Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Teste de robustez de chaves RSA

João Henrique Gonçalves de Sousa

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Me. Pedro Antônio Dourado de Rezende

Brasília
2016

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenador: Prof. Dr. Coordenador Rodrigo Bonifácio de Almeida

Banca examinadora composta por:

Prof. Me. Pedro Antônio Dourado de Rezende (Orientador) — CIC/UnB
Prof. Dr. Jan Mendonca Correa — CIC/UnB
Prof. Me. João José Costa Gondim — CIC/UnB

CIP — Catalogação Internacional na Publicação

Sousa, João Henrique Gonçalves de.

Teste de robustez de chaves RSA / João Henrique Gonçalves de Sousa.
Brasília : UnB, 2016.
129 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2016.

1. chaves públicas RSA, 2. módulo RSA, 3. máximo divisor comum,
4. teste de robustez, 5. ICP-Brasil

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Teste de robustez de chaves RSA

João Henrique Gonçalves de Sousa

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Me. Pedro Antônio Dourado de Rezende (Orientador)
CIC/UnB

Prof. Dr. Jan Mendonca Correa
CIC/UnB

Prof. Me. João José Costa Gondim
CIC/UnB

Prof. Dr. Coordenador Rodrigo Bonifácio de Almeida
Coordenador do Bacharelado em Ciência da Computação

Brasília, 26 de maio de 2016

Dedicatória

Dedico aos meus pais, Simara e João José, e ao meu irmão José Eduardo que sempre me apoiaram, especialmente em momentos dificeis ao longo do curso. Dedico à universidade UnB, que me deu excelente formação e por meio da UnB me apaixonei pelo curso e pelas ciências em geral. E a todos que me acompanharam no curso, amigos e professores amigos: Amaral, Guilherme Branco, Daniella dos Angelos, Pimenta, Lamar, Bruno Machiavello, Pedro Rezende, Genaina Nunes, Hemar Godinho, Igor Bonomo e tantos outros.

Agradecimentos

Agradeço a todos que me inspiraram, me ajudaram, me acompanharam e que estiveram comigo de alguma maneira ao longo deste curso de ciência da computação.

Resumo

A infraestrutura de chaves públicas é essencial na segurança da informática, e esta utiliza-se de certificados para a validação de chaves públicas RSA. Neste trabalho, foi feito um teste sobre os módulos RSA contidos nos certificados emitidos sob regime da ICP-Brasil. Para tal foi utilizado um algoritmo de MDC estendido de forma a verificar se existe algum primo em comum entre dois módulos RSA diferentes. Com isso, investigou-se a existência da algumas falhas específicas, de modo a concluir fatos sobre a robustez das chaves RSA emitidas sob regime da ICP-Brasil. Para o teste foram analisados 1153 certificados emitidos sob regime da ICP-Brasil encontrando um caso de módulos repetidos.

Palavras-chave: chaves públicas RSA,módulo RSA,máximo divisor comum,teste de robustez,ICP-Brasil

Abstract

Public key infrastructures are essencial in data security and for those infrastructure to work properly digital certificates are used to validate RSA public keys. In this work, it is executed a test at RSA moduli contained in digital certificates of ICP-Brazil. To achieve this goal, an extended comom greatest divisor algorithm was utilized. Investigating the existence of certain flaws at RSA keys that can compromise it's use.

Keywords: RSA public keys, RSA modulus, greatest commom divisor,digital certificates

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Cenário da segurança na informática

A segurança na informática está diretamente relacionada com a proteção de um conjunto de dados, no sentido de preservar o valor que possuem para um indivíduo ou uma organização [27]. São características básicas da segurança na informática as primitivas criptográficas de sigilo, integridade, disponibilidade e autenticidade, não estando a segurança restrita somente a sistemas computacionais, dados eletrônicos ou sistemas de armazenamento. Pode-se observar que valores econômicos enormes dependem dessas garantias oferecidas pelo uso correto da criptografia, dessa forma a criptografia ocupa um espaço cada vez maior na sociedade.

Hoje em dia, a segurança na informática está disseminada em diversas empresas e órgãos, já que as mesmas atribuem grande valor aos seus dados armazenados e por isso tem que proteger o valor associado a esses dados com uso da criptografia. Onde há valor associado a dados e um possível interesse de atacantes sobre esses dados cabe a aplicação de criptografia. E devem ser observadas boas práticas para aplicação da mesma.

É muito comum observarmos falhas no uso de criptografia. Estas falhas advêm muitas vezes da utilização ingênua das metodologias da segurança na informática, como a não observação das premissas necessárias para o funcionamento, pouca habilidade técnica na implementação das tecnologias ou até descaso. Por isso há muitos esforços para instruir as pessoas ao bom uso da criptografia. Neste trabalho alerta-se sobre a importância de algumas práticas na criptografia e alertamos sobre possíveis perigos.

1.2 Introdução ao trabalho

A criptografia é uma realidade na segurança na informática pessoal ou de grandes empresas. Diversos novos protocolos criptográficos vêm sendo propostos, nesse contexto os algoritmos assimétricos vem se destacando. O algoritmo assimétrico mais usado atualmente é o RSA (introduzido a frente no trabalho), que foi possível a partir da ideia de derivação de chaves de Diffie Helmann [16].

A chave RSA é composta por um módulo grande (número com mais de 1024 bits), produto de pelo menos dois números primos, e estes devem ser gerados aleatoriamente, com entropia máxima. Dessa forma a fatoração do módulo é inviável, já que não temos

algoritmos de fatoração eficientes para essa tarefa. Caso alguma das recomendações na geração de chaves RSA não seja seguida diversos ataques ao protocolo RSA podem ser realizados.

Ataques ao protocolo RSA podem acarretar uma série de prejuízos a indivíduos e empresas. Para citar, a quebra do sigilo de dados sensíveis de uma empresa ou uma autenticação indevida por um usuário malicioso.

Existem diversas falhas que podem comprometer o funcionamento do RSA, porém, o principal objetivo deste trabalho é coletar uma amostra de certificados digitais emitidos pela icp-brasil e extrair suas respectivas chaves públicas para testar se as mesmas possuem falhas em sua geração, falhas que serão explicadas mais à frente no trabalho.

Resumidamente, o foco do trabalho é a descoberta de possível baixa entropia na geração de chaves RSA no contexto da ICP-Brasil, infraestrutura de chaves públicas brasileira. O método consiste em obter um banco de certificados de chave pública com o maior número possível de certificados. Depois disso, aplicar o algoritmo de MDC estendido entre todos os pares de módulos obtidos desses certificados, verificando assim, se há reuso de números primos em chaves distintas.

Se a geração de primos aleatórios estiver sendo feita de forma correta, o MDC de todos os pares de módulos de chave pública contidos nos certificados será igual a 1 com probabilidade altíssima. Caso algum MDC seja diferente de 1 é possível aplicar um ataque às chaves comprometidas tornando-as ineficazes, tanto para cifragem quanto para assinatura.

Com isso, pode-se fazer uma análise da qualidade das chaves públicas RSA que estão sendo emitidas no regime da ICP-brasil, contribuindo para alertar sobre possíveis falhas de implementação na geração de chaves RSA neste contexto, dada a importância da dimensão jurídica da certificação digital no Brasil.

1.3 Motivação para o trabalho

Recentemente foram publicados trabalhos para verificar se as chaves públicas contidas em certificados na web estão sujeitas a falhas graves de implementação. Existem diversos atributos de chaves públicas para que as mesmas sejam consideradas robustas:

- Os primos P e Q não devem ser muito próximos, já que há um algoritmo eficiente que fatora P*Q quando isso acontece, a técnica usa tentativas para fatorar com primos próximos a $\sqrt{P * Q}$, diminuindo drasticamente o tempo de fatoração.
- O expoente não deve ser muito pequeno, pois dessa forma, em poucas iterações é possível decifrar mensagens cifradas utilizando logaritmos modulares.
- A geração dos primos P e Q deve ser feita usando um bom algoritmo PRNG (gerador de números "pseudo-randômicos") e com uma fonte de entropia satisfatória e não enviesada ([29], [26]).

Há dois trabalhos publicados recentemente que focam na qualidade dos primos gerados para compor o módulo RSA, os trabalhos são "mining your P's and Q's"[24] e "Ron is wrong, Whit is right"[28].

Estes autores fizeram um ótimo trabalho e constataram que a geração de primos não tem a entropia desejada, fazendo com que os módulos tenham primos repetidos entre

outros módulos muito mais frequentemente que o esperado. O resultado obtido em [28] é surpreendente pois mostra que 2 em 1000 chaves públicas RSA não oferecem segurança adequada. Dessa forma fica um grande alerta às implementações que usam RSA, para que fiquem atentos à correta geração de primos, pois implementações defeituosas com baixa entropia na geração de chaves as prejudicam, como será mostrado à frente.

Com base nas técnicas descritas nos trabalhos citados propõe-se, neste trabalho, fazer algo similar. Nos trabalhos citados acima os certificados de chave pública foram obtidos de forma irrestrita, em toda web. Neste trabalho o objetivo será aplicar a metodologia proposta, porém desta vez, no contexto do Brasil, mais especificamente sob certificados emitidos em regime da ICP-Brasil.

Dessa forma, visamos adquirir um banco de certificados tão grande quanto possível de certificados da ICP-brasil. Assim, testaremos se os certificados de chave pública obtidos não reusam primos, como desejado. Utilizando o algoritmo de Euclides de MDC e mostrando se há primos repetidos em diferentes certificados, o que mais tarde mostrarei que invalida o uso da chave pública onde ocorre reuso de números primos.

Ao final do teste verificaremos se há reuso indevido de números primos entre certificados. E poderemos demonstrar a viabilidade de ataque contra sigilo ou integridade (falsificação de assinatura digital) em caso de haver chaves comprometidas. De acordo com o resultado do teste poderemos concluir se as chaves estão sendo geradas da forma esperada ou não, na ótica da correta geração de primos para uso em protocolos RSA.

No caso de o teste corresponder à expectativa estatística, poderemos dizer que as chaves estão seguras quanto ao problema detectado nos trabalhos motivadores citados. Caso contrário poderemos afirmar que está ocorrendo alguma falha no processo de geração de chaves RSA utilizado por certificadoras homologadas pela ICP-Brasil.

1.4 Trabalhos relacionados

A maior inspiração para este trabalho foram os trabalhos de Lenstra: *ron is wrong, whit is right* [28]; e o trabalho de Alex Haldermann: *Mining your P's and Q's* [24]. Onde dou destaque para o trabalho Mining your P's and Q's, um excelente trabalho, sendo ótima referência para qualquer trabalho futuro nesta área.

Alex Haldermann e sua equipe fizeram a maior varredura na internet até o momento à procura de chaves públicas RSA. Para tal utilizaram a ferramenta NMAP, que teve grande importância para descobrir, dentro do espaço público de IP's, quais ofereciam serviço HTTPS na porta 443 ou SSH na porta 22. A partir disso, fecharam conexão com os IP's encontrados para obter o certificado de chave pública utilizado pelo servidor. Criando assim, um vasto banco de certificados X.509 e de chaves públicas SSH.

A partir daí extraíram os módulos da respectivas chaves e com eles fizeram um teste automatizado utilizando um algoritmo de MDC otimizado. E, com isso, foi possível observar falhas de entropia na geração aleatória de chaves RSA. Para resumir os resultados, eles encontraram 0.50% de chaves RSA comprometidas, permitindo o cálculo da chave privada correspondente (quebra total da segurança oferecida pela chave citada). O que é um número alarmante. Além disso, um grande número de chaves públicas repetidas foi encontrado, o que muitas vezes pode ser considerada uma falha de implementação, especialmente com grandes consequências em regimes jurídicos como o da ICP-Brasil como será mostrado no capítulo 4.

Ainda sobre as chaves públicas repetidas encontradas nos trabalhos citados deve-se destacar que este tipo de falha tem sérias implicações negativas no uso em protocolos de assinatura digital. O que ocorre é que quando uma entidade assina um documento digital espera-se que, como na assinatura de punho, só aquela entidade tenha possibilidade de assinar um documento com aquela assinatura. Porém, quando existem vários agentes possuindo aquela chave pública e sua respectiva chave privada o documento pode ter sido assinado por qualquer um desses agentes. Dessa forma, a assinatura digital não mais indica a origem daquele documento unicamente. Então, não seria correto, por exemplo, responsabilizar alguma pessoa pelo conteúdo de um documento, já que não se sabe ao certo qual dos agentes que possuem as mesmas chaves assinou o documento. Fato que prejudicaria o regime jurídico da ICP-brasil, onde o ônus da prova é invertido. [1]

Os autores dos trabalhos citados fizeram também uma grande investigação para achar o motivo de tais falhas na geração de chaves. Uma hipótese sustentada por eles foi o mau uso de um PRNG encontrado no linux, o *urandom*. Que pode ter sido utilizado sem a coleta de entropia adequada pelo sistema operacional, gerando números de certa forma previsíveis. Outra hipótese foi a utilização de chaves *default*, que não deveriam ser usadas de maneira alguma, nem ao menos deveriam ser fornecidas chaves desse tipo, que comprometem o uso do protocolo TLS. E a última hipótese foi a de má implementação da geração de chaves em dispositivos embarcados, que por sua simplicidade, não coletam entropia adequada para gerar chaves criptograficamente seguras ou não dão devida atenção à implementação.

Ainda sobre as contribuições deste artigo motivador, é possível ver como a geração de números randômicos é importante na criptografia, e que, mesmo com muita teoria já escrita à respeito, ainda se cometem muitas falhas na geração de números randômicos na prática. Estes erros comprometem o uso do algoritmo RSA. No artigo são propostas diversas soluções para usuários, fabricantes de sistemas operacionais e para quem utiliza estes protocolos em geral. Tudo isso pode ser consultado em [24] e no site que foi disponibilizado para este tópico pela mesma equipe: https://factorable.net/.

1.5 O problema

O problema que se tem como foco neste trabalho é de que no Brasil, com a recente legislação sobre certificação digital, coloca-se muita responsabilidade jurídica sobre quem usa, ou pior, quem é obrigado a usar esta tecnologia, como pessoas ligadas ao judiciário. Porém, como mostrado acima, existem algumas falhas que comprometem o uso de certificação digital, de forma que seria desaconselhável acreditar inteiramente no uso eficaz da certificação digital. Dessa forma, faremos uma investigação para verificar se as mesmas falhas apontadas por Haldermann em [24] acontecem no Brasil em certificados emitidos no regime da ICP-brasil.

1.6 Hipótese

Temos como hipótese que a geração de chaves RSA possa não estar sendo feita de forma adequada nos certificados emitidos sob regime da ICP-brasil. E que há uma baixa

[1] Legislação descrita no parágrafo primeiro, artigo décimo da medida provisória 2200-2

na entropia esperada dos pseudo-geradores de números randômicos utilizados. Incorrendo na geração de chaves criptográficas ineficazes.

