \prod_{i=1}^{u} k_i = \prod_{i=1}^{u} (k_i - 1)$$

$$\prod_{i=1}^{u} k_i = k(-1)^{u-1}$$

Onde o número de soluções possíveis para $\langle k, k_1, ..., k_u \rangle$:

$$\alpha(u) = \left\{ \begin{array}{ll} 0 & \text{se } u = 1 \text{ e } N \mod e = 1 \\ 1 & \text{se } u = 1 \text{ e } N \mod e <> 1 \\ (e-2)^{u-1} - \alpha(u-1) & \text{se } u > 1 \end{array} \right\}$$

$$\alpha(u) \leq (e-2)^{u-1}$$

Dados extras

Para o caso $u = 2$, Percival[?] formulou que existe e possíveis escolhas para $\langle k, k_1, k_2 \rangle$.

- E possível encontrar valores potenciais para o k?

Calculando os valores mais potenciais para k

- lembrar que $lg(d) \approx lg(N) = n$

Lema

Para um e pequeno, os $\frac{n}{u}$ bits mais significativos de d pode ser estimados eficientemente.

$$e.d = 1 + k\prod_{i=1}^{u}(r_i - 1)$$

caso u=2 $\quad e.d = 1 + k(N - r_1 - r_2 + 1)$
caso u=3 $\quad e.d = 1 + k(N - r_1.r_2 - r_1.r_3 - r_2.r_3 + r_1 + r_2 + r_3 - 1)$
caso u=4 $\quad e.d = 1 + k(N - r_1.r_2.r_3 - r_1.r_2.r_4 - r_1.r_3.r_4 - r_2.r_3.r_4 + ... + 1)$

Generalizando para o caso u temos:

$$e.d = 1 + k(N - \sum_{i=1}^{u}\frac{N}{r_i} + ... + (-1)^u) \quad (1)$$

Calculando os valores mais potenciais para k

$$d = \frac{1 + kN}{e} - \frac{k}{e}(\sum_{i=1}^{u} \frac{N}{r_i} - ... + (-1)^{u-1})$$

$$d = d_0 - d_1$$

$$|d1| \leq \frac{k}{e}\sum_{i=1}^{u} \frac{N}{r_i} \leq \sum_{i=1}^{u} \frac{N}{r_i} \approx u\frac{N}{r_i}$$

$$lg|d1| \approx lg(u) + lg(N) - lg(r_i)$$

$$lg|d1| \approx \frac{(u-1)n}{u} + lg(u)$$

- Se calculamos o valor de d_0 com o valor correto de k, vamos ter que d_0 tem os $\frac{n}{u} - lg(u)$ bits mais significativos de d.

(2)

Calculando os valores mais potenciais para k

- Temos $\tilde{d}$ com δn bits corretos

Calculamos o valor de d_0 para cada valor possível de $k(0 < k < e)$.

$$d_0 = \frac{1 + kN}{e}$$

d_0 tem o bits $\frac{n}{u}$ bits mais significativos de d.

- Os valores para k estão dados onde o d_0 e $\tilde{d}$ tenham a menor distancia hamming[6]

$$H = \sum_{i=\frac{(u-1)n}{u}}^{n} d_0[i] \oplus \tilde{d}[i]$$

onde $\tilde{d}[i]$ é conhecido

[6] Número de bits diferentes entre duas variáveis

Calculando $\langle k_1, ..., k_u \rangle$ a partir de k

Determinando os valores de $\langle k_1, ..., k_u \rangle$ a partir de k:

$$N\prod_{i=1}^{u} k_i = \prod_{i=1}^{u}(k_i - 1) \qquad \prod_{i=1}^{u} k_i = k(-1)^{u-1}$$

$$NK(-1)^{u-1} \equiv (k_1 - 1)(k_2 - 1)\prod_{i=3}^{u}(k_i - 1)$$

$$NK(-1)^{u-1} \equiv (-K(-1)^u k_2^{-1}\prod_{i=3}^{u}(k_i^{-1}) - 1)(k_2 - 1)\prod_{i=3}^{u}(k_i - 1)$$

caso u=2 $\quad -NK \equiv (-Kk_2^{-1} - 1)(k_2 - 1)$
caso u=3 $\quad NK \equiv (Kk_2^{-1}k_3^{-1} - 1)(k_2 - 1)(k_3 - 1)$
caso u=4 $\quad -NK \equiv (-Kk_2^{-1}k_3^{-1}k_4^{-1} - 1)(k_2 - 1)(k_3 - 1)(k_4 - 1)$

En conclusão temos uma equação quadrática com $u - 1$ variáveis.

