Fatoração do inteiro N multi-primo com bits aleatórios

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

Orientador: Routo Terada

IME - DCC - USP

Abril de 2013

Agenda

1 - Conceitos Básicos

Criptossistema RSA

O criptossistema RSA está conformado por 3 algoritmos

1.-Algoritmo de Geração das chaves

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

- Chave pública $pk\langle N, e\rangle$
- Chave privada $sk\langle N, d\rangle$

2.- Algoritmo de encriptação

$M \in \mathbb{Z}_N$, $pk\langle N, e\rangle$

$$C = M^e \mod N$$

2.- Algoritmo de decifração

C, $sk\langle N, d\rangle$

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

Dados extras

- caso $u = 2$ é conhecido como **Criptossistema RSA Básico**
- caso $u \geq 3$ é conhecido como **Criptossistema RSA Multi-primo**

PKCS - Public Key Cryptography Standards

- O PKCS é um grupo de padrões devenvolvido pelos *laboratorios RSA*[1]
- O PKCS 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.

Representação da Chave Privada RSA segundo ao Padrão PKCS #1

- $pk\langle N, e\rangle \qquad \rightarrow C = M^e \mod N.$

Representações da Chave Privada RSA segundo ao Padrão PKCS #1

- $pk\langle N, d\rangle \qquad \rightarrow M = C^d \mod N.$
- $sk\langle r_1, r_2, d_1, d_2, r_2^{-1}, \langle r_3, d_3, t_3\rangle, .., \langle r_u, d_u, t_u\rangle\rangle \qquad \rightarrow \text{TCR}^a.$

[a] Teorema Chinês do Resto

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

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

Representação ANS.1 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.
- $sk\langle N, e, d, r_1, r_2, d_1, d_2, r_2^{-1}, \langle r_3, d_3, t_3\rangle, .., \langle r_u, d_u, t_u\rangle\rangle$.

2 - Objetivos

Objetivos

- 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.

sk  Ataque *Cold Boot* $\rightarrow$ $\tilde{sk}(\delta)$  Algoritmo de Heninger $\rightarrow$ sk 

Objetivos

- Estudar o problema de fatoração do inteiro $N = \prod_{i=1}^{u} r_i$ tendo um porcentagem δ de bits aleatórios conhecidos dos seus primos.
- Estudar o comportamento do algoritmo de Heninger e Shacham para Reconstruir sk RSA tendo um porcentagem δ de bits aleatórios de $\tilde{sk}$.

3 - Fatoração do Inteiro N

Introdução

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

Ideia do algoritmo

$$f(x_1, x_2, ..., x_u) = N - \prod_{i=1}^{u} x_i \qquad \overset{\text{solução}}{\Longrightarrow} \qquad f(r_1, r_2, ..., r_u) = 0$$

Vamos supor então que temos

$$f(r'_1, r'_2, ..., r'_u) \pmod{2^j} \qquad \Longrightarrow \qquad f(x_1, x_2, ..., x_u) \pmod{2^{j+1}}$$

Funcionamento do algoritmo

$$f \pmod 2 \Rightarrow f \pmod{2^2} \Rightarrow ... \Rightarrow f \pmod{2^j} \Rightarrow f \pmod{2^{j+1}} \Rightarrow ... \Rightarrow f \pmod{2^{\frac{n}{u}}}$$

- Lembrar que os primos r_i tem a mesma quantidade de bits $(lg(r_i) = \frac{n}{u})$

$$f(r_1, r_2, ..., r_u) \in f \pmod{2^{\frac{n}{u}}}$$

Lema de Hensel

Lema de Hensel para multivariáveis

Uma raiz $r = (r_1, r_2, ..., r_u)$ do polinômio $f(x_1, x_2, ..., x_u) \mod \pi^j$ pode ser usada para gerar uma raiz $r + b \mod \pi^{j+1}$ se $b = (b_1\pi^j, b_2\pi^j, ..., b_u\pi^j)$, $0 \leq b_i \leq \pi - 1$ é uma solução para a equação

$$f(r + b) = f(r) + \sum_i b_i \pi^j f_{x_i}(r) \equiv 0 \pmod{\pi^{j+1}}$$

(onde, f_{x_j} é a derivada parcial de f com relação a x_j)