1.7 Objetivos

- Coletar o máximo possível de certificados de chave pública emitidos sob regime da ICP-brasil. De forma a ter uma amostra representativa de certificados.
- Implementar um programa que automatize os testes sobre os módulos contidos nos certificados obtidos, aplicando algoritmo eficiente de máximo divisor comum estendido sobre os módulos das chaves dois a dois.
- Analisar os resultados e indicar a quantidade de certificados com colisão de primos.
- Analisar os tipos de ataque possíveis a pares de chaves conhecidamente comprometidos, para dar prova de conceito sobre a importância desse teste de robustez.
- Mostrar certificados com colisão de primos obtidos e propor hipóteses sobre as circunstâncias de emissão desses certificados, analisando os dados contidos nos mesmos.

Capítulo 2

Introdução teórica

2.1 Criptografia assimétrica

O maior problema encontrado na criptografia simétrica está na dificuldade de compartilhar chaves secretas numa rede aberta insegura. Qualquer pessoa que conheça a chave secreta pode decifrar uma mensagem cifrada com a mesma. Uma resposta é a criptografia assimétrica, em que há duas chaves relacionadas - um par de chaves. Uma chave pública é disponibilizada a qualquer pessoa que queira fazer uso da mesma. Uma segunda chave, a privada, é mantida em sigilo para que somente seu titular a conheça [34].

Qualquer mensagem (texto, arquivos binários ou documentos) que é cifrada usando a chave pública só pode ser decifrada aplicando o mesmo algoritmo utilizando a chave privada correspondente. Similarmente, na assinatura digital, qualquer mensagem que é assinada utilizando uma chave privada só pode ser verificada utilizando a chave pública correspondente.

A inviabilidade de se obter a chave pública com a chave privada, característica dos algoritmos assimétricos, significa que pode-se compartilhar chaves públicas às claras na rede. Um problema com a criptografia assimétrica, no entanto, é que é mais lenta que a criptografia simétrica. Ela requer muito mais capacidade de processamento para criptografar e descriptografar uma mensagem.

Outro fator importante na criptografia assimétrica que deve ser levado em consideração é que a chave pública deve ser enviada para o interessado com integridade de origem e de conteúdo. Em outras palavras, deve-se garantir que a chave utilizada veio realmente da pessoa que se espera, e não de um terceiro tentando fraudar a origem da comunicação. Isso pode parecer um problema simples, mas em rede aberta isto é complexo e deu origem a infraestrutura de chaves públicas.

Outra diferença entre criptografia simétrica e assimétrica é o tamanho das chaves utilizadas. Para uso eficaz do algoritmo RSA, por exemplo, necessitamos de chaves muito grandes, da ordem de 2048 bits. Já nos algoritmos simétricos robustos, chaves de 128 bits já são satisfatórias. Isso vem da necessidade do módulo (componente das chaves RSA) no algoritmo RSA ser inviável de fatorar, já em algoritmos simétricos ditos robustos, só temos a necessidade de ter um espaço de chaves possíveis grande, de modo a evitar ataques por força bruta.

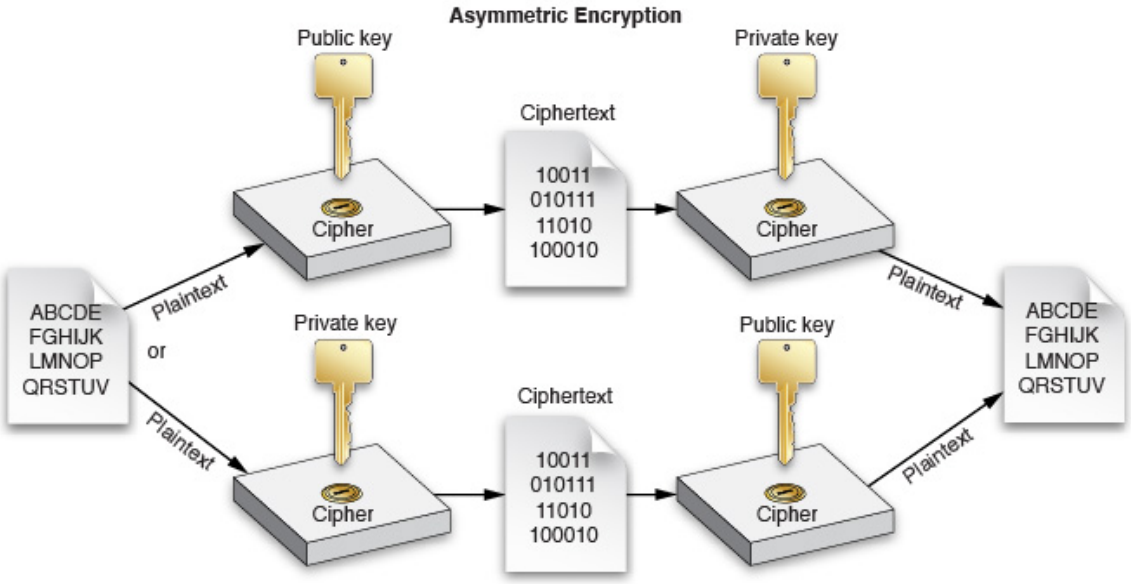

Figura 2.1: Esquema de criptografia assimétrica [1]

2.2 Algoritmo RSA

Para entender melhor a metodologia utilizada no trabalho deve-se primeiro entender o funcionamento do algoritmo RSA [32].

O RSA é um algoritmo criptográfico assimétrico. Diz-se que um algoritmo é assimétrico se é inviável deduzir uma chave a partir de seu par. Portanto, a chave pública pode ser transmitida em rede aberta às claras.

O RSA determina que um agente gere o par de chaves (pública e privada). E mantenha a chave privada consigo mesmo. A chave pública deve ser enviada para quem ele deseja ter sigilo na transmissão de dados ou para um possível verificador de sua assinatura. A chave pública tem o formato *(M,e)* e a chave privada tem o formato *(M,d)*. Onde 'M' é o produto entre dois números primos grandes. Geralmente 'M' tem mais que 1024 bits. Se considerarmos que $M = p_1 * p_2$,então:

$$e * d \equiv 1 \mod (p_1 - 1) * (p_2 - 1)$$

Para utilizar o RSA na cifragem, onde 'm' é uma mensagem qualquer e 'c' é a cifra dessa mensagem, considerando m $<$ M como representação de inteiros, deve-se fazer:

$$m^e \mod M \equiv c$$

E para decodificar o significado da cifra, tendo a chave privada faz-se:

$$c^d \mod M \equiv m$$

E para protocolo de assinatura digital deve-se gerar o assinador 's' da seguinte forma:

$$(hash(m))^d \mod M \equiv s$$

E para verificar a integridade da mensagem recebida deve-se fazer: $s^e \mod M$, e verificar se este valor é igual a hash(m), com 'm' sendo a mensagem recebida pelo verificador.

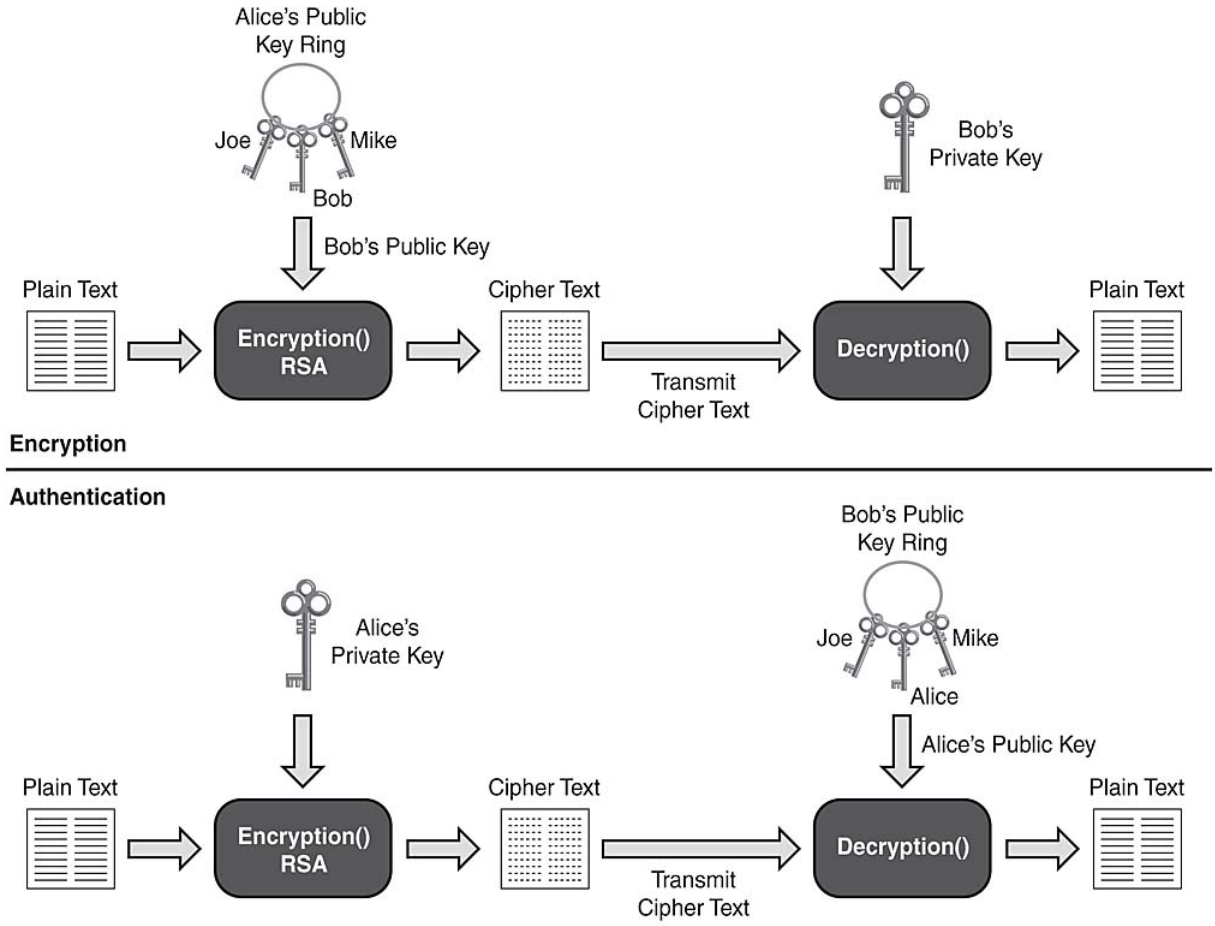

Figura 2.2: Protocolos para autenticação e sigilo com RSA [7]

2.3 Importância da geração de primos aleatórios para chaves RSA

A geração de números pseudo-aleatórios é essencial para a criptografia moderna, onde a segurança depende do fato das chaves serem escolhidas de forma aleatória em um espaço de chaves possíveis. É fácil entender a necessidade de uma boa implementação de geradores de números aleatórios no RSA. Digamos que temos um espaço de primos possíveis para comporem um módulo RSA de 1024 bits de 2^{503} possibilidades, conforme o teorema da distribuição dos números primos [12] e desconsiderando os primos que não são recomendáveis para compor um módulo RSA. Dessa forma se temos um ótimo algoritmo de geração de primos randômicos neste espaço amostral teremos 2^{503} possibilidades de primos a serem escolhidos efetivamente.

Este número citado acima é notoriamente grande, e é um número plausível para implementações do RSA com módulos de 1024 bits ou mais. Dessa forma para uma fatoração

por força bruta seria totalmente inviável para o atacante percorrer um espaço de 2^{503} primos dados os recursos computacionais e algoritmos que temos hoje em dia, levando aproximadamente 2^{450} anos. O que faria que a chave estivesse protegida contra este tipo de ataque.

Porém, se tivéssemos uma implementação ruim de gerador de números aleatórios que, hipoteticamente, só gerasse 2^{20} primos dentro do espaço de primos citado acima, teríamos um problema de entropia no gerador utilizado. Dessa forma, o atacante teria um espaço de primos bem menor a ser percorrido. Nessa situação, a baixa entropia dos primos gerados viabilizaria um ataque de força bruta.

Como o módulo RSA usa 2 primos, uma baixa entropia no PRNG utilizado na geração de chaves aumenta proporcionalmente a probabilidade de um mesmo primo ser escolhido para compor dois módulos RSA em chaves diferentes. Esta é a hipótese que os trabalhos de referência investigam, e que, este trabalho investiga no contexto das chaves geradas sob regime da ICP-Brasil.

Sabemos que o reuso de primos torna o RSA ineficaz já que facilita imensamente a fatoração por máximo divisor comum e posteriormente o ataque à chave. Para exemplificar, se a entropia da geração de primos for dada apenas pelo tempo como em *srand(time(NULL))* na linguagem C estaríamos correndo um sério risco de gerar primos com entropia inadequada.

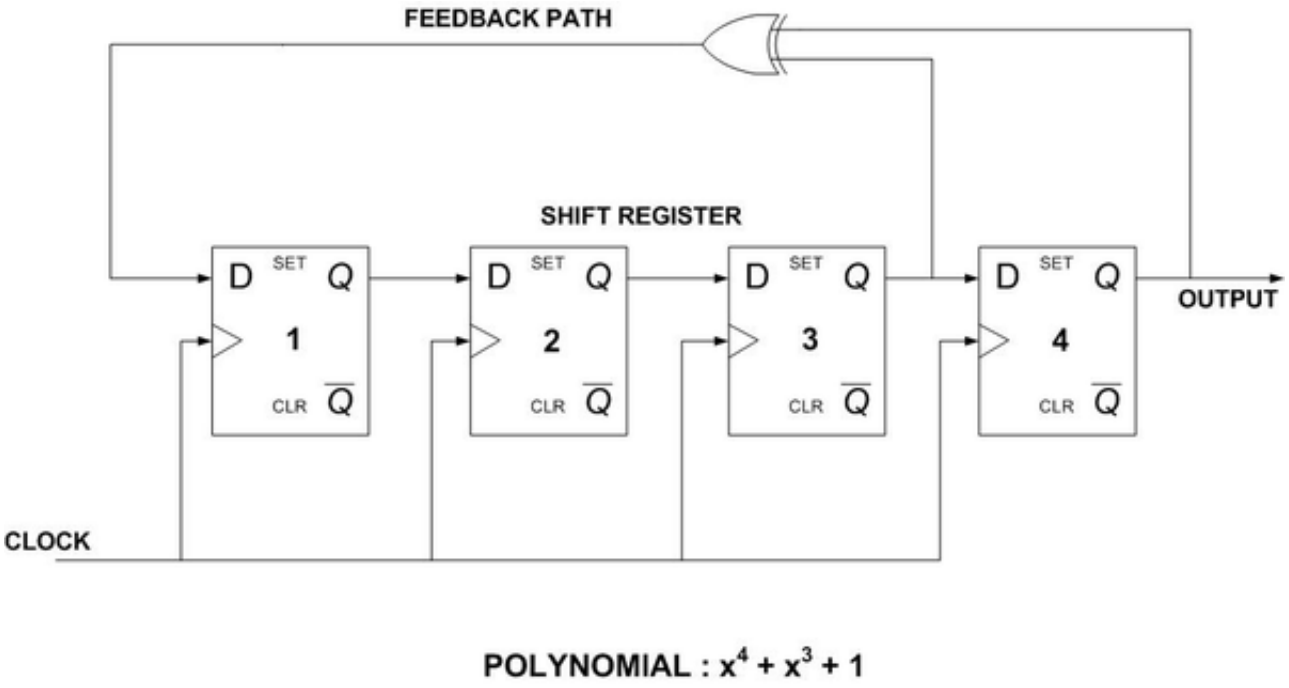

Figura 2.3: Funcionamento de um PRNG por linear feedback [5]

Input: $a, b \in R$.