Calculando $\langle k_1, ..., k_u \rangle$ a partir de k

$$NK(-1)^{u-1} \equiv (-K(-1)^u k_2^{-1} \prod_{i=3}^{u} (k_i^{-1}) - 1)(k_2 - 1) \prod_{i=3}^{u} (k_i - 1)$$

para calcular uma solução para k_2 atribuímos valores para $\langle k_3, ..., k_u \rangle$:

$$\begin{aligned} NK(-1)^{u-1} &\equiv (-K(-1)^u k_2^{-1} L - 1)(k_2 - 1)M \\ 0 &\equiv k_2^2 - k_2(K(-1)^u(NL^{-1} - M) + 1) - K(-1)^u M \\ 0 &\equiv ak_2^2 - k_2 b - c \end{aligned}$$

Para a solução da equação calculamos $D = b^2 - 4ac$

se $D = 0$ temos só uma solução $(2k_2 + b = 0)$

D for resíduo quadrático temos duas soluções $(2k_2 + b = \pm\sqrt{D})$

D não for resíduo quadrático não temos soluções

Depois de calcular o valor de k_2 podemos calcular o valor de k_1.

Calculando $\langle k_1, ..., k_u \rangle$ a partir de k

Sabemos que o número de equações para um k desconhecido esta dado por:

$$\alpha(u) \leq (e-2)^{u-1}$$

Supondo que temos a mesma quantidade de soluções $\langle k_1, ..., k_u \rangle$ para cada valor k $k \in \mathbb{Z}_e^*(0 < k < e)$, então temos para cada valor de k um promédio de soluções:

$$\beta(u) = \frac{\alpha(u)}{e-2} \leq \frac{(e-2)^{u-1}}{e-2}$$
$$\beta(u) \approx (e-2)^{u-2}$$

- Com o valor de k conhecido temos em promédio $(e-2)^{u-2}$ possíveis escolhas para $\langle k, k_1, ..., k_u \rangle$

Dados extras

Para o caso onde $u = 2$, foi analisado por Boneh[?]:

$$0 = k_2^2 - [k(N-1)+1]k_2 - k$$
$$0 = k + k_1 k_2$$

onde o número de possíveis escolhas para $\langle k, k_1, k_2 \rangle$ são 2.

Correção dos bits inferiores de:

Sabemos que r_i são primos portanto corrigimos os bit:

$r_1[0] = 1,\ r_2[0] = 1,\ \ldots,\ r_u[0] = 1$

seja $\tau(x)$ o maior expoente da potência de 2 tal que $2^{\tau(x)}$ divide x.

sabemos que $2|r_i - 1$, então temos que $2^{1+\tau(k_i)}|k_i(r_i - 1)$

$$ed_i = 1 \mod 2^{1+\tau(k_i)}$$
$$d_i = 1 \mod 2^{1+\tau(k_i)}$$

Agora podemos corrigir os $1 + \tau(k_i)$ primeiros bits de $\tilde{d}_i$ com 1. Usamos o mesmo conceito para corrigir $\tilde{d}$.

Algoritmo de Reconstrução

O algoritmo está baseado nas mudanças dos bits:

- Uma troca de valor no $r_i[j]$ afeta ao valor de $d_i[j + \tau(k_i)]$ e vice-versa.
- Uma troca de valor no qualquer $r_i[j]$ afeta ao valor de $d[j + \tau(k)]$

Portanto podemos definir o seguinte

$$\text{grupo}[j] = (r_1[j], ..., r_u[j], d[j + \tau(k)], d_1[j + \tau(k_1)], ..., d_u[j + \tau(k_u)])$$

onde o $\text{grupo}[0] = (1, ..., 1, d[\tau(k)], d_1[\tau(k_1)], .., d_u[\tau(k_u)])$

Algoritmo HS - Gerando possíveis soluções

Vamos supor que temos uma solução ate o grupo$[j-1]$. A ideia do algoritmo é gerar todos os possíveis soluções para o grupo$[j]$ a partir do grupo$[j-1]$.

$$\text{grupo}[0] \rightarrow \ldots \rightarrow \text{grupo}[j-1] \rightarrow \text{grupo}[j] \rightarrow \ldots \rightarrow \text{grupo}[\tfrac{n}{u}]$$

Para gerar todos as possíveis soluções para o grupo$[i]$ devemos fazer todas as atribuições possíveis para os bits

$$(r_1[j], \ldots, r_u[j], d[j+\tau(k)], d_1[j+\tau(k_1)], \ldots, d_u[j+\tau(k_u)])$$

- Então, temos 2^{2u+1} soluções para o grupo$[j]$?

