Reconstrução da Chave Privada RSA

Reynaldo C. Villena
Orientador: Routo Terada

Instituto de Matemática e Estatística
Universidade de São Paulo

Março de 2012

Agenda

1 - Introdução

RSA

- O RSA é o criptossistema de chave pública mais usado e implementado até a date.
- Seu nome é derivado das iniciais dos seus criadores: Ron (R)ivest, Adi (S)hamir e Len (A)dleman.
- Desde sua publicação, nenhum ataque conseguiu quebrá-lo, portanto não foi preciso mudar sua estrutura.
- Existem pesquisas sob casos onde o RSA é inseguro, mas isso é devido ao uso inadequado do mesmo.
- Foi criado em agosto de 1977 no MIT e publicado em fevereiro de 1978.

Algoritmos RSA (Geração das chaves)

1.- Geração das chaves

Usando um algoritmo probabilístico determinamos dois primos p e q (de preferencia com o mesmo tamanho de bits) e calculamos $N = pq$. A seguir, escolhemos um e co-primo a $\phi(N) = (p-1)(q-1)$ e calculamos o valor de d tal que que $ed = 1 \mod \phi(N)$. Publicamos $pk\langle N, e\rangle$ e Mantemos em segredo $sk\langle N, d\rangle$.

RSA multi-primo

O N é a multiplicação de três a mais primos distintos.

$$N = \prod_{i=1}^{u} r_i \ , \ \phi(N) = \prod_{i=1}^{u} (r_i - 1)$$

Algoritmos RSA (Encriptação - Decifração)

2.- Encriptação

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

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

A seguir, o criptograma C é enviado ao possessor da chave privada.

3.- Decifração

O possessor da chave privada recebe o criptograma C. Ele para conseguir ler o mensagem M deve calcular

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

Algoritmos RSA (Decifração usando TCR)

Foi proposto por J-J. Quisquater and C. Couvreur em 1982.
Esse método consiste numa modificação do da chave privada do RSA.
A chave privada está dada por $sk\langle p, q, d_p, d_q, q^{-1}\rangle$ onde:

$$d_p = d \mod (p-1),\ d_q = d \mod (q-1) \text{ e } q^{-1} = q^{-1} \mod p$$

Decifração usando TCR

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

Esse método ficou popular já que seu tempo de execução é 4 vezes mais rápido do que a aplicação de $M \leftarrow C^d \mod N$.

Segurança do RSA

3.- Requisito de um criptossistema de chave pública

A recuperação da chave privada sk a partir da chave pública pk é um problema computacionalmente inviável.

Temos os valores $\langle e, N \rangle$, como achar o d?
Para calcular o d podemos aplicar o algoritmo de Euclides Estendido a e e $\phi(N) = (p-1)(q-1)$, mas para achar $\phi(N)$ temos que ter o valores de p e q ou simplesmente fatorar N.

Segurança do RSA

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

Fatoração do N

O algoritmo mais rápido para a solução dele é conhecido como NFS *(Number Field Sieve)*. Este algoritmo estabeleceu um recorde em 1999 conseguindo fatorar um número de 465 bits(140 decimais) em vários meses utilizando centenas de estações de trabalho[5].
Observando a dificuldade de fatorar um inteiro em tempo polinomial surgiu a ideia:

Ideia

Será possível fatorar N em tempo polinomial tendo acesso a algum tipo de informação extra (neste caso bits dos fatores primos de N ou outros dados)?

Assistência por um oráculo

- Um oráculo é um programa que responde perguntas com uma resposta booleana

$$\text{Sim} = 1$$
$$\text{Não} = 0$$

sob informação relacionada a nossa chave privada[2].

Caso trivial

Para fatorar N só precisamos fazer $n/2$ perguntas ao oráculo. onde n é o número de bits de $N(n = \lg N)$.

Fatoração de N assistida por um oráculo

É possível fatorar N em tempo polinomial sabendo os:

- $n/4$ bits menos significativos de p[?].
- $n/4$ bits menos significativos de d[?].
- $n/4$ bits menos significativos de d_p[?].

O algoritmo de Heninger e Shacham a explicar pertence a essa linha de pesquisa, mas a diferencia com os ataques antes mencionados é que o atacante não tem controle das posições dos bits, ou seja, o atacante não escolhe os bits que quer saber. Com este algoritmo podemos fatorar N em tempo polinomial sabendo o:

- 27 % dos bits de p, q, d, d_p e d_q.
- 42 % dos bits de p, q e d.
- 57 % dos bits de p e q.

2 - Objetivos e Resultados Esperados

Objetivos e Resultados Esperados

Objetivos

- Implementação e análise do algoritmo Heninger-Shacham para o criptossistema RSA e RSA multi-primo.