Output: $g \in R$ the gcd of a and b together with $s, t \in R$ such that $g = s\,a + t\,b$.

$u_0 := \mathrm{lc}(a)$; $r_0 := \mathrm{normal}(a)$; $s_0 := u_0^{-1}$; $t_0 := 0$
$u_1 := \mathrm{lc}(b)$; $r_1 := \mathrm{normal}(b)$; $s_1 := 0$; $t_1 := u_1^{-1}$
$i := 2$
while $r_{i-1} \neq 0$ **repeat**
  $q_i := r_{i-2}$ quo r_{i-1}
  $r_i := r_{i-2}$ rem r_{i-1}
  $u_i := \mathrm{lc}(r_i)$
  $r_i := \mathrm{normal}(r_i)$
  $s_i := (s_{i-2} - q_i\, s_{i-1})/u_i$
  $t_i := (t_{i-2} - q_i\, t_{i-1})/u_i$
  $i := i + 1$
return$(r_{i-2}, s_{i-2}, t_{i-2})$

Figura 2.4: Algoritmo de euclides para obtenção do máximo divisor comum [2]

2.4 Fatoração do módulo RSA utilizando MDC

Adiante, veremos que a fatoração dos módulos RSA de fato inviabiliza o uso eficaz do RSA para o par de chaves comprometido. Como até hoje não temos algoritmos de fatoração eficientes o bastante para fatorar módulos grandes como de 2048 bits ou 4096 bits em tempo hábil o RSA se mostra um algoritmo eficaz, porém, o progresso da computação quântica pode comprometer o RSA, dado que a fatoração seria possível neste contexto.

O método de fatoração descrito neste trabalho pode reconhecer em tempo computacionalmente viável se há colisão de primos em dois módulos diferentes. Consta na figura 2.4, o algoritmo utilizado para obter o máximo divisor comum entre a e b.

2.5 Conceito de entropia

Em teoria da informação, a entropia (mais especificamente, entropia de Shannon) é o grau de incerteza contido em uma sequência de dados, segundo Shannon em [33] e [17]. Entropia caracteriza, assim, nossa incerteza sobre a nossa fonte de informação. Uma fonte de entropia também é caracterizada por uma distribuição de probabilidade das amostras

recolhidas. Sendo que a distribuição de probabilidades homogênea é a que fornece maior grau de entropia.

2.6 Entropia nos PRNG's atuais

Hoje em dia muitos sistemas operacionais fornecem implementações de geradores de números randomicos, porém já foram feitas observações de que alguns não geram a entropia mínima desejada.

Majoritariamente os sistemas operacionais tem como fonte de entropia eventos observados na execução do SO, como tempo da última leitura em disco, local da cabeça de leitura do disco rígido, memória RAM em utilização, tabela de processos e diversas fontes de entropia dentro do sistema.

Porém, Lenstra et al. em [28], observou que se o sistema tiver sido inicializado há pouco tempo, muitas dessas fontes de entropia não vão existir. Dessa forma o SO falha em fornecer um grau de entropia desejável para protocolos criptográficos que depende estritamente de um PRNG bem implementado e com entropia satisfatória. Outra falha observada, principalmente no linux, é que a função padrão de randomizacão (urandom, não blocante) não gera nenhuma alerta caso o nivel de entropia obtido no sistema esteja baixo, e fornece output mesmo assim.

O recomendado seria que o sistema operacional reconhecesse a baixa entropia e gerasse um número aleatório apenas quando o nivel de entropia fosse considerado satisfatório. Além disso, diversas implementações de PRNG's não levam a sério as boas práticas de implementação, e usam de fontes de entropia consideradas fracas, como tempo decorrido da inicialização do sistema ou tempo que a funcão foi chamada.

2.7 Assinatura digital

Em criptografia, a assinatura digital é um método de autenticação objetiva (oponível a terceiros) de dados digitais tipicamente tratada como análoga à assinatura física em papel [21]. A função é muito parecida com a função de uma assinatura de punho, que é um meio de alguém mostrar consentimento com o que foi escrito por meio de associar sua assinatura ao teor do documento, esta deve ser única, intransferível e verificável por terceiros.

A utilização da assinatura digital serve para associar um conteúdo digital a um agente, se todas as premissas do protocolo forem satisfeitas. Para verificar este requisito, um esquema de assinatura digital deve ter as seguintes propriedades:

- autenticidade - o receptor deve poder confirmar que a assinatura foi feita pelo emissor;
- integridade - qualquer alteração da mensagem faz com que a assinatura não corresponda mais ao documento;
- irretratabilidade ou não-repúdio - o emissor não pode negar a autenticidade da mensagem.

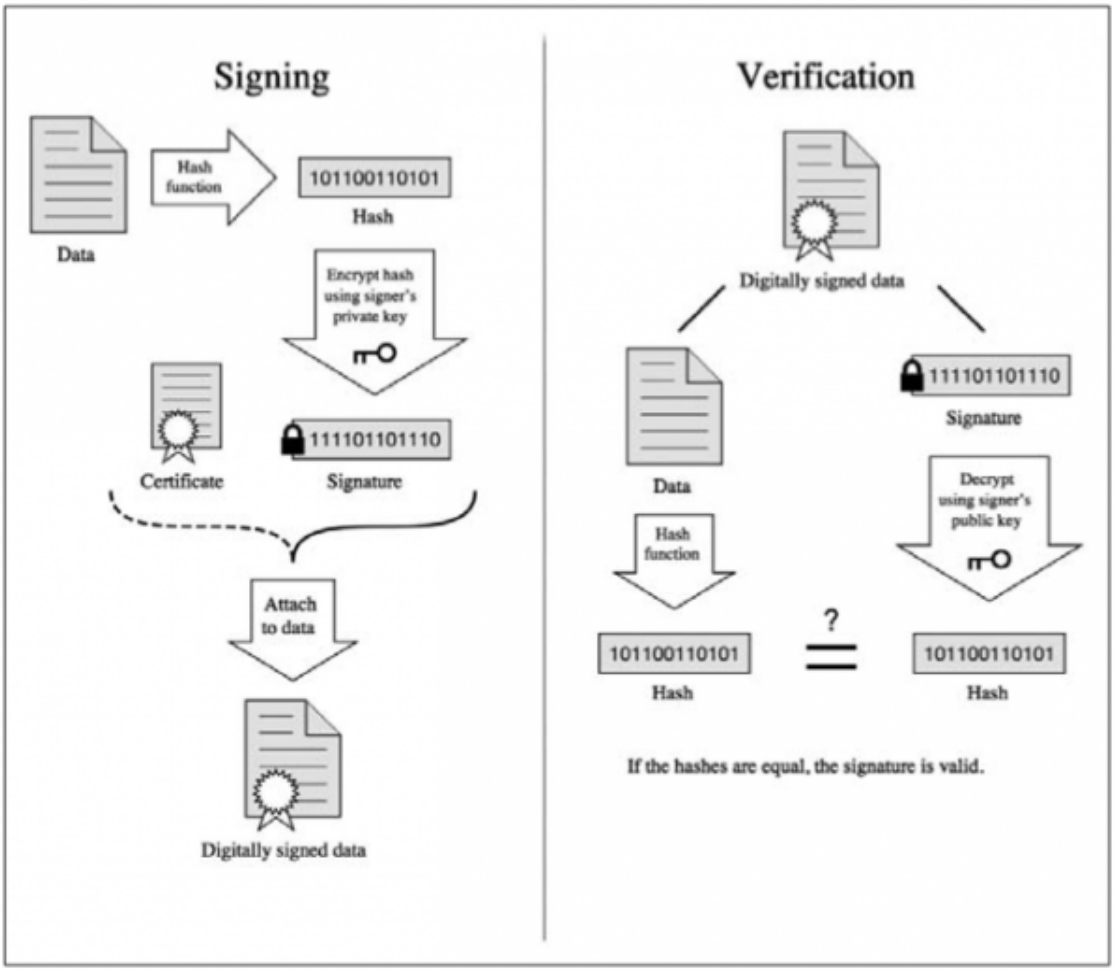

Figura 2.5: Funcionamento da assinatura digital [8]

2.8 Certificados de chave pública

Os certificados de chave pública são arquivos digitais que tem como objetivo a comunicação de uma chave pública para interessados de forma a garantir a origem desta chave [20]. Isso é feito comprovando-se que a chave pública é da pessoa que se diz dono dela no certificado por meio da assinatura digital de uma terceira entidade que é considerada confiável, em relação aos dados contidos no certificado, por uma cadeia de certificações. A assinatura emitida verifica a validade da chave pública e a pertinência da mesma a alguém ([14],[11]).

Além disso, deve-se verificar se o emissor do certificado é confiável, para isso deve haver alguma autoridade certificadora fechando a cadeia de certificação [35].

Hoje em dia temos a padronização destes arquivos de certificado pela ITU-T, de nome X.509. O padrão X.509 especifica o formato de certificados de chave pública, certificados de autoridades certificadoras, requisições de assinatura de certificados e outros arquivos necessários para a infraestrutura de chaves públicas. Os certificados digitais tem quatro campos básicos:

- Identificação do sujeito que está transmitindo sua chave pública, titular da mesma.
- A chave pública, composta por módulo e expoente no caso do RSA.
- A identificação do emissor do certificado, que assina o conteúdo do certificado, atestando que é válida a chave pública e que a mesma pertence ao sujeito identificado como titular.
- A assinatura do emissor do certificado, que pode ser verificada com a chave pública do emissor.

2.9 Utilização de certificados digitais

Os certificados digitais tem estas 4 utilidades primárias:

- Identificação:

  A autoridade certificadora (CA) atesta a identidade do titular do certificado pela assinatura emitida por esta CA contida no certificado.

- Sigilo:

  A chave pública contida no certificado pode ser utilizada para a cifragem de dados de forma que somente o dono da chave privada possa decifrar os dados e obter informação desses dados, conforme parâmetros no campo "Uso da chave pública".

- Integridade:

  A chave pública transportada pode ser utilizada por algum agente que deseja verificar a origem e integridade de um documento assinado digitalmente pela chave privada correspondente.

- Controle de acesso:

Controle de acesso pode ser implementado a partir de certificados digitais com o uso do mesmo para identificação, substituindo, no servidor, mecanismos que usam senha.

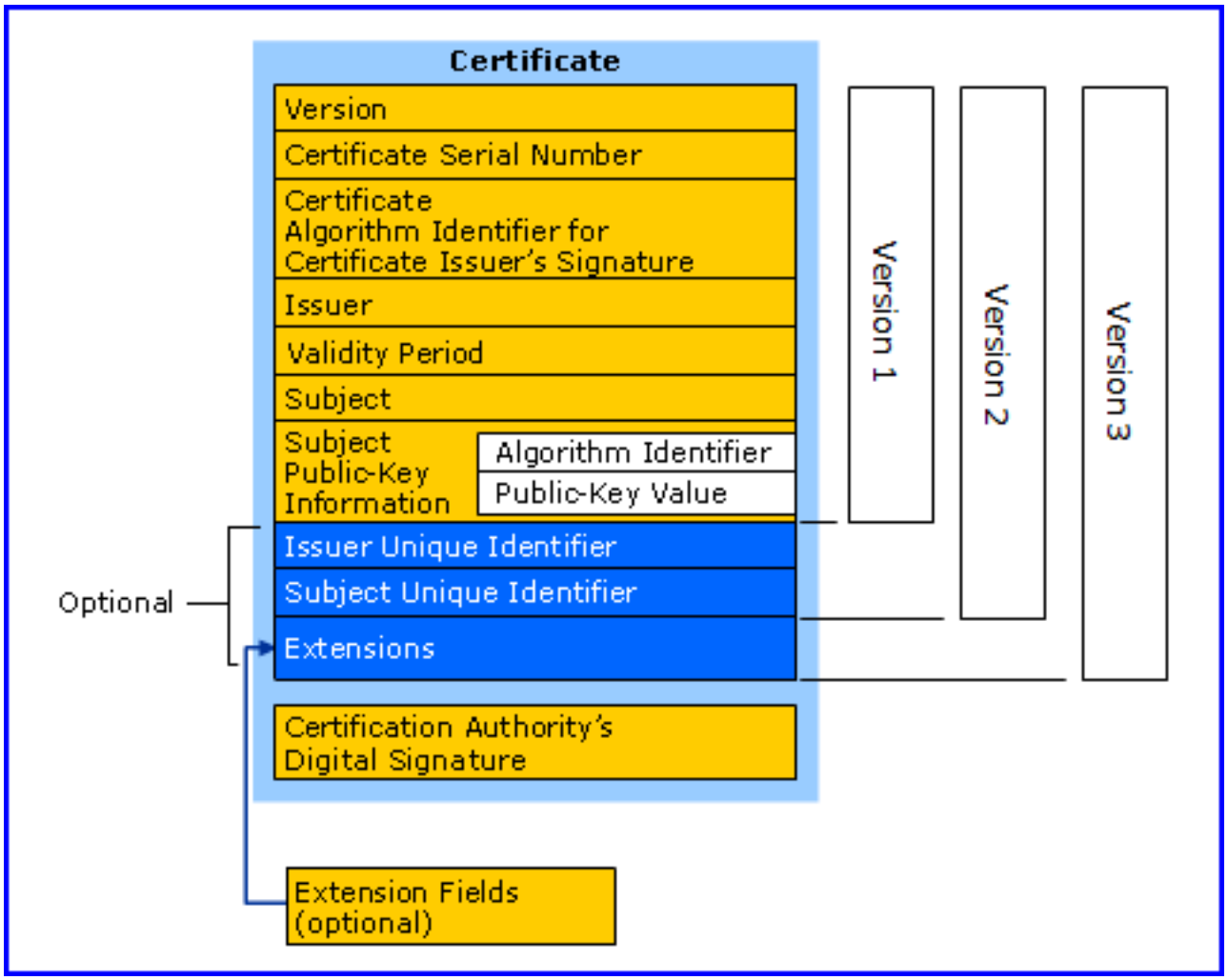

Figura 2.6: Formato de certificados digitais X.509 [4]

2.10 Infraestrutura de chaves públicas

A infraestrutura de chaves públicas é um conjunto de protocolos dedicados a produzir, utilizar, trasmitir e validar certificados digitais de chave pública. A ICP (infraestrutura de chaves públicas) é um conceito abstrato que abrange diversas técnicas que possibilitam a segurança em rede aberta a partir da utilização correta de chaves públicas [18].

Toda ICP depende de autoridades certificadoras e políticas bem definidas para geração e transmissão de certificados, geralmente definidas pelas próprias autoridades certificadoras ou orgãos de padronização.

O maior desafio é oferecer segurança nas informações transmitidas em rede aberta, para tal deve-se oferecer integridade das chaves públicas trasmitidas nesse contexto. Para isso a

ICP implementa cadeias de certificados que terminam em algum certificado auto assinado de uma autoridade certificadora que deve ter sua origem e integridade reconhecidas pelo aplicativo do cliente. São condições mínimas para que o cliente possa confiar que a chave pública recebida pelo mesmo pertence de fato ao titular indicado no certificado.