Temos $r(r'_1, r'_2, ..., r'_u)$ que é raiz do polinômio $f(x_1, x_2, ..., x_u) \pmod{2^j}$, com o qual obtemos a raiz $r(r'_1 + 2^j b_1, r'_2 + 2^j b_2, ..., r'_u + 2^j b_u)$ que é raiz de $f(x_1, x_2, ..., x_u) \pmod{2^{j+1}}$

$$\left(N - \prod_{i=1}^{u} r'_i\right)[j] = \sum_{i=1}^{u} b_i \pmod 2$$

Observar que para uma são raiz de $f \pmod{2^j}$ vamos gerar um total de 2^{u-1} raízes para $f \pmod{2^{j+1}}$

Algoritmo de Fatoração de Inteiro

Podemos definir

$$raiz[j-1] = \langle r'_1, r'_2, ..., r'_u \rangle \in f \pmod 2^j$$

onde $raiz[0] = \langle 1, 1, ..., 1 \rangle$

$$raiz[0] \Rightarrow ... \Rightarrow raiz[j-1] \Rightarrow raiz[j] \Rightarrow ... \Rightarrow raiz\left[\frac{n}{u}\right]$$

Tendo uma solução do $raiz[j-1] = \langle r'_1, r'_2, ..., r'_u \rangle$, as soluções para o $raiz[j]$ estão dadas por

$$raiz[j] = \langle r'_1 + 2^j r_1[j], r'_2 + 2^j r_2[j], ..., r'_u + 2^j r_u[j] \rangle$$

onde deve cumprir que

$$\left(N - \prod_{i=1}^{u} r'_i\right)[j] = \sum_{i=1}^{u} r_i[j] \pmod 2$$

Algoritmo de Fatoração de Inteiro

Algorithm 1: Factoring N

Input: $N, u, \langle \tilde{r_1}, \tilde{r_2}, ..., \tilde{r_u} \rangle$

Output: $root[\frac{n}{u}]$ where $\langle r_1, r_2, ..., r_u \rangle$ is in $root[\frac{n}{u}]$

1 $root[0] = [\langle 1_1, 1_2, ..., 1_u \rangle]$;
2 $j = 1$;
3 **for each** $\langle r'_1, r'_2, ..., r'_u \rangle$ **in** $root[j-1]$ **do**
4 **for all possible** $\langle r_1[j], r_2[j], ..., r_u[j] \rangle$ **do**
5 **if** $(N - \prod_{i=1}^{u} r'_i)[j] \equiv \sum_{i=1}^{u} r_i[j] \pmod 2$ **then**
6 $root[j].add(\langle r'_1 + 2^j r_1[j], r'_2 + 2^j r_2[j], ..., r'_u + 2^j r_u[j] \rangle)$
7 **if** $j < \frac{n}{u}$ **then**
8 $j := j + 1$;
9 go to step 3;
10 **return** $root[\frac{n}{u}]$;

- se $r_i[j]$ for conhecido então vai ter só um valor (fixo).
- se $r_i[j]$ for desconhecido então vai ter dois valores (0 ou 1).