3 - Conceitos Básicos

PKCS - Public Key Cryptography Standards

O PKCS é grupo de padrões que contem especificações para acelerar a implementação e desenvolvimento dos algoritmos dos criptossistemas de chave pública.
O PKCS #1 é o primeiro padrão e contém definições básicas e recomendações para a implementação do criptossistema RSA.
O PKCS # 1 especifica que podemos ter duas representações da chave privada:

- Primeira representação.

$$sk\langle N, d\rangle \overset{decriptar}{\rightarrow} M = C^d \mod N$$

- Segunda representação.

$$sk\langle p, q, d_p, d_q, q^{-1}\rangle \overset{decriptar}{\rightarrow} (TCR)$$

Chaves 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
}
```

Podemos observar que a chave privada é altamente redundante.

Ataques *side channel*

Informação *side channel*

A informação *side-channel* é toda informação que pode ser obtida do dispositivo de encriptação enquanto os processos de encriptação e decifração estão em operação. A informação *side-channel* não é nem a mensagem M e nem a criptograma C[1].

Um ataque *side-channel* é aquele que utiliza essas entradas e saídas adicionais, como por exemplo:

- tempo de demora da execução dos processos
- consumo de eletricidade
- radiação de vários tipos
- sons produzidos (criptoanálise acústica)
- variação do voltagem na fonte de alimentação

Ataques *side channel* - Ataques *cold boot*

Vamos definir antes,

cool-boot ou _hard-boot_

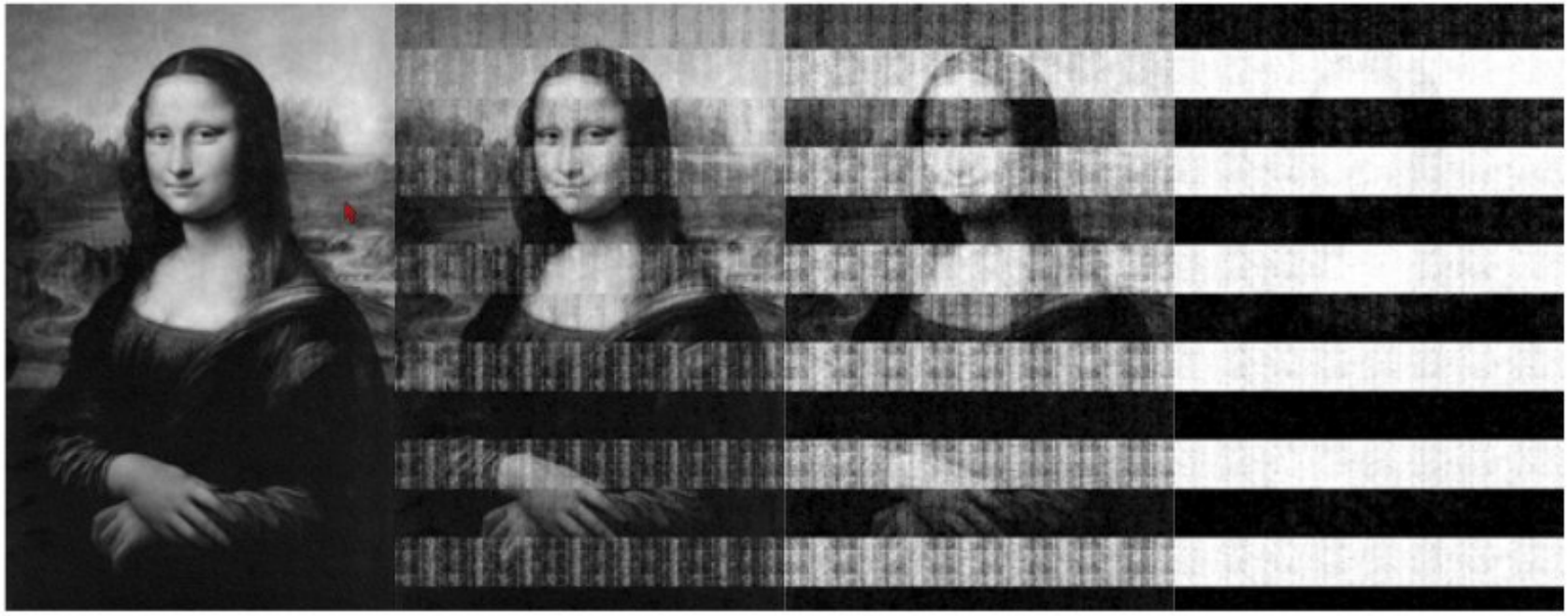
Se refere ao reiniciamento de um computador depois de ter sido cortada sua fonte de energia abruptamente.

Remanência da memória DRAM

Capacidade de conservação de dados da memória DRAM/SRAM depois de se aplicar um cold-boot.

Um ataque *cold-boot* é um tipo de ataque *side-channel* onde o atacante utiliza dados que foram obtidos da memória DRAM/SRAM depois de aplicar um *cold-boot* ao computador [3].