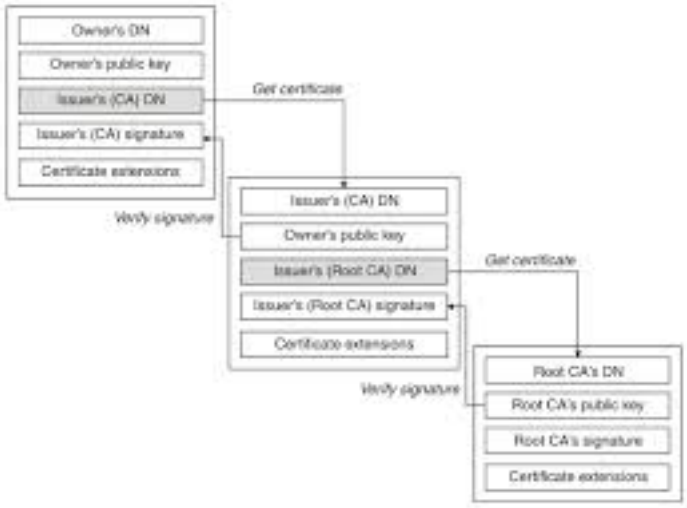

Figura 2.7: Cadeia de certificados [8]

2.11 ICP Brasil

A ICP-Brasil (Infraestrutura de chaves públicas brasileira), http://www.iti.gov.br/icp-brasil, foi implementada em 2001 pela medida provisória 2200 (MP2200), e é uma insfraestrutura de raiz única, que é a AC RAIZ. A ICP Brasil foi proposta para que o Brasil tivesse uma propria autoridade certificadora que fosse utilizada para emitir certificados no contexto do ordenamento jurídico brasileiro. Para isso, os documentos assinados por chaves públicas reconhecidas pela ICP-Brasil tem valor jurídico disciplinado pela MP2200.

A infraestrutura está crescendo, e já conta com 13 autoridades certificadoras validadas pela ICP-Brasil, com cerca de 8 milhões de certificados já emitidos [1]. Essas validações dependem do compromisso das AC's com as políticas definidas pela ITI.

A ICP-Brasil não conta com seu certificado raiz embutido nos binários dos navegadores utilizados hoje em dia. O que representa um problema, pois para que a ICP tenha validade os clientes devem instalar manualmente o certificado raiz em seus navegadores ou aplicativos. O que pode ser inseguro, já que é feito em rede aberta.

Mesmo assim, a ICP-Brasil está se tornando uma realidade no Brasil, com diversas autoridades certificadoras credenciadas e grandes empresas e orgãos públicos utilizando a infraestrutura. As leis introduzidas e as políticas definidas pela ICP-Brasil geram controvérsia, e alguns desconfiam da corretude dos procedimentos utilizados para transmissão e autenticação de chaves públicas. Neste trabalho queremos testar certificados emitidos

[1] Conforme declaração do presidente da associação de entidades de registro da ICP-BR, Nivaldo Cleto, em evento do comitê gestor da internet no Brasil

pela ICP-Brasil para verificar a robustez destas chaves públicas no quesito da geração de chaves RSA.

A Medida Provisória nº 2.200-2, de 24 de agosto de 2001 define as regras para a criação da ICP-Brasil e da DPC (declaração de práticas de certificação) associada, bem como a utilização de certificados digitais no Brasil, aspectos legais e aspectos necessários para uma entidade se tornar uma AC Intermediária e assim emitir certificados digitais para outras entidades "garantindo"autenticidade, integridade, não repúdio e validade jurídica de trâmites eletrônicos por decreto.

A Lei 11.419 de 19 de dezembro de 2006 fundamenta os processos judiciais eletrônicos no Brasil. Nela, existe o artigo 20 do capítulo 4, que altera o artigo 38 do Código de Processo Civil (Lei 5.869, de 11 de janeiro de 1973) de forma que a autenticação por certificados digitais também seja legalmente válida.

Essas mudanças de como os certificados digitais são encarados legalmente são recentes e mostram como a certificação está entrando no cotidiano dos brasileiros. A lei basicamente assegura que tudo que estiver assinado de forma válida segundo as políticas da ICP-Brasil tem valor jurídico com presunção de veracidade com inversão do ônus da prova[2]. Assim, deve-se atentar no uso dos certificados, que mesmo utilizando todos os cuidados possíveis podem ser envolvidos em casos de fraude.

2.12 O padrão X.509

O padrão X.509 começou em associação com o padrão X.500 em 1998 (versão 1) e assumiu um sistema hierárquico das autoridades de certificação para emissão de certificados [30]. Em 1993, uma versão melhorada do X.509 - versão 2, foi introduzida com a adição de mais dois campos, de apoio e controle de acesso de diretório.

A versão 3 do padrão foi implementada em junho de 1997 devido a ineficiência encontrada nas versões anteriores. Foram adicionados campos de extensão, o que torna o certificado mais flexível, com expansões na utilização. Adicionou-se a compatibilidade com outras topologias, como malhas e pontes, bem como a opção de usá-lo em uma rede P2P. Mais recentemente, foi lançada a versão 4 do padrão X.509. Esta versão está definida na recomendação X.509/ISO/TEC 9594-8 2001.

O X.509 é um padrão proposto pela ITU-T que traz especificações para infraestruturas de chaves públicas e estrutura de certificados de chave públicas, requisições de assinatura de certificados, validação de cadeias de certificados e lista de revogação de certificados. E até hoje este é o formato melhor aceito pelo mercado para codificar certificados, diversos padrões de armazenamento e transmissão utilizam estas especificações:

- .pem – (Privacy-enhanced Electronic Mail) É uma codificação em base 64 de certificados tipo DER, o formato é reconhecido pelas diretivas -—-BEGIN CERTIFICATE—–"e -—-END CERTIFICATE——"no inicio e fim do arquivo respectivamente.
- .cer, .crt, .der – São normalmente uma codificação do tipo DER em binário.
- .p7b, .p7c – São formatos que geralmente armazenam certificados ou requisisões de assinatura de certificados.

[2]Conforme o parágrafo primeiro do artigo décimo da medida provisória 2200-2

- .p12 – Contém certificados de chave pública e chaves privadas, com a ultima protegida por senha.
- .pfx – Predecessor do PKCS12.

Para ilustrar melhor como é a estrutura de um certificado X.509 a figura a seguir é um exemplo de um certificado de chave pública de acordo com os padrões definidos no X.509:

2.13 O protocolo SSL

O protocolo SSL foi criado pela Netscape com o objetivo de proporcionar mecanismos de autenticação e sigilo entre duas aplicações cliente-servidor que se comunicam via algum protocolo de comunicação (por exemplo, o http) [31]. Outros aspectos importantes considerados no momento de sua concepção foram: interoperabilidade, permitindo a comunicação com outra aplicação sem que haja a necessidade de entrar em detalhes a respeito de sua implementação; extensiblidade, que permite criar novas rotinas e funcionalidades baseadas em mecanismos pré-existentes do protocolo; por fim, eficiência, tornando o protocolo viável para o uso entre aplicações cliente-servidor via Internet.

A arquitetura do SSL é disposta em camadas, a exemplo do TCP/IP. Uma delas, a chamada Record Layer, recebe informações não encriptadas das aplicações, dispondo-as em blocos numerados seqüencialmente. Estes blocos então podem passar por uma compressão, seguida da geração de códigos de autenticação (apenas o MAC tem sido implementado). Em seguida, os blocos são encriptados com a chave de sessão transmitida via envelope digital durante a execução da camada inicializadora (*handshake layer*), e depois, enviados.

Os protocolos sobre os quais o SSL é construído incluem um especialmente concebido com o objetivo de sinalizar transações entre estratégias de cifragem usadas na sessão. Este protocolo é denominado Change Cipher Spec Protocol. Outro protocolo importante é o Alert Protocol, usado na sinalização de erros e em notificações de fechamento de conexão.

Ainda há o Handshake Protocol, que estabelece os parâmetros criptográficos da sessão, operando ao topo da Record Layer. No início da sessão, o cliente envia ao servidor uma 'hello message', informando os algoritmos e protocolos disponibilizados por sua implementação do SSL, sendo eles criptográficos ou não (por exemplo, os algoritmos de compressão associados também são informados). O servidor, baseado nos algoritmos e protocolos que a ele estão disponíveis, escolhe alguns dos parâmetros informados pelo cliente para serem usados no estabelecimento da sessão (conforme os que tem implementado), notificando o cliente através de uma outra 'hello message'.

Em seguida, o servidor envia seu certificado (ou informações associadas a um protocolo usado para troca de chaves, caso ele não tenha um certificado de chave pública ou seu certificado possa ser usado apenas para verificar assinaturas digitais), e o cliente opcionalmente faz o mesmo. Logo após, a chave de sessão é instituída, através dos métodos criptográficos estabelecidos na troca das 'hello messages' (por exemplo, Diffie-Hellmann ou RSA).

```
Certificate:
  Data:
    Version: 1 (0x0)
    Serial Number: 7829 (0x1e95)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
          OU=Certification Services Division,
          CN=Thawte Server CA/emailAddress=server-certs@thawte.com
    Validity
       Not Before: Jul  9 16:04:02 1998 GMT
       Not After : Jul  9 16:04:02 1999 GMT
    Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
          OU=FreeSoft, CN=www.freesoft.org/emailAddress=baccala@freesoft.org
    Subject Public Key Info:
       Public Key Algorithm: rsaEncryption
       RSA Public Key: (1024 bit)
          Modulus (1024 bit):
             00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
             33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
             66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
             70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17:
             16:94:6e:ee:f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:
             c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77:
             8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
             d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8:
             e8:35:1c:9e:27:52:7e:41:8f
          Exponent: 65537 (0x10001)
  Signature Algorithm: md5WithRSAEncryption
    93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
    92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
    ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67:
    d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:1b:41:72:
    0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1:
    5a:de:9d:ea:63:cd:cb:cc:6d:5d:01:85:b5:6d:c8:f3:d9:f7:
    8f:0e:fc:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22:
    68:9f
```

Figura 2.8: Exemplo de certificado de acordo com o padrão X.509 [9]

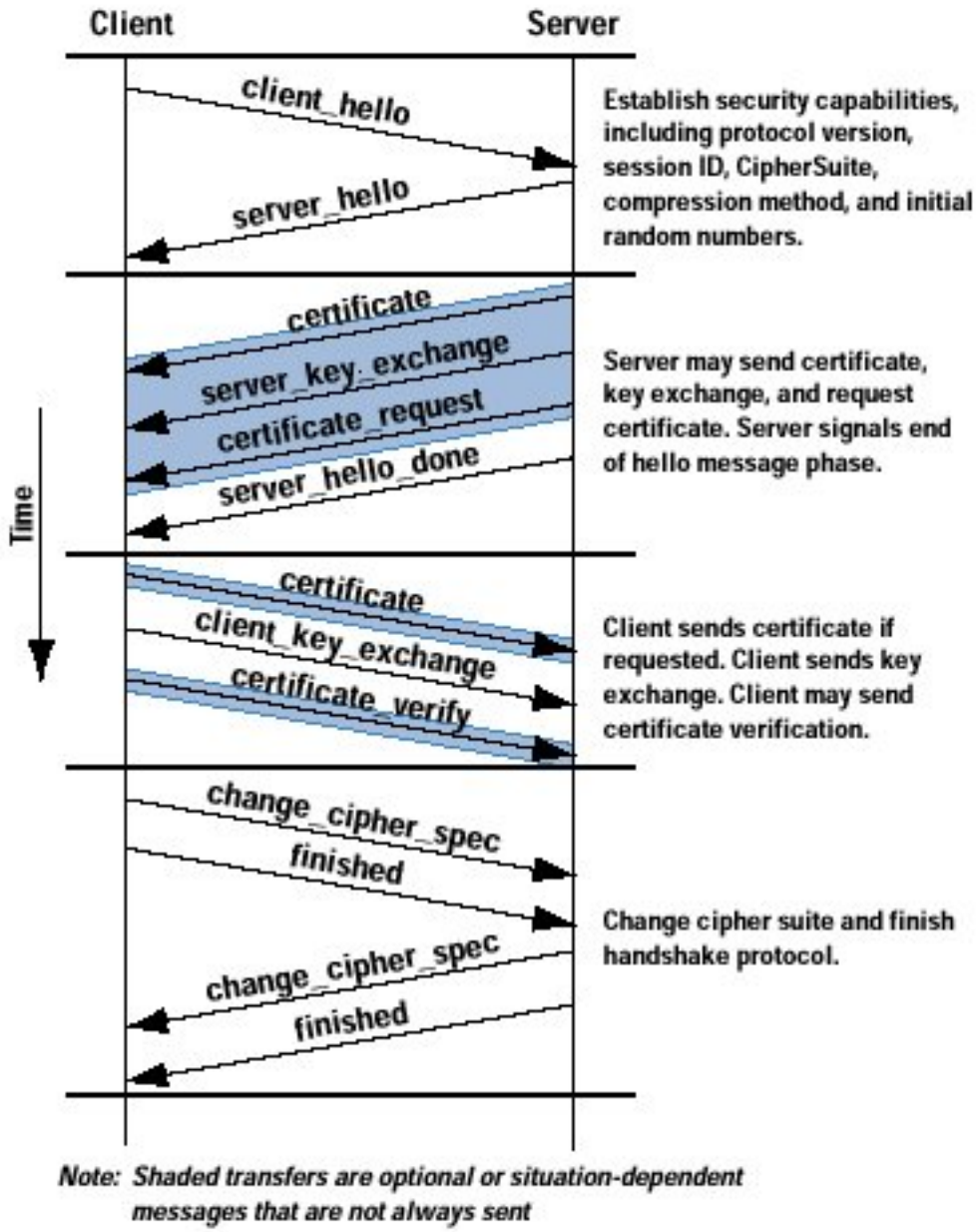

Figura 2.9: Funcionamento do protocolo SSL [6]

2.14 A biblioteca OpenSSL

A OpenSSL é uma biblioteca que implementa a comunicação em rede através do protocolo TLS e SSL. É uma das bibliotecas mais usadas no mundo para tal fim [22], [10]. A OpenSSL será muio útil neste trabalho pois ela implementa várias ferramentas para utilização de certificados em diversos formatos, inclusive o PEM, DER e CER.

O projeto OpenSSL disponibiliza um toolkit em código livre, que implementa o protocolo SSL e vários algoritmos e primitivas criptográficas de uso comum, incluindo algoritmos para troca de chaves, funções de *hash*, algoritmos simétricos e assimétricos. O *toolkit* se apesenta na forma de duas bibliotecas e um conjunto de programas que implementam as rotinas por elas disponibilizadas. Os mecanismos do SSL estão implementadas na libssl, e os outros algoritmos estão implmentados na libcrypto.

Neste trabalho a biblioteca será utilizada principalmente para fazer o *parsing* dos certificados de chave pública e obter as chaves públicas dos mesmos. Muito já foi falado sobre a falta de documentação e falhas de segurança dessa biblioteca, mas como é uma das poucas opções com licença livre disponíveis ela foi escolhida neste trabalho.