Restrições

Restrições RSA temos:

$$N - \prod_{i=1}^{u} r_i' \equiv 0 \mod 2^j$$

$$k\prod_{i=1}^{u}(r_i' - 1) + 1 - ed' \equiv 0 \mod 2^{j+\tau(k)}$$

$$k_1(r_1' - 1) + 1 - ed_1' \equiv 0 \mod 2^{j+\tau(k_1)}$$

$$\vdots$$

$$k_u(r_u' - 1) + 1 - ed_u' \equiv 0 \mod 2^{j+\tau(k_u)}$$

- Com estas restrições o número de soluções para o grupo$[i]$ diminui a 2^{u-1}.

$$r_u[j] = F(r_1[j], ..., r_{u-1}[j], N[j])$$

$$2^{2u+1} \rightarrow 2^{u-1}$$

- Então, temos 2^{u-1} soluções para o grupo$[i]$?

Algoritmo HS - Número de soluções

Lembrando da chave privada $\tilde{sk}$ $\langle \tilde{d}, \tilde{p}, \tilde{q}, \tilde{d_p}, \tilde{d_q}, \tilde{q}^{-1}, \langle \tilde{r_3}, \tilde{d_3}, \tilde{t_3} \rangle, .., \langle \tilde{r_u}, \tilde{d_u}, \tilde{t_u} \rangle \rangle$ com um porcentagem de bits conhecidos.
Vamos denotar como bit fixo $s[j]$ se o bit i da variável s é conhecido, portanto todas nossas soluções para o grupo$[j]$ devem satisfazer as restrições com esse bit fixo(bits conhecidos).

$$\text{grupo}[j-1] \Rightarrow \text{grupo}[j]$$

- Em total vamos a ter 0,1,2,...2^{u-1} soluções dependendo de quantos bits fixos temos para o grupo$[j]$.