Decaimento gradual dos bits na memoria DRAM/SRAM



No pior dos casos, o decaimentos das carregas dos capacitadores começa aos 2.5 segundos [4].

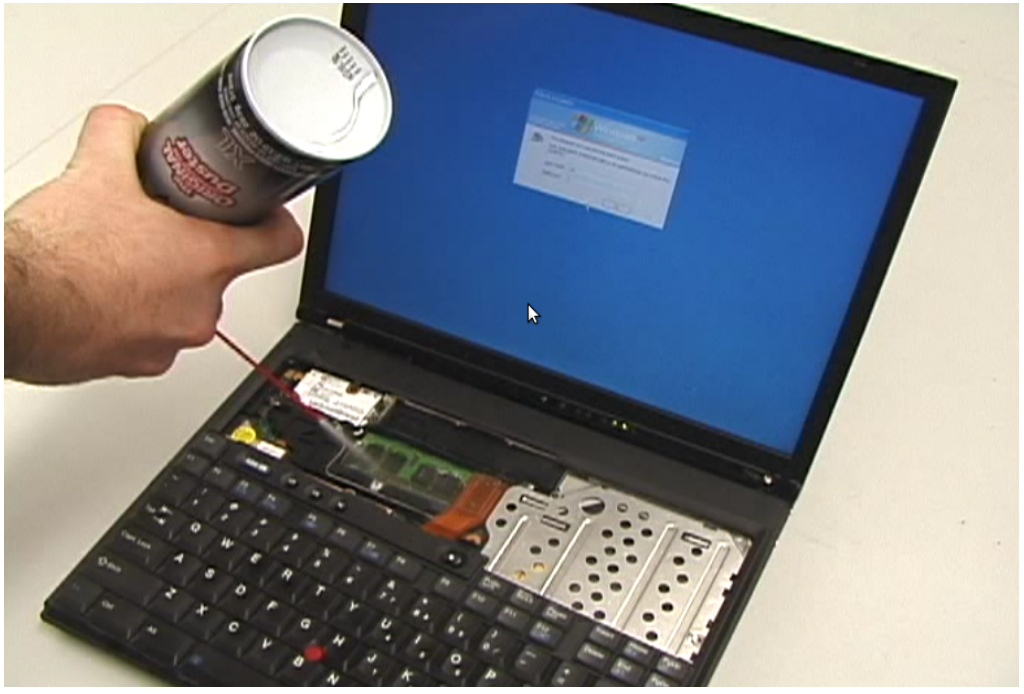
Ataque *cold boot*

Passos para efetuar um ataque *cold boot*

1. Efetuar um *cold boot* ao dispositivo enquanto os processos de encriptação ou decifração são executados.
2. Devemos inicializar um sistema operacional pequeno (PXE, Syslinux bootloader) no dispositivo que possa usar instruções de acesso a memória para obter uma copia da memória DRAM.

Para obter uma imagem da memória DRAM sem erros só temos 2.5 segundos para realizar o passo 2 depois do passo 1, mas isso pode levar mais tempo. Para acrescentar esse tempo podemos usar técnicas de esfriamento.

Esfriando a memória DRAM



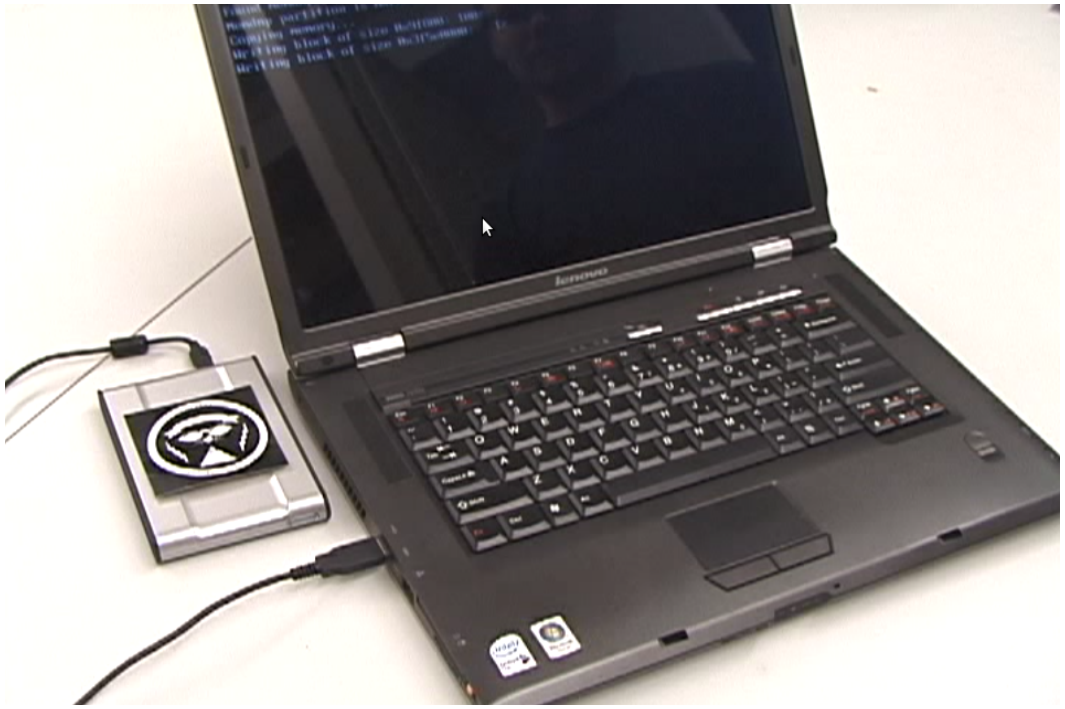
Esfriamento do chip da memória DRAM usando um *canned air*.

Decaimento lento dos bits esfriando a memória DRAM

Aplicando um *Canned air* (-50°C)
0.0001% a 0.2% decaimento dos bits
depois do 1-5 minuto(s)

Submergindo em nitrogênio liquido(-196°C)
$< 0.1\%$ decaimento dos bits
depois de 60 minutos

Gerando imagem da memória DRAM



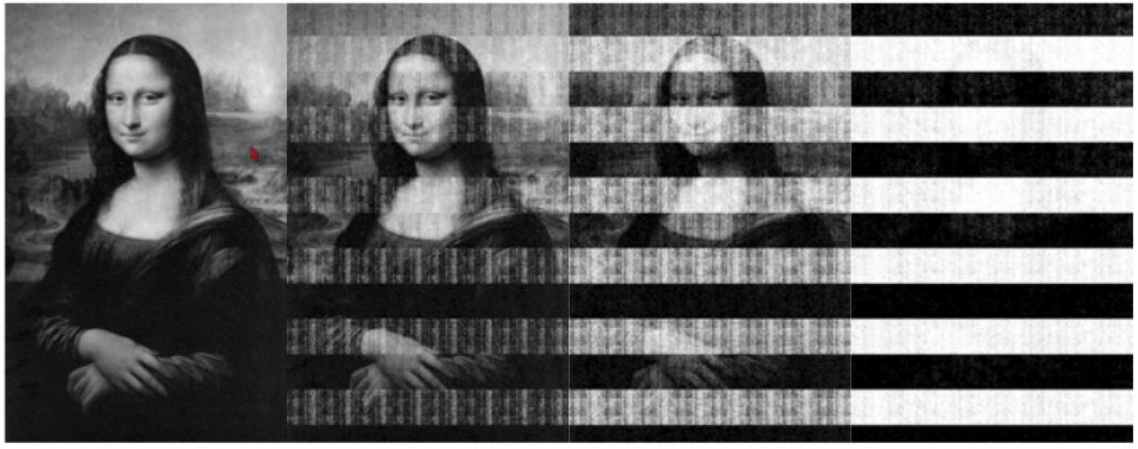
Iniciando um programa *Syslinux bootloader* para obter uma imagem da memória DRAM.

Fracção de bits corretos na imagem da DRAM

DRAM armazena cada bit de dados num condensador ou capacitor.

capacitor com carrega representa o bit 1 (estado 1).
capacitor sem carrega representa o bit 0 (estado 0).

Depois do *cold boot* o decaimento dos bits pode ocorrer em duas direções dependendo das regiões da memória.

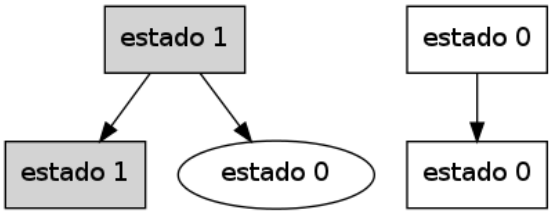


5s. 30s. 1m. 5m.

Fracção de bits corretos na imagem da DRAM

Decaimento dos bits na:

Região 0



Região 1

Fracção de bits corretos na imagem da DRAM

$$\delta = \frac{1-\rho}{2}$$