Capítulo 3

Metodologia

3.1 Metodologia para a coleta de certificados

A coleta de certificados é essencial para o sucesso do trabalho, dado que com uma amostra de certificados maior se pode ter resultados mais representativos sobre a certificação da ICP-Brasil. Como uma primeira abordagem para coleta de certificados utilizou-se um web crawler, para coletar certificados digitais presentes em sites web.

Web crawler, em português rastreador web, é um programa que navega pela internet de uma forma metódica e automatizada analizando o conteúdo de páginas Web para obter dados de interesse ao usuário.

O 'web crawler' HTTrack, disponível em https://www.httrack.com/, foi utilizado neste trabalho. O aplicativo começa com uma lista de URLs para visitar, chamadas sementes. Quando o rastreador visita estes URLs identifica todos os 'hiperlinks' na página e adiciona-os à lista de URLs para visitar, chamada de fronteira de rastreamento. URLs da fronteira são recursivamente visitados de acordo com um conjunto de políticas. Se o 'crawler' está realizando arquivamento de dados de sites ele copia e salva os dados requeridos. Os arquivos geralmente são armazenados de tal forma que eles possam ser vistos como eram no site original, de acordo com [36] e [13].

O web crawler precisa priorizar o que vai ser baixado da web, dado o conteúdo enorme encontrado na web. Dessa forma, deve definir politicas para visitação de sites, para evitar também duplicação de conteúdo.

No inicio do trabalho foi utilizado o webcrawler HTTrack. Ao decorrer da utilização desta ferramente houve diversos problemas: dificuldade com a ferramenta e inexperiência com web crawling em geral. Assim, não foi obtido um número satisfatório de certificados ICP-brasil com esta abordagem, porém os certificados obtidos desta maneira foram adicionados ao banco de certificados utilizado posteriormente.

A coleta de quantidade significativa de certificados emitidos sob regime da ICP-BR acabou se tornando o calcanhar de Aquiles deste trabalho, requerendo uma abordagem adaptativa uma vez iniciada a fase de programação. Isso ocorreu porque com as primeiras metodologias utilizadas não se obteve um número satisfatório de certificados para o teste.

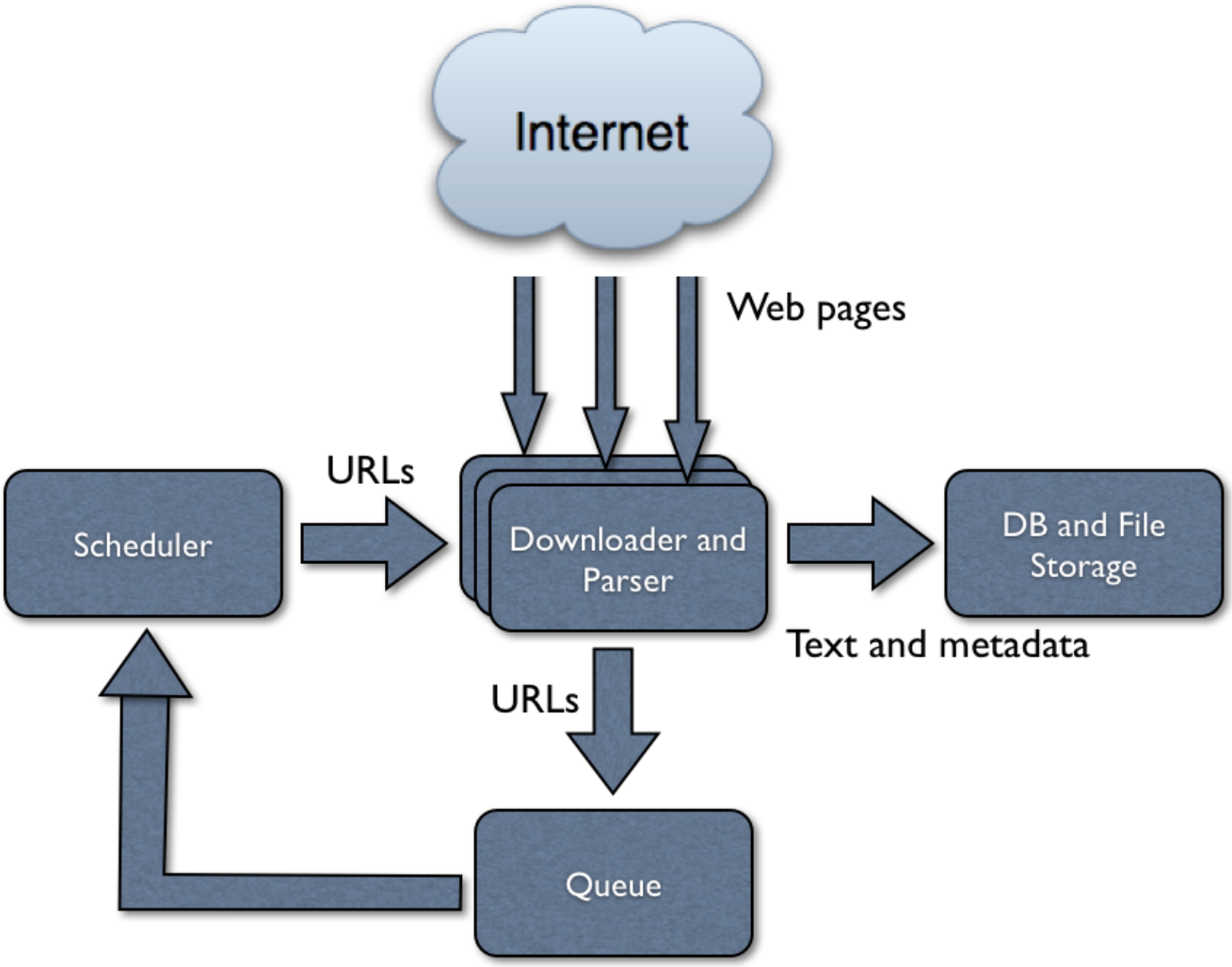

Figura 3.1: Funcionamento básico de um web crawler [3]

3.1.1 Obtenção de certificados

Os certificados utilizados neste trabalho foram obtidos do projeto EFF SSL observatory, encontrado em [25]. Este projeto fez uma grande mineração de certificados de chave pública pela WEB. Foram vasculhados todos os endereços IPV4 respondendo na porta 443, de forma a obter os certificados fornecidos pelos domínios de acordo com o protocolo TLS ou SSL.

Como resultado, o projeto encontrou 11.3 milhões de servidores aceitando o *handshake* SSL. Com isso, a EFF publicou um banco de dados mySQL de 12 gigabytes contendo 1.5 milhões de certificados digitais válidos de toda a WEB, a diferença com o número de 11.3 milhões citado acima vem de que grande parte das cadeias de certificados coletadas são inválidas, ou também, pelo fato de que muitos desses servidores utilizam mesmos certificados que outros servidores analizados. Para realização do projeto utilizamos este banco de certificados provido pela EFF.

Porém, como a quantidade de dados é muito grande no arquivo da EFF (para a complexidade quadrática do programa utilizado para analizar todos os pares de módulos), utilizou-se uma abordagem de separar cada certificado em um arquivo, de modo a facilitar que testes sobre amostras específicas fossem realizados. Além disso, criou-se diversos 'scripts' para acessar os certificados e extrair dados relevantes.

No projeto da EFF divulgou-se diversas estatísticas sobre os certificados encontrados na WEB, mostrando erro de módulos comuns, primo em comum, expoente pequeno e chaves fracas em geral. Sobre chaves fracas, antes do trabalho publicado por Lenstra, já havia sido descoberto um problema com o OpenSSL que resultava em geração de chaves facilmente previsíveis. Porém, neste trabalho, foi feita uma análise de chaves RSA mais específica, de modo a analizar apenas os certificados que foram emitidos sob regime da ICP-brasil, para colher elementos sobre o processo de certificação que está ocorrendo no Brasil.

3.1.2 Extração de certificados da ICP-Brasil

O banco de certificados da EFF continha certificados utilizados no mundo todo para HTTPS e SSH, mas era preciso reconhecer quais deles eram certificados emitidos sob regime da ICP-Brasil. Para isso foi usado o comando egrep sobre o repositório EFF, de forma a separar, apenas, certificados emitidos por certificadoras homologadas pela ICP-brasil. As certificadoras que são de interesse para o trabalho serão mostradas em tabela à frente. E a seguir está o script utilizado para extrair, da amostra de certificados da EFF, os certificados emitidos sob regime da ICP-Brasil.

```
egrep -i -l -w -r 'certisign|digitalsign|advogados|soluti|
boavista|fenacon|fenacor|imesp|icpbrasil|notarial|petrobras|
prodemge|proderj|prodest|serasa|serpro|sincor|acboavista|
serjus|certdigital|fercomercio|acertcon|doccloud|jucmg' ./
>> ../certsBrasil2
```

Tabela 3.1: Lista de certificadoras homologadas pela ICP-brasil

Autoridade certificadora	**Endereço web**
AC BOA VISTA	http://icpbrasil.vpki.certificadob oavista.com.br/acboavista
AC CAIXA	http://icp.caixa.gov.br/ asp/repositorio.asp
AC CERTISIGN	http://icpbrasil.certisign.com.br/ repositorio/AC_Certisign.htm
AC DIGITALSIGN ACP	http://www.digitalsignc ertificadora.com.br/rep ositorio/acp/
AC FENACON CERTISIGN SRF	http://icp-,brasil.acfenacon.com.b r/repositorio/index.htm
AC FENACOR	http://serasa.certificad odigital.com.br/ajuda/r epositorio/acsvinculadas-a-icpbrasil/ac-fenacor/
AC INSTITUTO FENACON	http://icp-,brasil.acfenacon.com.b r/repositorio/ACInstitutoFenacon/index.htm
AC NOTARIAL SRF	http://www.acnotarial.c om.br/repositorio.php
AC OAB	http://icpbrasil.certisign.com.br/ repositorio/AC_OAB.ht m
AC PETROBRAS	http://icpbrasil.certisign.com.br/ repositorio/acpetrobras/index.htm
AC PRODEMGE	http://icpbrasil.certisign.com.br/ repositorio/AC_PRODE MGE.htm
AC PRODERJ	https://ccd.serpro.gov. br/acproderj
SERASA ACP	http://serasa.certificadodig ital.com.br/ajuda/repositor io//
AC SERPRO	https://ccd.serpro.gov. br/acserpro/
AC SINCOR	http://icpbrasil.acsincor.com.br/rep ositorio/
AC SOLUTI	https://ccd.acsoluti.com.br /ac-soluti.html

3.2 Metodologia dos testes

Os testes de MDC entre módulos RSA foram feitos de forma similar ao aprensentado no artigo do Lenstra [28]. Ao longo do trabalho verificou-se que em [24] foi publicado o código fonte utilizado para os testes de MDC, de qualquer forma, foi feito um programa para estes testes neste trabalho. Possibilitando a adaptação para as especificidades deste trabalho.

Para os propósitos deste trabalho foi preciso manipular números muito grandes, tal qual o módulo de chaves públicas, que tem entre 512 até 4095 bits de tamanho. Para isso é necessário uso de aritmética extendida nos cálculos com estes dados [23]. Para isto utilizou-se a biblioteca Miracl da Certivox, que será apresentada nas próximas páginas.

Além disso, o programa deve fazer o parsing de certificados públicos, para obter os dados de interesse. Deve-se ter em vista que esta não é uma tarefa muito simples, já que existem inúmeros formatos de arquivos que codificam certificados X.509: PEM, cer, crt, DER, PKS, key e outros [19]. Para isso utilizou-se uma ferramenta já consolidada para 'parsing' e conversão de certificados, que faz parte do pacote OpenSSL.

O programa principal desenvolvido neste trabalho se chama certTest e é um programa de linha de comando implementado em C. Ele recebe como input um diretório, onde se supõe que devem haver certificados de chave pública, caso contrário o programa acusará erro de formato e não processará os arquivos. A partir daí o programa trabalhará com um grafo implícito, onde os vértices são os módulos das chaves públicas RSA, e as arestas serão o MDC entre eles.

Depois disso serão identificadas as arestas valoradas com um número diferente de 1, indicando colisão de primos, um deles ou ambos. No caso de todos os primos que compõem os módulos serem iguais nota-se um caso de módulos iguais em chaves públicas, o que pode ser uma falha pois, coincidindo também os expoentes (caso comum, dada a utilização de um expoente padrão na maioria das bibliotecas de geração de chaves RSA), qualquer dado cifrado por qualquer das chaves será legível pelos múltiplos donos das chaves privadas associadas às chaves públicas. Examinaremos as vulnerabilidades associadas à colisão de módulos no capítulo 4, com ênfase nas implicações sobre esquemas de assinatura digital.

Na outra hipótese em que apenas um dos primos é repetido em dois módulos, este é o caso que merece mais atenção, pois inviabiliza o uso eficaz das respectivas chaves. Isso acontece pois qualquer pessoa que tenha acesso a estas chaves poderá aplicar um algoritmo de MDC e descobrir o primo utilizado em comum. A partir daí descobre-se o outro primo utilizado no módulo por uma simples divisão, e consequentemente, possibilita o cálculo das respectivas chaves privadas. Isto torna as chaves ineficazes (quebradas) sem qualquer sinal externo de problemas para os agentes interessados no uso das chaves.

Ilustrando: se temos duas chaves públicas com módulos A e B na forma:

$$A = I * J$$

$$B = J * K$$

Tal que A e B são módulos RSA e I,J e K são primos grandes.

Aplicando o algoritmo de MDC extendido em A e B descobriremos a repetição do primo J. Despois disso faremos:

$$I := A/J$$

$$K := B/J$$

Assim fatoramos os dois módulos, o que torna as chaves A e B ambas inseguras, pois podemos facilmente calcular o inverso multiplicativo do expoente contido na chave pública 'e' no módulo adequado, dessa forma, obtem-se a chave privada pelo algoritmo de euclides extendido, usado também para obter o expoente privado durante a geração do par de chaves:

$$e^{-1} \equiv d \mod (I-1)*(J-1)$$

Assim chegamos à chave privada relativa ao módulo A, descrita por (A,d). De forma análoga é possível também calcular a chave privada relativa ao módulo B.

3.2.1 A biblioteca Miracl

A Miracl é uma biblioteca criptografica com foco em implementação eficiente de aritmética extendida, disponível em https://www.miracl.com/index. A biblioteca tem funções básicas de aritmética de números grandes como soma, produto, divisão e subtração, mas também implementa procedimentos especificos de protocolos criptográficos. Como a exponenciação modular, máximo divisor comum, computação do inverso multiplicativo modular e input e output de números grandes.

A miracl é uma biblioteca em C mantida pela Certivox, e tem como principais objetivos ser eficiente, simples e portável. Para que a biblioteca seja de fato eficiente ela tem diversas funções criticas implementadas em baixo nível (assembly). Também é possível observar que a Miracl utiliza os mais eficientes algoritmos conhecidos na literatura.