Complexidade do Algoritmo de Fatoração de Inteiro

Comportamento do Algoritmo de Fatoração de Inteiro

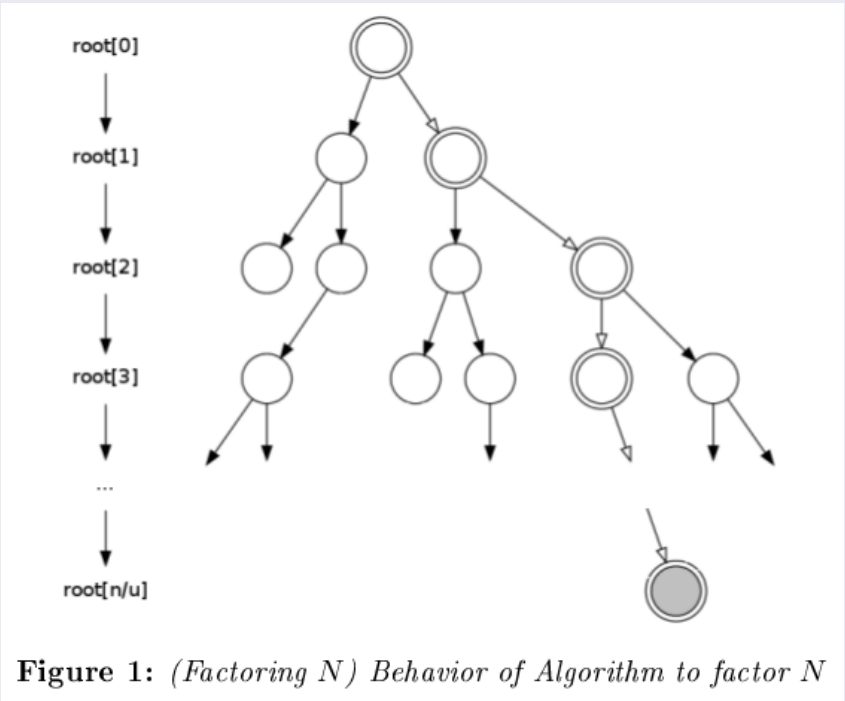


Figure 1: *(Factoring N) Behavior of Algorithm to factor N*

Análise da Complexidade do Algoritmo de Fatoração do Inteiro

- G: Número de raízes incorretas geradas por uma raízes boa.
- B: Número de raízes incorretas geradas por uma raízes incorreta.
- X_j: Número de raízes incorretas geradas no nível j.

Número de raízes geradas por uma raiz boa

Temos

- A raiz boa de $raiz[j-1]$
- Alguns bits conhecidos de $\langle r_1[j], r_2[j], ..., r_u[j] \rangle$ (devido que temos um porcentagem δ de bits conhecidos $\langle \tilde{r_1}, \tilde{r_2}, ..., \tilde{r_u} \rangle$)

$$\left(N - \prod_{i=1}^{u} r'_i \right)[j] = \sum_{i=1}^{u} r_i[j] \pmod 2$$

Número de raízes geradas por uma raiz boa

Seja h o número de bits que desconhecemos em $\langle r_1[j], r_2[j], ..., r_u[j] \rangle$

Casos	Nro. de raízes geradas
$1 \leq h \leq u$	2^{h-1}
$h = 0$	1

Lembrar que uma raiz boa do $raiz[j-1]$ sempre produz a raiz boa de $raiz[j]$ (o qual é única em todo nível).

Número de raízes incorretas geradas por uma raiz boa (B)

Número de raízes incorretas geradas por uma raiz boa (B)

Casos	Nro. de soluções incorretas geradas
$1 \leq h \leq u$	$2^{h-1} - 1$
$h = 0$	0

Valor Esperado de G $(\mathbb{E}[G])$

$$\mathbb{E}[G] = \sum_{h=1}^{u} (2^{h-1} - 1) P(b_u = h)$$

$$= \sum_{h=1}^{u} (2^{h-1} - 1) \binom{u}{h} (1-\delta)^h (\delta)^{u-h}$$

com $P(b_u = h) = P(bits_{desconhecidos} = h)$

Número de raízes incorretas geradas por uma raiz incorreta

Vamos definir

$$c_1 = \left(N - \prod_{i=1}^{u} r'_i \right) [j]$$

que é calculado pela raiz boa $raiz[j-1]$.

Tipos de raízes incorretas no $raiz[j-1]$

Vamos ter 2 tipos de raízes incorretas

$$c_1 \equiv \left(N - \prod_{i=1}^{u} r'_i \right) [j] = \sum_{i=1}^{u} r_i[j] \pmod 2$$
$$\overline{c_1} \equiv \left(N - \prod_{i=1}^{u} r'_i \right) [j] = \sum_{i=1}^{u} r_i[j] \pmod 2$$

Número de raízes incorretas geradas por uma raiz incorreta

Número de raízes incorretas geradas por uma raiz incorreta

Número de bits conhecidos	$c_1 \equiv \left(N - \prod_{i=1}^{u} r'_i\right)[j]$	$\overline{c_1} \equiv \left(N - \prod_{i=1}^{u} r'_i\right)[j]$
$1 \leq h \leq u$	2^{h-1}	2^{h-1}
$h = 0$	1	0

Valor Esperado de B ($\mathbb{E}[B]$)

$$\begin{aligned}\mathbb{E}[B] &= \sum_{h=1}^{u} 2^{h-1} P(b_u = h) P(c_1) + \sum_{h=1}^{u} 2^{h-1} P(b_u = h) P(\overline{c_1}) + P(b_u = 0) P(c_1) \\ &= \frac{(2-\delta)^u}{2}\end{aligned}$$

onde $P(c_1) \approx P(\overline{c_1}) \approx P\left(\left(N - \prod_{i=1}^{u} r'_i\right)[j] = 1\right) \approx P\left(\left(N - \prod_{i=1}^{u} r'_i\right)[j] = 0\right) \approx \frac{1}{2}$.