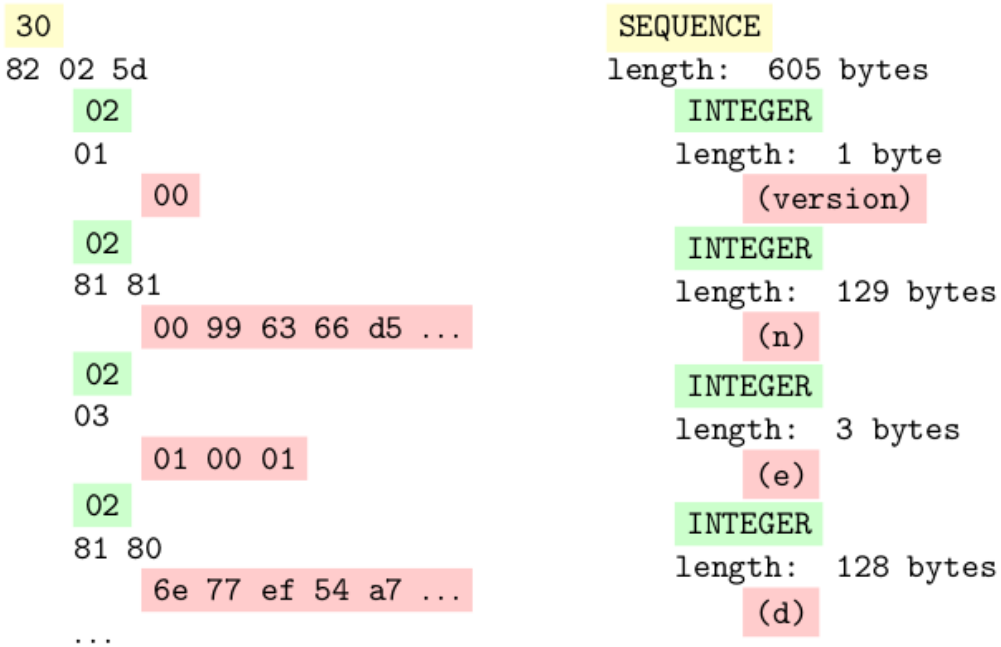
onde ρ é a fracção de bits decaídos, supondo que o decaimento de bits acontece na mesma quantidade nas regiões 1 e 0.

Identificação da chave privada RSA na memória

- Tentar multiplicar blocos de memória próximos? [**?**]
- Tentar decifrar com cada bloco de memória? [**?**]
- Procurar a sequencia de bytes 02-01-00-02? [**?**]

Codificação BER - PKCS #1

Chave privada RSA codificada na memória:



O que fazer em uma imagem com erros?

Procurar a estrutura com a menor distancia Hamming de $\tilde{N}$ e $\tilde{e}$ com respeito à chave privada $sk\langle N, e\rangle$.

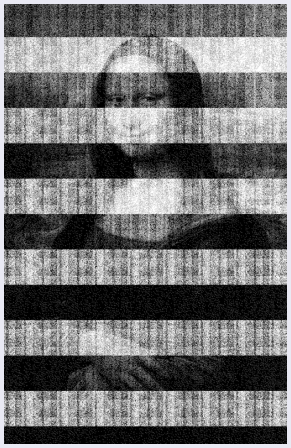
4 - Algoritmo de Heninger-Shacham

Algoritmo de reconstrução da chave privada

Da imagem residual da memória DRAM podemos obter:

$$\tilde{sk}\ \langle \tilde{d}, \tilde{p}, \tilde{q}, \tilde{d}_p, \tilde{d}_q, \tilde{q}^{-1} \rangle \text{ com um } \delta.$$

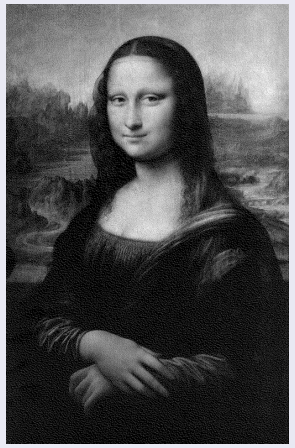
Chave privada com erros



$\tilde{sk}$

$\xrightarrow{HS}$

Chave privada



sk

Redundância da chave privada

Redundância de dados que oferece a chave privada

$$sk = (N, e, p, q, d, d_p, d_q, q^{-1})$$

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

$$\begin{aligned}
pq &= N && \Rightarrow && pq &= N \\
ed &= 1 \mod (p-1)(q-1) && \Rightarrow && ed &= 1 + k(p-1)(q-1) \\
ed_p &= 1 \mod (p-1) && \Rightarrow && ed_p &= 1 + k_p(p-1) \\
ed_q &= 1 \mod (q-1) && \Rightarrow && ed_q &= 1 + k_q(q-1)
\end{aligned}$$

- Como calcular o valor de k, k_p, k_q?

Calculando o valor de K

Cada candidato é obtido pela atribuição de um valor para k.

Equação RSA

$$ed = 1 + k(p-1)(q-1)$$

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

O número de candidatos para k está dado pelo valor "$e - 1$"

$$(0 < k < e)$$

Dado

Para a maioria de criptossistemas RSA definem o valor de $e = 2^{16}+1$, portanto temos 2^{16} candidatos para k