Neste trabalho, o uso da biblioteca Miracl será, principalmente, para trabalhar com aritmética de números grandes, utilizando funções de divisão, MDC e comparação. Destaque à função xgcd, que recebe dois números no formato big e retorna o MDC entre eles e a combinação linear dos números da equação do MDC:

int xgcd (big x, big y, big xd, big yd, big z)

3.2.2 Implementação

Para implementação dos programas necessários foi utilizado linguaguem C e C++. A plataforma utilizada tanto para programação quanto para testes foi um laptop quad core i7 com 1T de disco rígido, 8G de memória RAM, arquitetura 64 bits e sistema operacional ubuntu 15.10. Não foi utilizada programação multi-thread.

Para realizar as tarefas necessárias neste trabalho foram utilizados 3 programas principais: certTest, que realiza os teste de MDC entre os pares de módulos; pegarModulos, que recebe um certificado e retorna o módulo RSA contido no mesmo e o auxiliar, que lê o arquivo repositório da EFF e extrai todos os certificados para arquivos separados.

O certTest é um programa de linha de comando que deve receber 2 argumentos: o nome do diretório que contém módulos RSA a serem testados (os módulos devem estar em codificação hexadecimal e estarem em 1 arquivo por módulo) e o nome do arquivo de

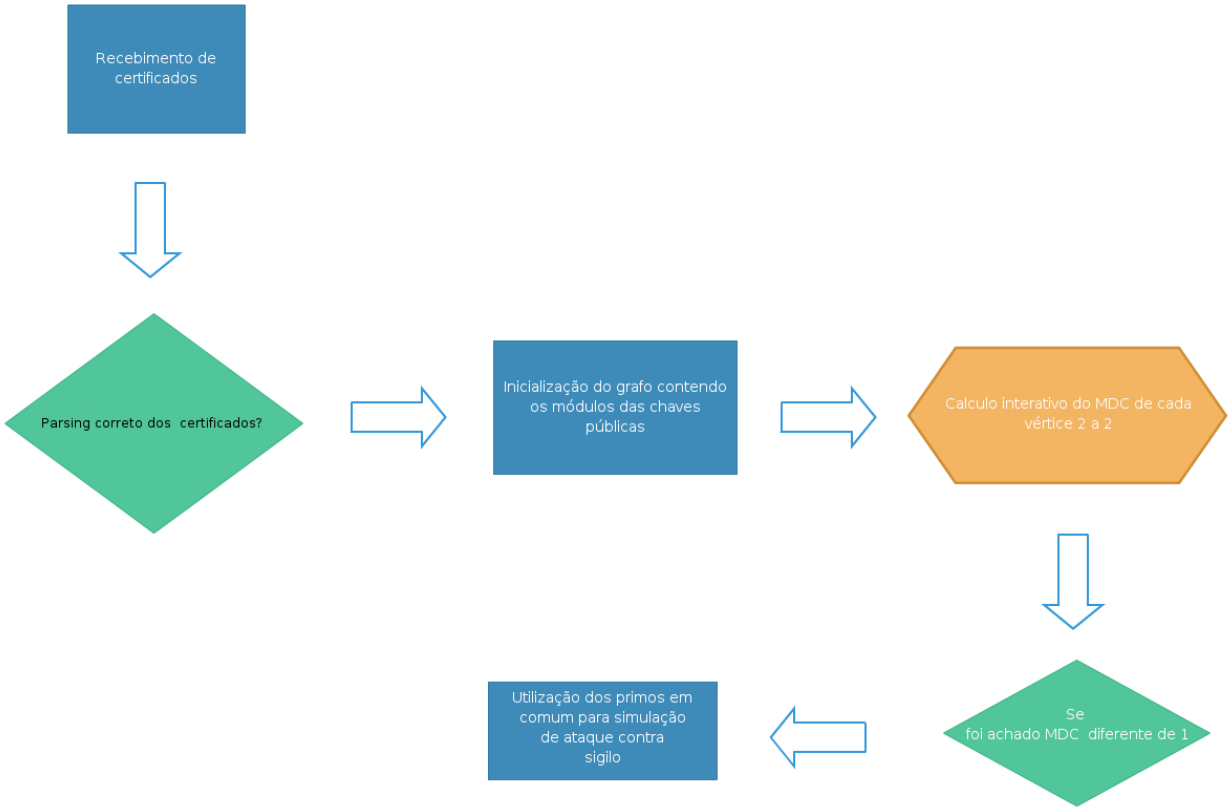

Figura 3.2: Fluxograma indicando funcionamento básico do programa certTest para teste de robustez de chaves públicas

output a ser gerado, que mostra quais módulos tiveram colisão de primos e de que tipo foi esta colisão.

O pegarModulos recebe um certificado e simplesmente extrai o módulo RSA do mesmo e coloca em um novo arquivo. O auxiliar recebe como argumento o arquivo repositório da EFF e extrai os certificados para um diretório especificado.

3.3 Dificuldades encontradas

Este trabalho se divide basicamente em duas partes. A primeira parte é a de obter um banco de certificados emitidos sob regime da ICP-brasil de tamanho satisfatório, ou seja, mais que 4000 certificados, já que este seria um tamanho adequado para que fosse possível encontrar colisões em módulos RSA com probabilidade alta, assumindo-se que as taxas de colisão de 2 a cada 1000 módulos encontradas nos trabalhos de referência se repetissem neste contexto. Essa parte do trabalho demandou muito esforço e, apesar de tudo, só foram obtidos 1153 certificados emitidos sob regime ICP-brasil. Então os resultados encontrados não tem a representatividade estatística esperada, mas mesmo assim, é possivel inferir diversas conclusões sobre os certificados obtidos.

Primeiramente a abordagem para obter os certificados utilizada foi de procurar algum banco de certificados fornecido por alguma entidade certificadora. Não houve sucesso nisso, dado que as certificadoras não publicam ou possivelmente não mantém guardado

os certificados que emitem. Porém, se confirmado que as certificadoras não guardam os certificados emitidos isto iria contra as normas descritas na MP2200 sobre cadastro de certificados.

Depois uma abordagem de web crawler foi utilizada, com o 'web crawler' HTTrack. Essa abordagem foi interessante pois o aplicativo tem uma interface bem simples e com algumas horas de processamento conseguimos 100 certificados da ICP-brasil. Porém, o programa tem algumas limitações quanto a flexibiliade do 'web crawling' e o resultado foi menor do que necessitávamos.

Numa outra tentativa, pesquisou-se a abordagem utilizada em [24], utilizando NMAP para mapear os endereços IP que respondiam requisição na porta 443 e proviam certificados para conexão TLS. Essa abordagem, que foi a mesma utilizada pelo EFF, é boa para achar certificados utilizados na web, porém, não faria sentido fazer este mesmo processo sabendo que a EFF disponibilizou todo banco de certificados obtidos dessa maneira.

Posteriormente, descobriu-se o banco de certificados obtidos pela EFF. Banco que foi obtido pela organização EFF utilizando abordagem explicada acima e requisição de conexão TLS e posterior armazenamento do certificado recebido. Este banco de certificados tem 1.3 milhões de certificados distintos encontrados na web, resultando em um banco MYSQL de 17GB. Primeiramente, tentou-se abrir o arquivo em um gerenciador de bancos de dados MYSQL, esta tentativa falhou, por algum motivo o computador sempre travava ao tentar abrir o arquivo, que provavelmente é grande demais para a plataforma utilizada.

Depois utilizou-se programas auxiliares para extrair cada certificado e colocar cada certificado em um arquivo, de forma a facilitar consultas posteriores. Porém houve um problema nessa parte. Como a quantidade de arquivos era muito grade os i-nodes do sistema de arquivos acabaram. Para superar este problema, os arquivos foram analisados em diversos lotes, um por um. Depois de retirado um lote de certificados criamos um script para extrair os módulos RSA de cada um dos certificados, que é o que necessitamos para fazer o MDC e verificar possíveis primos em comum.

A partir disso foi utilizado o programa certTest para fazer MDC entre pares de módulos RSA em um dirétorio contendo certificados. Este programa tem complexidade quadrática no número de certificados. Como o banco era muito extenso foram necessários diversos dias rodando o programa para finalizar. Para obter amostras comparativas fiz o mesmo teste para certificados genéricos do banco da EFF, certificados utilizados no Brasil e certificados emitidos sob regime da ICP-Brasil. E os resultados serão mostrados à frente.

Capítulo 4

Resultados

4.1 Testes realizados

O foco deste trabalho foram os testes realizados sobre a massa de certificados obtida. Dessa forma diversos testes foram realizados ao longo do trabalho, sendo realizados em conjuntos de certificados diversos utilizando algumas abordagens distintas. Todos os arquivos de certificados referidos neste trabalho foram obtidos do repositório EFF, e a numeração dos arquivos foi feita durante a extração dos certificados pelo programa implementado no trabalho 'auxiliar.cpp', sendo que a numeração dos arquivos foi feita de acordo com a ordem em que os certificados apareceram no repositório EFF.

Primeiro, para contextualizar o leitor, a formatação dos certificados utilizados neste trabalho é a seguinte:

```
(
  `Authority Information Access:1.3.6.1.4.4308.10.50 - URI` varchar(49) DEF
  `fetchtime` int(11) DEFAULT NULL,
  `fingerprint` char(80) NOT NULL,
  `id` varchar(6) DEFAULT NULL,
  `ip` varchar(15) DEFAULT NULL,
  `Issuer` varchar(367) DEFAULT NULL,
  `moz_valid` varchar(394) DEFAULT NULL,
  `ms_valid` varchar(394) DEFAULT NULL,
  `path` varchar(59) DEFAULT NULL,
  `RSA Public Key:Modulus` varchar(6146) DEFAULT NULL,
  `RSA_Modulus_Bits` varchar(5) DEFAULT NULL,
  `Serial Number` varchar(59) DEFAULT NULL,
  `Signature` varchar(1535) DEFAULT NULL,
  `Signature Algorithm` varchar(24) DEFAULT NULL,
  `Subject` varchar(5045) DEFAULT NULL,
  `Subject Public Key Info:DSA Public Key:G` varchar(386) DEFAULT NULL,
  `Subject Public Key Info:DSA Public Key:P` varchar(386) DEFAULT NULL,
  `Subject Public Key Info:DSA Public Key:pub` varchar(386) DEFAULT NULL,
  `Subject Public Key Info:DSA Public Key:Q` varchar(62) DEFAULT NULL,
  `Subject Public Key Info:Public Key Algorithm` varchar(14) DEFAULT NULL,
  `Subject Public Key Info:RSA Public Key:Exponent` varchar(24) DEFAULT NUI
```

```
  `transvalid` varchar(3) DEFAULT NULL,
  `valid` tinyint(1) DEFAULT NULL,
  `Validity:Not After` varchar(25) DEFAULT NULL,
  `Validity:Not Before` varchar(25) DEFAULT NULL,
  `Version` varchar(8) DEFAULT NULL,

  `X509v3 Policy Constraints:Require Explicit Policy` varchar(12) DEFAULT N
  `X509v3 Policy Mappings:2.16.840.1.101.3.2.1.1.2` varchar(228) DEFAULT NU
  `X509v3 Policy Mappings:2.16.840.1.101.3.2.1.3.3` varchar(232) DEFAULT NU
  `certid` int(11) NOT NULL DEFAULT '0',
  `startdate` datetime DEFAULT NULL,
  `enddate` datetime DEFAULT NULL,
  `Validity:Not Before datetime` datetime DEFAULT NULL,
  `Validity:Not After datetime` datetime DEFAULT NULL,
  PRIMARY KEY (`certid`),
  UNIQUE KEY `fingerprint` (`fingerprint`)
)
```

Destaco, neste trabalho, a realização de diversos testes genéricos (em certificados do mundo todo, irrestritamente) onde o teste realizado com a maior amostragem foi feita sobre 62 mil certificados e teve como resultado o relato de 2551 certificados RSA com módulos repetidos, o que indica uma quantidade de 4.1% de certificados com módulos repetidos na amostra. Para ilustrar o resultado deste teste apresento dois desses certificados que tiveram módulos iguais:

```
Filename: cert19049

1292591849
'SHA1 Fingerprint=AC:09:F7:2F:F7:A9:0D:5D:F5:98:2B:97:54:99:E8:
91:54:E6:F1:3E'
'46686'
'118.155.225.146'
' C=JP, O=SECOM Trust.net, CN=SECOM Passport for Web CA'
'Yes'
'Yes'
'/space2/scan//118.x.x.x/118.x.x.146/118.155.225.146.results'
'00:9b:13:22:db:9c:57:9f:a5:ae:b3:9b:02:0b:1c:ae:23:70:1f:58:
3d:fd:82:83:cc:63:c2:87:49:9c:37:63:7e:e0:3a:19:c3:1f:8b:37:9
d:9b:63:5c:d0:ff:47:69:fc:44:ca:0e:b5:97:26:ef:a9:07:8b:b7:da
:45:f4:b4:6a:65:7d:31:01:a8:1b:1c:89:fd:d5:15:66:18:51:92:b3:
76:18:ba:6f:bb:59:45:1a:46:c8:93:36:a2:cf:aa:f2:8b:85:ac:78:9
e:f9:d5:51:f7:f3:5b:62:7a:ec:94:5e:01:09:da:dc:03:5f:92:3b:fc
:07:f8:f2:16:18:2c:85'
'1024'
' 21463 (0x53d7)'
'0c:6f:59:24:1d:f7:e2:5c:7d:e6:c2:32:ff:75:e8:06:05:02:9f:55:
bc:e9:0b:76:02:96:52:83:79:82:1b:62:58:b1:ad:a6:27:14:48:53:c
0:73:87:e5:7e:10:a9:b0:1f:64:61:63:e2:f5:07:10:67:fa:67:91:c4
```

```
:ea:72:fc:b6:24:51:40:12:0c:43:8b:ee:1d:ae:dd:d5:5c:57:94:39:
bc:3b:de:5c:79:8c:3d:4c:33:50:f0:36:ab:4c:8c:bd:d8:4f:d6:87:f
f:8b:74:cc:d0:3c:14:48:d0:ba:2b:45:0f:b0:57:50:cc:96:2a:07:f4
:fe:ba:a7:11:b5:0c'
' sha1WithRSAEncryption'
' C=JP, ST=Aichi, L=Nagoya-shi, O=PROTO CORPORATION,
OU=System-2, CN=secure.veeschool.com'
' rsaEncryption'
' 65537 (0x10001)'
1
' Dec  2 14:59:59 2011 GMT'
' Dec  2 08:04:46 2009 GMT'
' 3 (0x2)'
'SSL Server'
'C6:1B:17:CF:2E:4E:62:FE:03:AF:BB:8D:96:CE:87:6B:42:D4:0D:CA'
' 1.2.392.200091.100.701.2:Policy: 1.2.392.200091.100.701.2:
CPS=== https://repo1.secomtrust.net/spcpp/pfw/pfwca/'
'URI===http://repo1.secomtrust.net/spcpp/pfw/pfwca/fullcrl2.crl'
'TLS Web Server Authentication'
'(critical) Digital Signature, Key Encipherment'
'07:58:AA:03:91:82:FE:4F:FC:6B:4E:86:34:E8:A3:C2:D5:25:32:77'
47668
'2009-12-02 08:04:46'
'2011-12-02 14:59:59'
'2009-12-02 08:04:46'
'2011-12-02 14:59:59'
```

```
Filename: cert21741

1292588479
'SHA1 Fingerprint=01:94:F7:DD:E0:AD:D9:8B:BC:44:D4:D5:B6:A6:52:37:
AE:63:16:A5'
'123226'
'118.155.225.226'
' C=JP, O=SECOM Trust.net, CN=SECOM Passport for Web CA'
'Yes'
'Yes'
'/space2/scan//118.x.x.x/118.x.x.226/118.155.225.226.results'
'00:9b:13:22:db:9c:57:9f:a5:ae:b3:9b:02:0b:1c:ae:23:70:1f:58:
3d:fd:82:83:cc:63:c2:87:49:9c:37:63:7e:e0:3a:19:c3:1f:8b:37:9
d:9b:63:5c:d0:ff:47:69:fc:44:ca:0e:b5:97:26:ef:a9:07:8b:b7:da
:45:f4:b4:6a:65:7d:31:01:a8:1b:1c:89:fd:d5:15:66:18:51:92:b3:
76:18:ba:6f:bb:59:45:1a:46:c8:93:36:a2:cf:aa:f2:8b:85:ac:78:9
e:f9:d5:51:f7:f3:5b:62:7a:ec:94:5e:01:09:da:dc:03:5f:92:3b:fc
```

```
:07:f8:f2:16:18:2c:85'
'1024'
' 22876 (0x595c)'
'15:e0:a3:96:4c:69:31:07:b0:a4:40:4f:ec:f2:68:7b:dd:e0:49:d4:
78:89:a1:bd:0e:23:49:75:ab:91:6e:1a:01:32:7d:d9:1a:c7:8e:62:4
f:0e:dc:58:59:fc:a3:7e:3c:f1:f8:39:64:ff:a7:5b:25:bf:db:92:ee
:4e:ff:8b:33:8e:76:91:4c:f6:39:2c:2a:fd:b1:47:ff:e0:fd:69:99:
f1:9d:6d:29:c5:a4:93:19:5c:89:5f:c0:80:81:f3:99:bf:61:b8:f3:a
d:04:54:95:82:76:45:e0:94:26:f4:48:e0:b9:a0:91:89:02:4e:bc:67
:55:3a:c9:40:17:03'
' sha1WithRSAEncryption'
' C=JP, ST=Aichi, L=Nagoya-shi, O=PROTO CORPORATION,
OU=System-1, CN=mdev.kaigo-kyuujin.com'
' rsaEncryption'
' 65537 (0x10001)'
1
' May 12 14:59:59 2011 GMT'
' May 10 03:14:33 2010 GMT'
' 3 (0x2)'
'SSL Server'
'C6:1B:17:CF:2E:4E:62:FE:03:AF:BB:8D:96:CE:87:6B:42:D4:0D:CA'
' 1.2.392.200091.100.701.2:Policy: 1.2.392.200091.100.701.2:
CPS=== https://repo1.secomtrust.net/spcpp/pfw/pfwca/'
'URI===http://repo1.secomtrust.net/spcpp/pfw/pfwca/fullcrl2.crl'
'TLS Web Server Authentication'
'(critical) Digital Signature, Key Encipherment'
'07:58:AA:03:91:82:FE:4F:FC:6B:4E:86:34:E8:A3:C2:D5:25:32:77'
51925
'2010-05-10 03:14:33'
'2011-05-12 14:59:59'
'2010-05-10 03:14:33'
'2011-05-12 14:59:59'
```