Número de Soluções Incorretas Geradas no nível j

Função de Recurrência: $X_j = X_{j-1}B + G$

Valor Esperado de X_j

$$\mathbb{E}[X_j] = \mathbb{E}[G]\frac{1-\mathbb{E}[B]^j}{1-\mathbb{E}[B]}$$

$$\begin{aligned}\mathbb{V}ar[X_j] =& \mathbb{E}[B]^{2(j-1)}\left[-\frac{\mathbb{E}[G][\mathbb{E}[B^2]-\mathbb{E}[B]+\mathbb{E}[B]\mathbb{E}[G]]\mathbb{E}[B]}{(1-\mathbb{E}[B])(1-\mathbb{E}[B]^2)}\right] + \mathbb{E}[G]\frac{1-\mathbb{E}[B]^j}{1-\mathbb{E}[B]} \\ & - \mathbb{E}[B]^{j-1}\left[\frac{\mathbb{E}[G][\mathbb{E}[B^2]-\mathbb{E}[B]+2\mathbb{E}[B]\mathbb{E}[G]]}{(1-\mathbb{E}[B])^2}\right] - \left[\mathbb{E}[G]\frac{1-\mathbb{E}[B]^j}{1-\mathbb{E}[B]}\right]^2 \\ & \frac{1}{1-\mathbb{E}[B]^2}\left[\frac{\mathbb{E}[G][\mathbb{E}[B^2]-\mathbb{E}[B]+\mathbb{E}[B]\mathbb{E}[G]]}{1-\mathbb{E}[B]}\right]\end{aligned}$$

A definição de $\mathbb{E}[X_j]$ e $\mathbb{V}ar[X_j]$ estão em função de j e δ.

Número de raízes Incorretas Geradas no nível j

Para definir $\mathbb{E}[X_j]$ e $\mathbb{V}ar[X_j]$ em função só de δ devemos declarar que $\mathbb{E}[B] < 1$ já que $\mathbb{E}[B]^{j\to\infty} = 0$

Valor Esperado de X_j ($\mathbb{E}[X_j]$) quando $j \to \infty$

$$\mathbb{E}[X_j] \leq \mathbb{E}[X_{j\to\infty}] = \frac{\mathbb{E}[G]}{1-\mathbb{E}[B]}$$

$$\mathbb{V}ar[X_j] \leq \mathbb{V}ar[X_{j\to\infty}] = \frac{1}{1-\mathbb{E}[B]^2}\Big[\frac{\mathbb{E}[G](\mathbb{E}[B^2]-\mathbb{E}[B]+2\mathbb{E}[G]\mathbb{E}[B])}{1-\mathbb{E}[B]} + \mathbb{E}[G^2]-\mathbb{E}[G]+\mathbb{E}[G](1+\mathbb{E}[B]) - \frac{\mathbb{E}[G]^2(1+\mathbb{E}[B])}{1-\mathbb{E}[B]}\Big]$$

A definição de $\mathbb{E}[X_j]$ e $\mathbb{V}ar[X_j]$ estão em função de δ.

Número de raízes incorretas analisadas na execução de algoritmo

$$\mathbb{E}\left[\sum_{j=1}^{\frac{n}{u}} X_j\right] = \mathbb{E}[X_1 + ... + X_{\frac{n}{u}}] = \mathbb{E}[X_1] + ... + \mathbb{E}[X_{\frac{n}{u}}] \leq \mathbb{E}[X_{j\to\infty}] + ... + \mathbb{E}[X_{j\to\infty}]$$

$$\leq \frac{n}{u}\mathbb{E}[X_{j\to\infty}]$$

$$\mathbb{V}ar\left[\sum_{j=1}^{\frac{n}{u}} X_j\right] = \mathbb{E}\left[\left(\sum_{j=1}^{\frac{n}{u}} X_j\right)^2\right] - \mathbb{E}\left[\sum_{j=1}^{\frac{n}{u}} X_j\right]^2 = \sum_{l=1}^{\frac{n}{u}}\sum_{j=1}^{\frac{n}{u}} Cov(X_l, X_j)$$