- E possível encontrar valores potenciais para o k?

Procura de valores potenciais para o k

Teorema

Para um e pequeno, a metade dos bits mais significativos de d pode ser estimados eficientemente [3].

denotamos como k' ao k sendo atribuído por um valor.

$$ed = 1 + k(p-1)(q-1) = 1 + k(N+1) - k(p+q)$$

$$d' = \frac{k'(N+1)+1}{e}$$

Se o valor de $k' = k$ então temos a metade superior dos bit de d.

Candidatos potenciais para o k são

Aqueles k' que tem a menor distancia Hamming entre os a metade superior dos bits entre d' e $\tilde{d}$.

Determinação dos valores de k_p e k_q

Temos as equações:

$$ed = 1 + k(p-1)(q-1)$$
$$ed_p = 1 + k_p(q-1)$$
$$ed_q = 1 + k_q(p-1)$$

$\Rightarrow$

Aplicando (mod e)

$$0 = 1 + k(p-1)(q-1)$$
$$0 = 1 + k_p(q-1)$$
$$0 = 1 + k_q(p-1)$$

A solução ao sistema de equações (mod e) está dada por:

$$k_p^2 - [k(N-1)+1]k_p - k = 0$$
$$k + k_p k_q = 0$$

O numero de soluções para $\langle k_p, k_q \rangle$ esta dado por 2^m, onde m é o numero de fatores primos de e. Sendo $e = 2^{16} + 1$ só temos duas soluções.

Correção dos bits superiores

Temos os seguintes dados $\tilde{sk}\langle \tilde{p}, \tilde{q}, \tilde{d}, \tilde{d_p}, \tilde{d_q} \rangle$, k, N e e

Determinando a metade superior de bits de d

$$D = \frac{k(N+1)+1}{e}$$

Pelo teorema já visto sabemos que o D tem a metade superior dos bits de d, portanto podemos corrigir a metade superior de $\tilde{d}$

Correção dos bits superiores

para i desde $\lfloor \frac{n}{2} \rfloor + 1$ até n:

$$\tilde{d}[i] = D[i]$$

Correção dos bits inferiores de:

Sabemos que p e q são primos portanto corrigimos os bit:

$p[0] = 1,\ q[0] = 1$

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

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

$$ed_p = 1 \mod 2^{1+\tau(k_p)}$$

$$d_p = 1 \mod 2^{1+\tau(k_p)}$$