Sobre os certificados acima, é possivel se observar que eles foram emitidos para uma mesma organização: PROTO CORPORATION. Porém seus identificadores são diferentes, mais ainda, o 'commom name' do sujeito do primeiro certificado é secure.veeschool.com e o do segundo é mdev.kaigo-kyuujin.com (totalmente diferentes). Contudo, não é possível saber se a certificadora emitiu estes dois certificados com mesmos módulos propositalmente, nem se isto pode acarretar alguma falha na utilização dos mesmos para os interessados, já que não se sabe de que forma os agentes pretendem utilizar estes certificados. Mas se supormos que há interesse em identificar unicamente um assinante de um documento digital, nesse contexto não seria possível, já que qualquer um dos sujeitos acima assinaria um documento com uma mesma chave privada.

4.2 Extendendo os testes - ICP-BR versus EFF geral

Outro teste importante realizado foi o teste envolvendo certificados emitidos no regime da ICP-Brasil (emitidos pelas certificadoras citadas na tabela 4.1) e certificados genéricos do mundo todo. No trabalho chamamos este tipo de teste de cruzamento, pois envolvia duas amostragens diferentes. O resultado deste teste, para nossa surpresa, foi que nenhum módulo de chaves públicas da ICP-Brasil teve colisão de primos em relação à amostra de certificados genéricos, nem de módulos iguais, nem de primo comum em módulos distintos. Isto, à princípio é um ótimo sinal, pois indica que os certificados emitidos no regime da ICP-brasil não apresentam o problema de baixa entropia segundo este primeiro teste.

Porém, há diversos fatores a se analizar. A amostra obtida neste trabalho foi de apenas 1153 certificados, uma amostra pequena para a realização de um teste deste tipo. Para trabalhos futuros seria uma boa idéia trabalhar com uma metodologia diferente e obter um número maior de certificados da ICP-brasil, para que os testes tenha maior representatividade do conjunto de certificados alvo. Isto é difícil porque a maioria dos certificados emitidos sob regime da ICP-brasil é utilizado de forma pessoal para assinatura digital e não para SSL, inviabilizando o método de crawling para obte-los. Dessa forma, nosso trabalho não teve acesso a estes certificados utilizados especificamente para assinatura digital, dado que o banco de certificados da EFF só tem certificados que foram obtidos à partir do fechamento de conexões HTTPS.

Outro teste realizado foi entre certificados da ICP-brasil sem cruzar com outros certificados da amostragem da EFF. Este teste foi sobre 1153 certificados emitidos sob regime da ICP-brasil e teve como resultado 1 par de certificados com módulos e expoentes iguais. O que nos traria uma estatística de 0.09% de módulos repitidos na amostra. Vale ressaltar que como a amostra foi de apenas 1153 certificados este valor estatístico não é muito representativo. Mas é um indicador importante de que, se esta colisão foi encontrada num espaço tão pequeno, espera-se que mais dessas colisões sejam encontradas num espaço amostral maior. E, como visto na sessão 1.4, certificados com chaves públicas iguais (módulo e expoente) representam um grande perigo para protocolos de assinatura digital, pois inviabilizam a identificação do agente que assinou algum documento digital unicamente. Além disso, quando estamos falando de protocolos para sigilo usando chaves RSA, módulos repetidos também são uma ameaça ao protocolo, dado que não há sigilo na comunicação entre os agentes que possuirem as chaves RSA com módulos repetidos, como veremos adiante.

Resumindo, este resultado é um sinal de alerta para a ICP-brasil, dado que se tivermos mais ocorrências de módulos iguais neste contexto teríamos os protocolos de assinatura digital fragilizados neste regime, comprometendo também o regime jurídico de inversão do ônus da prova. Segue abaixo o par de certificados emitidos sob regime da ICP-Brasil com módulos iguais e expoentes iguais:

```
filename: 1_cert252438

1292703172
'SHA1 Fingerprint=6D:0D:2F:4C:56:5D:01:54:89:CD:EA:E4:3C:DC:D7:
79:72:52:3A:E8'
'1320'
'187.58.136.143'
```

```
' C=BR, O=ICP-Brasil, OU=Autoridade Certificadora da Justica-
AC-JUS, CN=AC CAIXA-JUS'
'self-signed: in certificate chain '
'Yes'
'/space2/scan//187.x.x.x/187.x.x.143/187.58.136.143.results'
'00:c5:b5:9f:3c:45:7a:19:2b:14:4c:72:9f:f9:fb:1d:7b:35:
a9:f5:9b:31:4d:17:9f:6a:3b:ea:81:44:cd:b2:1b:43:df:44:f
4:0a:0a:51:a2:26:85:7a:1f:c2:e3:e0:83:44:23:0e:4b:ca:73
:aa:0a:9a:a8:5f:87:8a:a3:51:a8:e2:25:94:6c:77:ba:66:95:
a2:c6:9f:3d:20:fd:f6:ae:bf:94:ec:d7:a9:58:a4:d6:21:75:f
7:7a:60:5a:94:f1:81:0e:6c:36:d5:a1:ee:59:d9:eb:8e:db:d0
:5a:ea:00:73:d0:88:32:75:cf:29:83:50:94:18:af:97:2e:b7:
df'
'1024'
' 1144424788 (0x44368954)'
'42:7b:79:75:7c:1f:96:e0:c2:ea:21:da:13:2a:98:20:84:a9:d5:2a:
73:d2:13:dc:87:27:9f:a8:2b:29:87:02:67:c3:d5:35:ca:2d:d4:54:8
e:28:4c:8d:54:ab:b7:15:9f:5f:6c:33:45:4a:73:4d:69:15:bf:86:8a
:45:3e:25:36:8e:d3:a4:07:23:7e:56:66:a0:96:6e:5a:1e:b0:dc:22:
2e:16:60:7e:2f:ef:28:39:94:80:56:f2:56:5d:18:db:29:54:88:c3:e
6:00:20:91:f8:74:6b:f3:61:3e:0d:83:b2:c8:07:75:84:d3:1d:97:24
:13:b2:41:44:43:14:aa:a6:88:97:37:89:89:80:3b:a7:1a:f7:ab:28:
93:ba:ce:76:87:40:72:42:21:26:c4:ae:03:ea:35:a4:66:c9:fb:4a:3
c:60:5d:09:a5:f1:9b:ae:1d:7a:85:2f:4b:49:f7:11:3b:2f:b7:a8:87
:b4:2e:24:d8:f6:22:3f:27:77:94:42:8b:26:b0:28:0c:c5:b1:7a:f2:
a3:ca:62:d1:cb:d3:22:07:74:ed:58:00:83:85:7e:8b:ca:79:4b:2b:e
6:c7:ee:c8:e8:e7:36:0c:75:13:c0:73:c1:0e:73:d9:8b:8c:11:f9:22
:92:d2:25:7b:e3:3d:e1:16:fa:8a:3f:3c'
' sha1WithRSAEncryption'
' C=BR, O=ICP-Brasil, OU=Autoridade Certificadora da Justica
- AC-JUS, OU=Cert-JUS Equipamento Servidor - A1, OU=TRT4,
OU=SECRETARIA DE TECNOLOGIA DA INFORMACAO, CN=extranet.trt4.jus.br'
' rsaEncryption'
' 65537 (0x10001)'
1
' May 31 15:53:36 2011 GMT'
' Aug 10 14:33:17 2010 GMT'
' 3 (0x2)'
'Unknown==0 ANDALSO Unknown==..V7.1....'
'0C:D0:D7:FE:03:05:9B:BD:21:89:04:3F:12:C6:AF:A3:AA:D7:9E:2E'
'FALSE'
' 2.16.76.1.2.1.22:Policy: 2.16.76.1.2.1.22:CPS==== http://icp.caixa.gov.br/
'URI====http://icp.caixa.gov.br/repositorio/accaixajus.crl ANDALSO URI====htt
'TLS Web Server Authentication, TLS Web Client Authentication'
'(critical) Digital Signature, Non Repudiation, Key Encipherment'
'email====alberto.muller@trt4.jus.br, othername:<unsupported>, othername:<un
```

598157
'2010-08-10 14:33:17'
'2011-05-31 15:53:36'
'2010-08-10 14:33:17'
'2011-05-31 15:53:36'

Filename: 1_cert252414

1292674960
'SHA1 Fingerprint=85:06:D8:A4:72:04:44:52:00:14:E6:FE:9C:6A:27:
9E:DE:E9:84:1D'
'1110'
'187.58.136.133'
' C=BR, O=ICP-Brasil, OU=Autoridade Certificadora da Justica -
AC-JUS, CN=AC CAIXA-JUS'
'self-signed: in certificate chain '
'Yes'
'/space2/scan//187.x.x.x/187.x.x.133/187.58.136.133.results'
'00:c5:b5:9f:3c:45:7a:19:2b:14:4c:72:9f:f9:fb:1d:7b:35:a9:f5:
9b:31:4d:17:9f:6a:3b:ea:81:44:cd:b2:1b:43:df:44:f4:0a:0a:51:a
2:26:85:7a:1f:c2:e3:e0:83:44:23:0e:4b:ca:73:aa:0a:9a:a8:5f:87
:8a:a3:51:a8:e2:25:94:6c:77:ba:66:95:a2:c6:9f:3d:20:fd:f6:ae:
bf:94:ec:d7:a9:58:a4:d6:21:75:f7:7a:60:5a:94:f1:81:0e:6c:36:d
5:a1:ee:59:d9:eb:8e:db:d0:5a:ea:00:73:d0:88:32:75:cf:29:83:50
:94:18:af:97:2e:b7:df'
'1024'
' 1144424686 (0x443688ee)'
'08:c0:ec:26:bd:ac:43:88:c6:fa:18:79:5e:3f:cb:45:36:c8:1c:33:
a7:40:ac:49:aa:76:99:50:40:f8:19:be:76:ea:38:5f:88:e3:b3:2e:b
1:48:8e:c5:aa:52:96:e6:76:ef:0c:59:03:d7:ac:80:d5:b5:1a:47:5e
:97:b6:a9:08:2a:3d:9c:83:e0:eb:cb:0f:86:30:77:eb:01:dc:c6:71:
f1:c8:4f:71:c6:62:20:f7:28:31:99:29:db:b1:33:d8:07:c9:88:e3:0
7:25:9b:ce:2d:d7:87:be:43:27:2b:9e:f7:cf:a4:ba:4c:04:8a:9a:f0
:ad:81:31:cc:f7:28:70:69:73:fc:ad:58:9d:a9:65:ae:e7:e3:6c:8f:
2c:71:99:16:40:bc:32:46:94:ef:0f:bd:32:94:96:0d:b2:f8:cb:d4:0
2:a6:30:06:e3:5a:f3:5b:3a:1b:1c:5b:b9:5c:9b:0d:1a:b3:7f:04:01
:2f:7e:c2:c9:b2:2c:3c:6e:cc:73:e0:c6:cd:53:32:41:a1:cc:6f:4d:
18:c8:19:e4:ad:7e:32:15:a0:bf:a9:66:68:9a:9c:1c:95:5a:8b:e4:7
6:17:1e:5f:e1:39:82:16:bb:83:bf:1e:29:31:fa:96:88:c9:39:45:05
:7e:4f:d1:3f:c3:ba:a6:3b:06:c9:86:ff'
' sha1WithRSAEncryption'
' C=BR, O=ICP-Brasil, OU=Autoridade Certificadora da Justica -
AC-JUS, OU=Cert-JUS Equipamento Servidor - A1, OU=TRT4,
OU=SECRETARIA DE TECNOLOGIA DA INFORMACAO, CN=www.trt4.jus.br'

```
' rsaEncryption'
' 65537 (0x10001)'
1
' May 31 15:53:36 2011 GMT'
' Aug  9 16:24:16 2010 GMT'
' 3 (0x2)'
'Unknown==0 ANDALSO Unknown==..V7.1.... '
'0C:D0:D7:FE:03:05:9B:BD:21:89:04:3F:12:C6:AF:A3:AA:D7:9E:2E'
'FALSE'
' 2.16.76.1.2.1.22:Policy: 2.16.76.1.2.1.22:CPS=== http://icp.caixa.gov.br/
'URI===http://icp.caixa.gov.br/repositorio/accaixajus.crl ANDALSO URI===htt
'TLS Web Server Authentication, TLS Web Client Authentication'
'(critical) Digital Signature, Non Repudiation, Key Encipherment'
'email===alberto.muller@trt4.jus.br, othername:<unsupported>, othername:<ur
598069
'2010-08-09 16:24:16'
'2011-05-31 15:53:36'
'2010-08-09 16:24:16'
'2011-05-31 15:53:36'
```

Sobre os certificados mostrados acima, é possível notar que os expoentes de suas chaves públicas são iguais: 65537 e os módulos também são os mesmos, como visto acima. Isto indica que suas respectivas chaves privadas são idênticas. Além do mais seus identificadores são diferentes, dessa forma não são o mesmo certificado utilizado duas vezes, são certificados que poderiam ter sido emitidos para entidades diferentes.