$$\leq \sum_{l=1}^{\frac{n}{u}}\sum_{j=1}^{\frac{n}{u}} \sqrt{\mathbb{V}ar[X_l]\mathbb{V}ar[X_j]}$$

$$\leq \sum_{l=1}^{\frac{n}{u}}\sum_{j=1}^{\frac{n}{u}} \mathbb{V}ar[X_{j\to\infty}]$$

$$\leq \left(\frac{n}{u}\right)^2 \mathbb{V}ar[X_{j\to\infty}]$$

Algoritmo de Fatoração de Inteiro

Teorema de Chebyshev

A desigualdade de Chebyshev proporciona um intervalo para a probabilidade de que uma v.a. fique longe de certo número de desviações estândar do valor esperado.

$$P(\mathbb{E}[X] - c\sigma < X < \mathbb{E}[X] + c\sigma) \geq 1 - \frac{1}{c^2}$$

A probabilidade de que qualquer v.a. tenha um valor dentro das c desviações estândar do valor esperado é pelo menos de $1 - \frac{1}{c^2}$.

Onde aplicado ao problema de Fatoração de Inteiros, temos que a probabilidade de analisar mais de

$$\mathbb{E}[\sum_{j=1}^{\frac{n}{u}} X_j] + n\sqrt{\mathbb{V}ar[\sum_{j=1}^{\frac{n}{u}} X_j]} \leq \frac{n}{u}\mathbb{E}[X_{j\to\infty}] + \frac{n^2}{u}\sqrt{\mathbb{V}ar[X_{j\to\infty}]}$$

raízes incorretas é menor a $\frac{1}{n^2}$.

Algoritmo de Fatoração de Inteiro

Complexidade do Algoritmo

É possível fatorar $N = \prod_{i=1}^{u} r_i$ em tempo polinomial $O(n^2)$ com uma probabilidade maior a $1 - \frac{1}{n^2}$ quando temos um porcentagem δ de bits conhecidos de $\langle \tilde{r_1}, \tilde{r_2}, ..., \tilde{r_u} \rangle$ maior a $2 - 2^{\frac{1}{u}}$ ($\delta > 2 - 2^{\frac{1}{u}}$).

$$\mathbb{E}[B] = \frac{(2-\delta)^u}{2} < 1 \qquad \Rightarrow \qquad \delta > 2 - 2^{\frac{1}{u}}.$$

Alguns Resultados

- Para fatorar $N = \prod_{i=1}^{2} r_i$ é preciso um $\delta > 2 - 2^{\frac{1}{2}} = 0.5857$ ($\delta \geq 0.59$)
- Para fatorar $N = \prod_{i=1}^{3} r_i$ é preciso um $\delta > 2 - 2^{\frac{1}{3}} = 0.7401$ ($\delta \geq 0.75$)
- Para fatorar $N = \prod_{i=1}^{4} r_i$ é preciso um $\delta > 2 - 2^{\frac{1}{4}} = 0.8108$ ($\delta \geq 0.82$)

Implementação do Algoritmo de Fatoração

- O Algoritmo de Fatoração 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 inteiros N de 2048 bits e para δ específicos.
- Para cada δ for gerado um total de 100 inteiros N.
- para cada inteiro N foi gerado 100 entradas com um porcentagem δ de bits corretos.
- Os experimentos foram feitos para inteiros $N = \prod_{i=1}^{4} r_i$ com $2 \leq u \leq 4$.

Resultados

- Para $N = \prod_{i=1}^{2} r_i$ é $\delta = 0.59$ vamos analisar menos de $15n + 15n^2$ soluções incorretas.

	Quantidade de soluções analisadas			# exp.	Tempo
δ	Mínimo	Máximo	Média	(> 1M)	Média
0.62	1861	347138	3709	0	0.047510
0.61	1983	945728	4949	0	0.115277
0.60	2233	789608	6344	0	0.119484
0.59	2411	928829	8953	2	0.187600
0.58	2631	987577	14736	7	0.250224
0.57	3436	994640	24281	29	0.531079
0.56	4012	998414	42231	134	0.722388

Resultados

- Para $N = \prod_{i=1}^{3} r_i$ é $\delta = 0.75$ vamos analisar menos de $3n + 4n^2$ soluções incorretas.