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

Algoritmo de Reconstrução

O algoritmo está baseado nas mudanças dos bits:

- Uma troca de valor no $p[i]$ afeta ao valor de $d_p[1+\tau(k_p)]$
- Uma troca de valor no $q[i]$ afeta ao valor de $d_q[1+\tau(k_q)]$
- Uma troca de valor no $q[i]$ ou $p[i]$ afeta ao valor de $d[1+\tau(k)]$

Portanto podemos definir o seguinte

$$\text{grupo}[i] = (p[i], q[i], d[i+\tau(k)], d_p[i+\tau(k_p)], d_q[i+\tau(k_q)])$$

onde o $\text{grupo}[0] = (1, 1, d[\tau(k)], d_p[\tau(k_p)], d_q[\tau(k_q)])$

Algoritmo HS - Gerando possíveis soluções

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

$$\text{grupo}[i-1] \rightarrow \text{grupo}[i]$$

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

$$(p[i], q[i], d[i+\tau(k)], d_p[i+\tau(k_p)], d_q[i+\tau(k_q)])$$

- Então, temos $2^5 = 32$ soluções para o grupo$[i]$?

Restrições

Restrições RSA temos:

$$
\begin{aligned}
N - p'q' \equiv c_1 2^i &\equiv 0 \mod 2^i \\
k(N+1) + 1 - k(p'+q') - ed' \equiv c_2 2^{i+\tau(k)} &\equiv 0 \mod 2^{i+\tau(k)} \\
k_p(p'-1) + 1 - ed'_p \equiv c_3 2^{i+\tau(k_p)} &\equiv 0 \mod 2^{i+\tau(k_p)} \\
k_q(q'-1) + 1 - ed'_q \equiv c_4 2^{i+\tau(k_q)} &\equiv 0 \mod 2^{i+\tau(k_q)}
\end{aligned}
$$

- Com estas restrições o número de soluções para o grupo[i] diminui a 2. (32 sol $\rightarrow$ 2 sol)
- Então, temos 2 soluções para o grupo[i]?

Algoritmo HS - Número de soluções

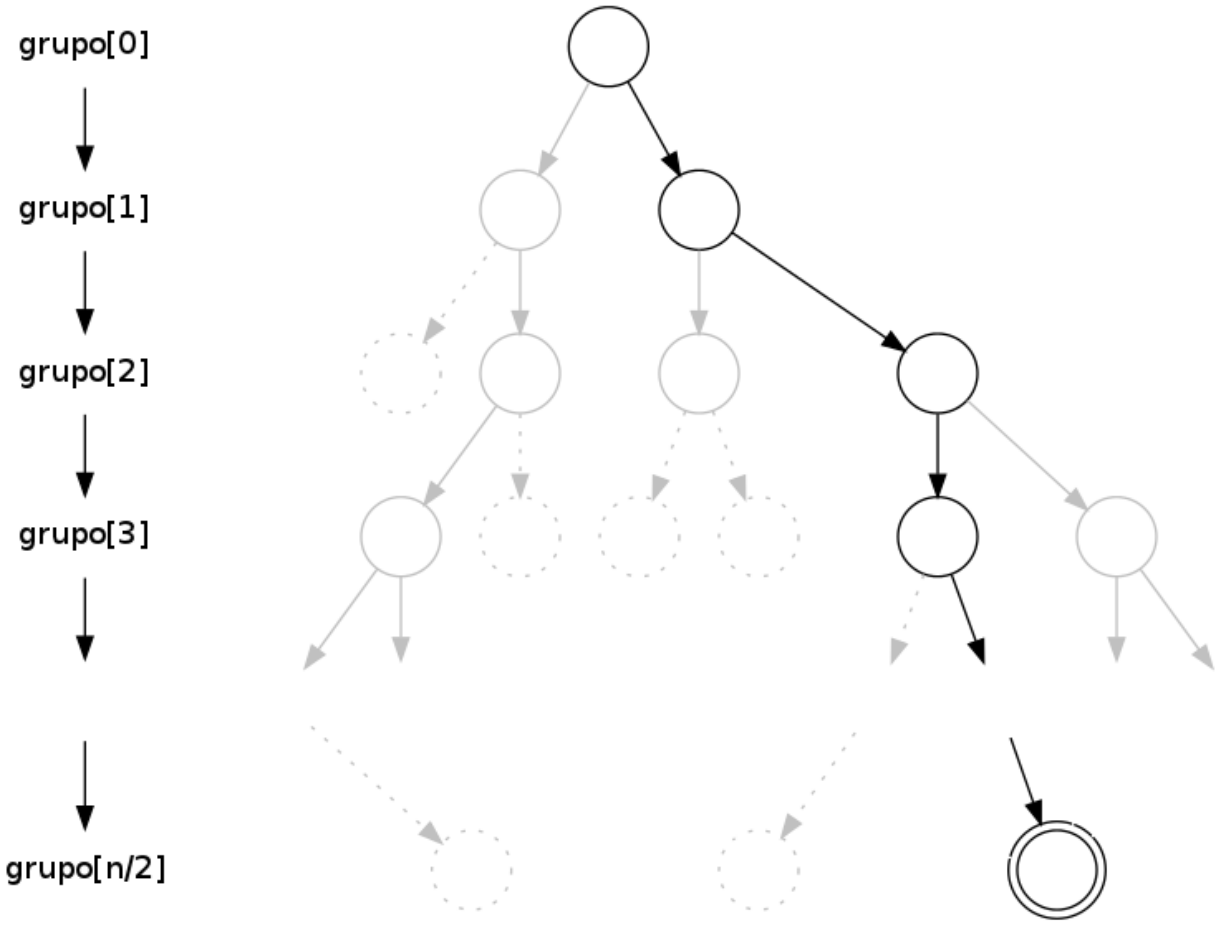
Lembrando de da chave privada $\tilde{sk}$ $\langle \tilde{d}, \tilde{p}, \tilde{q}, \tilde{d}_p, \tilde{d}_q \rangle$ e o δ, temos um porcentagem de bits conhecidos.
Vamos denotar como bit fixo $s[i]$ se o bit i da variável s é conhecido, portanto todas nossas soluções para o grupo$[i]$ devem satisfazer as restrições com esse bit fixo(bits conhecidos).

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