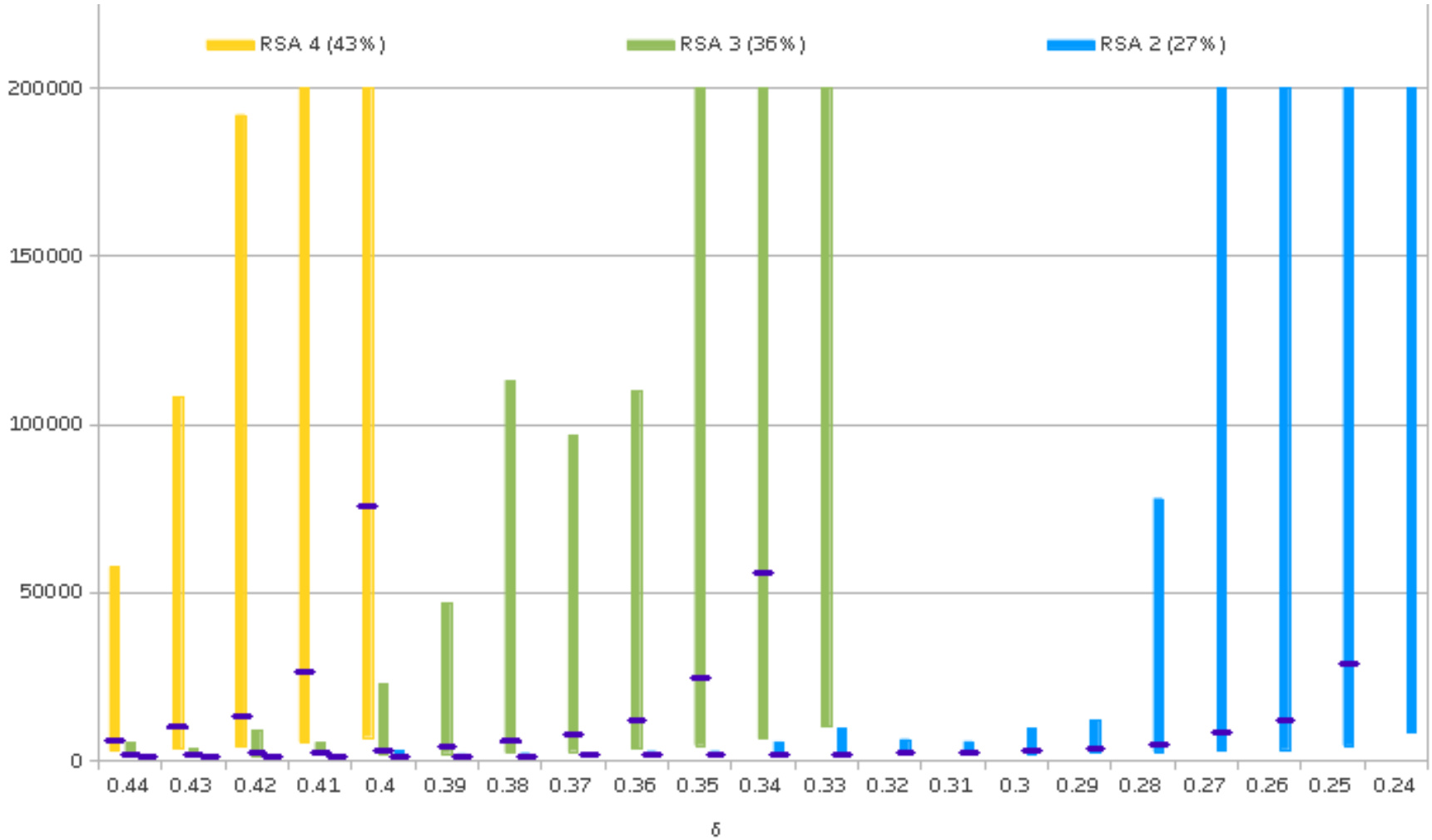
Implementação do algoritmo HS

- O algoritmo de reconstrução de HS foi implementado na linguagem C usando a biblioteca *Relic-toolkit* e testado sob um processador Intel Core I3 2.4 Ghz com 3 Mb de cache e 4 Gb de memória DDR3.
- Os experimentos foram feitos para chaves 2048 bits e para um $\delta \in [0.24, 0.44]$.
- Para cada chave de tamanho n foi gerado 25 criptossitemas e para cada criptossistema e cada δ foi gerado 50 chaves privadas com bits modificados.
- Todos os criptossistemas tinham o expoente de encriptação $e = 2^{16} + 1$.
- Os experimentos foram feitos para os criptossistemas RSA onde $2 \leq u \leq 4$.
- Os experimentos foram testados com os valores corretos de $\langle k, k_1, k_2, ..., k_u \rangle$ para evitar dados inecessários.

Tempo promédio onde o N foi fatorado com sucesso

	$N = pq$	$N = pqr_3$	$N = pqr_3r_4$
$\delta = 0.44$	0.047536(s)	0.195040	2.101040
0.43	0.049672	0.214640	3.469680
0.42	0.050328	0.269440	4.314320
0.41	0.004235	0.308880	9.469762
0.40	0.047660	0.392860	28.948557
0.39	0.058360	0.502500	-
0.38	0.060420	0.728760	-
0.37	0.064860	1.576420	-
0.36	0.068480	2.663820	-
0.35	0.071900	3.320801	-
0.34	0.077740	6.956159	-
0.33	0.087000	27.875515	-
0.32	0.096760	-	-
0.31	0.106080	-	-
0.30	0.124360	-	-
0.29	0.148960	-	-
0.28	0.213180	-	-
0.27	0.362280	-	-
0.26	0.636560	-	-
0.25	1.116501	-	-
0.24	10.376401	-	-

Número de candidatos analisados (n=2048 bits)



Quadro de comparações

- Seja u o número de fatores primos, r_i seja primos e e seja um primo pequeno.

$u =$	2	3	4
	$N = r_1 r_2$	$N = r_1 r_2 r_3$	$N = r_1 r_2 r_3 r_4$
Número de soluções para $\langle k_1, .., k_u \rangle$			
com k desconhecido	$\alpha(2) \leq e - 2$	$\alpha(3) < (e-2)^2$	$\alpha(4) < (e-2)^3$
com k conhecido	2	$e - 2$	$(e-2)^2$
Comportamento da algoritmo			
árvore 2^{u-1}-aría	$2^{2-1} = 2$	$2^{3-1} = 4$	$2^{4-1} = 8$
nível máximo (n/u)	$n/2$	$n/3$	$n/4$
Fatoração em TP[7]	$\delta = 2 - 2^{\frac{4}{5}} \approx 0.27$	$\delta = 2 - 2^{\frac{5}{7}} \approx 0.37$	$\delta = 2 - 2^{\frac{6}{9}} \approx 0.44$

Portanto enquanto o valor de u seja maior, vamos ter mais candidatos para $\langle k_1, .., k_u \rangle$ e o valor de δ também é maior.

[7] Tempo Polinomial

Calculo de $\langle k_1, .., k_u \rangle$

- Supondo que temos um processador que trabalha a 1 MIPS e que em cada instrução vamos calcular uma possível escolha para $\langle k, k_1, .., k_u \rangle$ (conhecendo o valor correto de k) com $e = 2^{16} + 1$.

u	Calculando	tempo
2	$\langle k, k_1, k_2 \rangle$	$<$ 1 segundo
3	$\langle k, k_1, k_2, k_3 \rangle$	$<$ 1 segundo
4	$\langle k, k_1, k_2, k_3, k_4 \rangle$	5 segundos
5	$\langle k, k_1, k_2, k_3, k_4, k_5 \rangle$	4 dias
6	$\langle k, k_1, k_2, k_3, k_4, k_5, k_6 \rangle$	600 anos

Usar o criptossistema RSA multi-primo

- tem um maior número de escolhas possíveis para $\langle k_1, ..., k_u \rangle$.
- precisa uma maior quantidade de bits corretos aleatórios (δ) para fatorar N.

em relação ao RSA básico.

Obrigado!!!