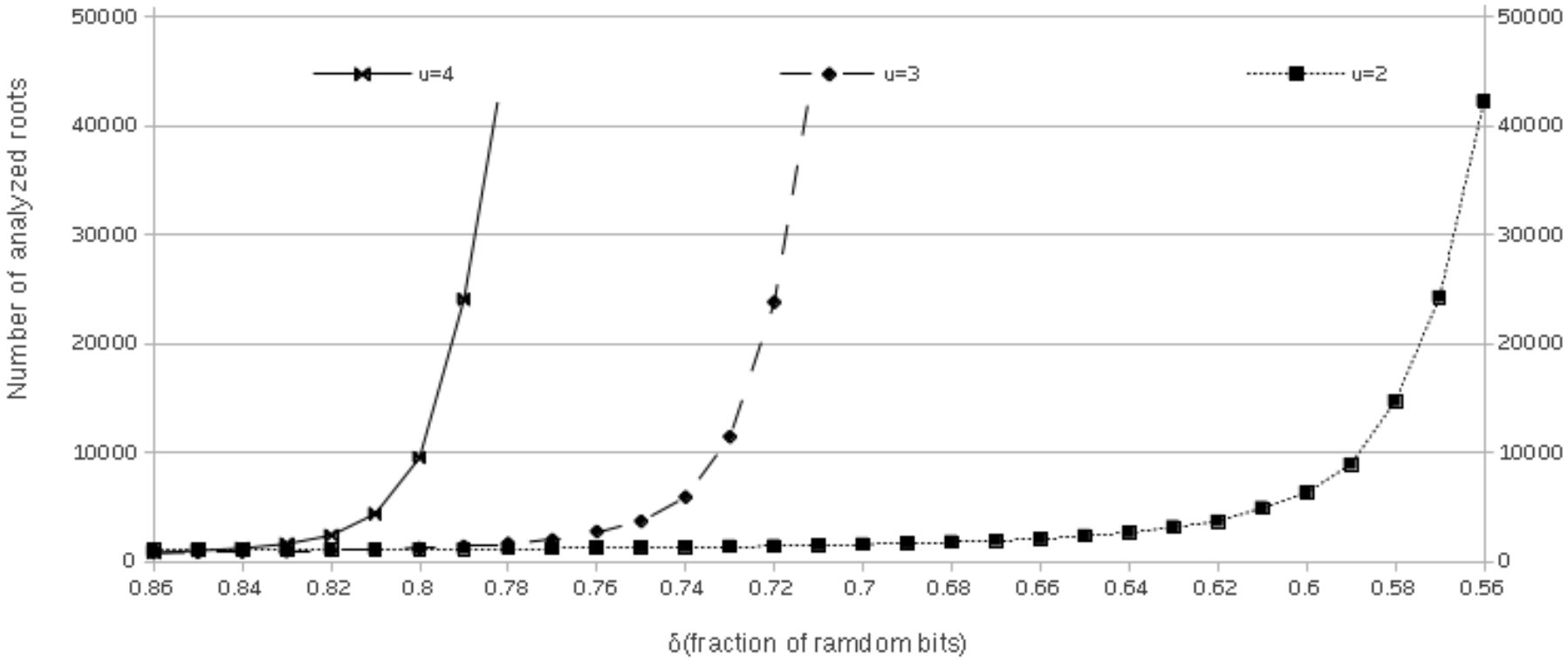
	Quantidade de soluções analisadas			# exp.	Tempo
δ	Mínimo	Máximo	Média	(> 1M)	Média
0.78	985	35509	1676	0	0.032866
0.77	1128	171142	2022	0	0.033884
0.76	1205	323228	2777	0	0.049238
0.75	1380	177293	3723	1	0.099373
0.74	1607	571189	5941	1	0.197553
0.73	1681	999766	11470	11	0.281414
0.72	2087	983404	23826	50	0.995017

Resultados

- Para $N = \prod_{i=1}^{4} r_i$ é $\delta = 0.82$ vamos analisar menos de $2n + 2n^2$ soluções incorretas.

	Quantidade de soluções analisadas			# exp.	Tempo
δ	Mínimo	Máximo	Média	(> 1M)	Média
0.85	692	32620	1026	0	0.019939
0.84	716	31447	1245	0	0.024748
0.83	823	67456	1649	0	0.040714
0.82	931	217391	2424	0	0.063754
0.81	1044	558521	4408	1	0.111688
0.80	1249	994386	9571	14	0.236320
0.79	1632	972196	24085	58	0.609435

Resultados de Fatoração do Inteiro N



Obrigado!!!