- Em total vamos a ter 0,1 ou 2 soluções dependendo de quantos bits fixos temos para o grupo$[i]$.

Complexidade do Algoritmo HS

Gerando a árvore do funcionamento do algoritmo HS



Complexidade do Algoritmo HS

Para determinar a complexidade, vamos denotar as seguintes variáveis:

Z_g: número de soluções incorretas geradas por uma solução boa.

W_b: número de soluções incorretas geradas por uma solução incorreta.

X_i: número de soluções incorretas geradas no nível i.

Valor esperado de Z_g:

$$\mathbb{E}Z_g = \delta(1-\delta)^4+(1-\delta)^5$$

Valor esperado de W_b:

$$\mathbb{E}W_b = (2-\delta)^5/16$$

Valor esperado de X_i:

$$\mathbb{E}X_i = \frac{\mathbb{E}Z_g(1-(\mathbb{E}W_b)^i)}{1-\mathbb{E}W_b}$$

Resultados do Algoritmo de Heninger e Shacham

Se o atacante tem conhecimento parcial de...	... então a recuperação é eficiente para...
d, p, q, d_p, d_q	$\delta \geq 0.27(9n^2 + 71n)$
d, p, q	$\delta \geq 0.42(22n^2 + 24n)$
p, q	$\delta \geq 0.59(29n^2 + 29n)$
p	Problema aberto
	Fracção de bits conhecidos

Implementação do algoritmo HS

- O algoritmo de reconstrução de HS foi implementado em Python 2.7.1+ 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 de 64, 128, 256 e 512 bits e para um $\delta \in [0.24, 0.4]$.
- Para cada chave de tamanho n foi gerado 10 criptossitemas e para cada criptossistema e cada δ foi gerado 10 chaves privadas com bits modificadas.
- Todos os criptossistemas tinham o expoente de encriptação $e = 2^{16} + 1$.
- Os experimentos foram testados com os valores corretos de k, k_p, k_q para evitar dados inecessários.
- Em total foram testados 6800 experimentos.

Tempo médio (segundos) onde o N foi fatorado com sucesso

δ	$n = 64$ bits	128 bits	256 bits	512 bits
0.40	0.01051379(s)	0.07464193	0.41150759	2.28971588
0.39	0.01233571	0.07639466	0.49165601	2.49166408
0.38	0.01583051	0.1015213	0.53972828	2.68270419
0.37	0.01442633	0.08561128	0.6288269	2.50079252
0.36	0.01764429	0.11193392	0.69634689	3.06340508
0.35	0.01965901	0.11074164	0.75505082	3.12433709
0.34	0.01657437	0.12185847	0.82748678	3.64784701
0.33	0.01613043	0.12720953	0.92805324	4.39663128
0.32	0.0179989	0.14180567	0.99390682	4.50099887
0.31	0.02370106	0.15494912	1.02700172	5.01726369
0.30	0.02022137	0.17714891	1.02716925	5.54644315
0.29	0.02669204	0.19780392	1.57295997	7.507961915
0.28	0.03046309	0.25644511	1.70409487	11.26124344
0.27	0.04655601	0.35181175	2.06926785	16.36289062

Referencias

- H. Bar-El, *Introduction to side channel attacks*, White Paper. Discretix Technologies Ltd, 2003.
- Nadia Heninger e Hovav Shacham, *Reconstructing rsa private keys from random key*, Cryptology ePrint Archive, Report 2009/510, 2009. http://eprint.iacr.org/.
- J.A. Halderman, S.D. Schoen, N. Heninger, W. Clarkson, W. Paul, J.A. Calandrino, A.J. Feldman, J. Appelbaum e E.W. Felten, *Lest we remember: cold-boot attacks on encryption keys*, Communications of the ACM, 52(5):91-98, 2009.
- S. Skorobogatov, *Low temperature data remanence in static ram*, University of Cambridge Computer Laborary Technical Report, 536, 2002.
- Routo Terada, *Segurança de dados: criptografia em redes de computador*, Edgard Blucher, 2000.

Referencias

- J. Jonsson e B. Kaliski, *Public-key cryptography standards (pkcs) # 1: Rsa cryptography specifications version 2.1*, Relatório técnico, RFC 3447, February, 2003.
- U.M. Maurer, *Factoring with an oracle*, Em Proceedings of the 11th annual interna- tional conference on Theory and application of cryptographic techniques, páginas 429-436. Springer-Verlag, 1992.
- S. Maitra, S. Sarkar e S. Sen Gupta A.J. Feldman, J. Appelbaum e E.W. Felten, *Publishing upper half of rsa decryption exponent*, Advances in Information and Computer Security, páginas 25-39, 2010.