Ainda sobre implicações da ocorrência de módulos comuns em chaves RSA distintas, este fato pode dar brecha para ataques em módulo comum, referenciado em [15], onde são atribuidos chaves com mesmo módulo para diferentes usuários interessados em comunicação sigilosa. Qualquer mensagem cifrada para mais de um usuário como módulo comum pode ser facilmente vazada neste contexto, se os respectivos expoentes públicos forem primos entre si.

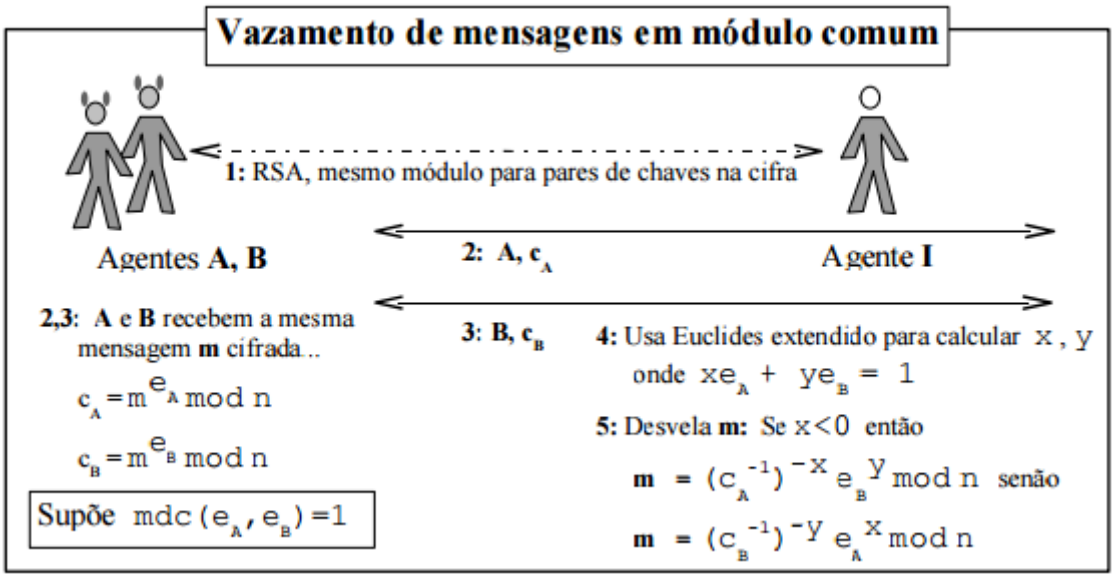

Figura 4.1: Esquema de ataque sobre RSA em módulos comuns [15]

Capítulo 5

Conclusões

5.1 Considerações finais

Seguindo a abordagem utilizada, foram encontrados 1153 certificados digitais emitidos sob regime da ICP-brasil num universo de 1.3 milhões de certificados do mundo todo. Numa primeira abordagem foi feito o MDC entre todos os pares desses 1153 certificados, que totaliza 664128 pares de certificados digitais, nos quais foram aplicados algoritmo de MDC extendido. Esse teste sobre 664128 pares demora aproximadamente cinco minutos em um laptop comum. Essa abordagem explicitou que entre esses certificados exitem dois certificados diferentes com um mesmo módulo, o que significa uma falha potencial, dado que não há sigilo entre as duas entidades que emitiram certificados com mesmo módulo, ou pior ainda, não há como associar um agente com o teor de um documento assinado com a chave privada associada a estas chaves públicas, impossibilitando a responsabilização para qualquer conteúdo contido no documento.

Os dados tem pouco valor estatístico, dado que a amostra foi pequena. Contudo, se este padrão de uma colisão de módulos a cada 1153 certificados emitidos se repetir em situações adversas, estaríamos frente a um problema para o esquema de assinatura digital no contexto da ICP-Brasil, dado que a legislação disciplinada na MP-2200 indica que um agente pode ser responsabilizado judicialmente pelo conteúdo de um documento digital assinado por ele no Brasil com inversão do ônus da prova. Com a hipótese de se encontrar de fato 1 colisão a cada 1153 certificados satisfeita, poderia-se incorrer em diversos casos de falhas judiciais, tendo em vista que, por exemplo, se um agente malicioso 'A' assinar um documento se identificando como um outro agente 'B', tendo que os módulos das chaves utilizadas por A e B sejam iguais, então suas repectivas chaves privadas também são iguais (dado que os expoentes sejam iguais também, caso comum). Isso possibilitaria que um terceiro verificador acredite que o documento, na verdade assinado por A, tenha sido assinado por B, pois a chave pública fornecida por B verifica o conteúdo assinado por A, incorrendo em responsabilização equivocada do agente B. Esta situação possibilita diversos tipos de golpes.

5.2 Trabalhos futuros

Como trabalho futuro seria indicado um trabalho visando obter uma maior amostra de certificados emitidos sob regime da ICP-Brasil. Temos como hipótese que não encontramos um número maior de certificados pois a maior parte dos certificados emitidos sob regime da ICP-Brasil são utilizados para assinatura de documentos de valor jurídico por advogados ou pessoas ligadas ao governo federal. E o trabalho realizado só teve alcance a certificados emitidos para uso em protocolo TLS.

Dessa forma, se algum trabalho futuro conseguir uma amostra maior de certificados da ICP-Brasil com uma outra abordagem que consiga certificados utilizados para assinatura digital, poderia-se fazer testes com um maior valor estatístico, fornecendo uma visão melhor sobre a colisão de primos em módulos RSA e suas possíveis implicações no contexto da ICP-Brasil.

Outro trabalho futuro interessante seria melhorar o algoritmo aqui utilizado, que tem complexidade quadrática no número de módulos RSA recebido pelo programa. Dessa forma, um teste com 3.6 bilhões de pares de módulos RSA demorou aproximadamente 24 horas. Isto impediu a execução completa de alguns testes maiores.

Referências

[1] https://developer.apple.com/. vii, 7

[2] http://www.csd.uwo.ca/ moreno//cs874. vii, 10

[3] http://www.richwan.com. vii, 22

[4] technet.microsoft.com. vii, 14

[5] www.actel.com. vii, 9

[6] www.cisco.com/c/en/us/about/press/internet-protocol-journal/. vii, 19

[7] www.networkworld.com/article/2268575/lan-wan/. vii, 8

[8] www.tech-faq.com. vii, 12, 15

[9] www.tech-faq.com. vii, 18

[10] Ashraf Abusharekh and Kris Gaj. Comparative analysis of software libraries for public key cryptography. *Software Performance Enhancement for Encryption and Decryption, SPEED*, pages 11–12, 2007. 20

[11] Carlisle Adams, Stephen Farrell, Tomi Kause, and Tero Mononen. Internet x. 509 public key infrastructure certificate management protocol (cmp). *Request for Comments (RFC)*, 4210, 2005. 13

[12] Luiz Emílio Allem and Vilmar Trevisan. O teorema dos números primos. *Salão de Iniciação Científica (14.: 2002: Porto Alegre). Livro de resumos. Porto Alegre: UFRGS, 2002.*, 2002. 8

[13] Carlos Castillo. Effective web crawling. In *ACM SIGIR Forum*, volume 39, pages 55–56. ACM, 2005. 21

[14] Santosh Chokhani and Warwick Ford. Internet x. 509 public key infrastructure certificate policy and certification practices framework. 1999. 13

[15] Pedro Antônio Dourado de Rezende. *Algoritmos criptográficos de chave pública*, volume 1. vii, 36, 37

[16] Whitfield Diffie and Martin E Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976. 1

[17] Fred Dretske. Knowledge and the flow of information. 1981. 10

[18] Carl Ellison and Bruce Schneier. Ten risks of pki: What you're not being told about public key infrastructure. *Comput Secur J*, 16(1):1–7, 2000. 14

[19] Warwick Ford. Advances in public-key certificate standards. *ACM SIGSAC Review*, 13(3):9–15, 1995. 25

[20] Behrouz A Forouzan. *Cryptography & Network Security*. McGraw-Hill, Inc., 2007. 13

[21] João Agnaldo Donizeti Gandini, Diana Paola da Silva Salomão, and Cristiane Jacob. A segurança dos documentos digitais. *Disponivel em< http://www. jus. com. br>. Acesso em: Agosto*, 2001. 11

[22] Martin Georgiev, Subodh Iyengar, Suman Jana, Rishita Anubhai, Dan Boneh, and Vitaly Shmatikov. The most dangerous code in the world: validating ssl certificates in non-browser software. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 38–49. ACM, 2012. 20

[23] Stef Graillat, Fabienne Jézéquel, Shiyue Wang, and Yuxiang Zhu. Stochastic arithmetic in multiprecision. *Mathematics in Computer Science*, 5(4):359–375, 2011. 25

[24] Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J Alex Halderman. Mining your ps and qs: Detection of widespread weak keys in network devices. In *USENIX Security Symposium*, pages 205–220, 2012. 2, 3, 4, 25, 28

[25] https://www.eff.org/observatory. The eff ssl observatory. 23

[26] Russell Impagliazzo, Ronen Shaltiel, and Avi Wigderson. Extractors and pseudo-random generators with optimal seed length. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 1–10. ACM, 2000. 2

[27] Marcos Aurelio Pchek Laureano. Gestão de segurança da informação. 2005. 1

[28] Arjen Lenstra, James P Hughes, Maxime Augier, Joppe Willem Bos, Thorsten Kleinjung, and Christophe Wachter. Ron was wrong, whit is right. Technical report, IACR, 2012. 2, 3, 11, 25

[29] Mark RV Murray. An implementation of the yarrow prng for freebsd. In *BSDCon*, pages 47–53, 2002. 2

[30] José Maurício S. Pinheiro. Tópicos avançados ii - padrão x.509. 16

[31] Eric Rescorla. *SSL and TLS: designing and building secure systems*, volume 1. Addison-Wesley Reading, 2001. 17

[32] Ronald L Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. 7

[33] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001. 10

[34] Gustavus J Simmons. Symmetric and asymmetric encryption. *ACM Computing Surveys (CSUR)*, 11(4):305–330, 1979. 6

[35] David Solo, Russell Housley, and Warwick Ford. Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile. 2002. 13

[36] Mike Thelwall. A web crawler design for data mining. *Journal of Information Science*, 27(5):319–325, 2001. 21

Anexo I

Programa para coletar certificados do arquivo da EFF

```
//auxiliar.cpp

#include <cstdio>
#include <iostream>
#include <vector>
#include <fstream>
#include <string>
#include <cstdlib>

#define DIR certificados/
#define NUMCERT 400000

using namespace std;

int main(int argc,char** argv){
        ifstream inCerts;
        ofstream output;

        if(argc != 4){
                printf("arg errados\n");
                return 1;

        }

        inCerts.open(argv[1]);

        if(!inCerts.is_open()){
                printf("arquivo indisponivel\n");
                return 1;

        }
```

```
int achou = 0;
if(atoi(argv[3]) == 0){
    inCerts.seekg(1200,inCerts.beg);

    while(!achou){
        if(inCerts.get() == 'V' && inCerts.get() == 'A' && inCerts.
                achou = 1;
        }
    }
}else{
    inCerts.seekg(atoi(argv[3]),inCerts.beg);
}


int numCert = 1;
char aux;
int count = 0;
int escaped;
int sair;

string meuDir(argv[2]);
string nomeCert;
nomeCert += meuDir;
nomeCert += "/cert";
string nomeFinal;
while(count < NUMCERT){
        nomeFinal.clear();
        nomeFinal += nomeCert;
        nomeFinal += to_string(numCert);
        output.open(nomeFinal.c_str());

        if(!output.is_open()){
            cout << "Problema ao gerar arquivo output,
            veja memoria" << endl;
            exit(1);
        }

        while(inCerts.get() != '('){
        }

        escaped = 0;
        sair = 0;
        while(!sair){

                while((aux = (char)inCerts.get()) != ',' || escaped
                        if(!escaped && aux == ')'){
```

```
                                        sair = 1;
                                        break;
                                }

                                if(aux == (char)39){
                                        if(escaped){
                                                escaped = 0;
                                        }else{
                                                escaped = 1;
                                        }
                                }

                                output << aux;

                        }

                        output << endl;

                }
                count++;
                output.close();
                numCert++;
        }

        cout << "Lido ate posicao: " << inCerts.tellg() << endl;



        return 0;
}
```

Anexo II

Programa para extrair os módulos dos certificados

```
//pegarosmod.cpp

#include <cstdio>
#include <fstream>
#include <cstring>
#include <iostream>

using namespace std;

int main(int argc,char** argv){
    ifstream input;
    ofstream output;
    if(argc != 2){
        printf("Erro args\n");
        return 1;
    }

    input.open(argv[1]);
    if(!input.is_open()){
        printf("Erro ao abrir cert\n");
    }

    int i,j;
    char straux[2000];
    for(i = 0;i<9;i++){
        input.getline(straux,2000);
    }

    input.getline(straux,2000);

    char strModulus[2000];
```

```
    j = 0; i = 0;
    while(straux[i] != '\0'){
        if(straux[i] != ':' && straux[i] != '\''){
            strModulus[j] = straux[i];
            j++;
        }
        i++;
    }
    strModulus[j] = '\0';

    char outName[300] = "modDir/";

    strcat(outName,argv[1]);

    output.open(outName);
    output << strModulus;

    input.close();
    output.close();


    return 0;
}
```

Anexo III

CertTest - programa para realizar MDC entre pares de módulos

```
//main2_certtest.c

#include "miracl.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#include <dirent.h>

#define ANSI_COLOR_RED     "\x1b[31m"
#define ANSI_COLOR_GREEN   "\x1b[32m"
#define ANSI_COLOR_YELLOW  "\x1b[33m"
#define ANSI_COLOR_BLUE    "\x1b[34m"
#define ANSI_COLOR_MAGENTA "\x1b[35m"
#define ANSI_COLOR_CYAN    "\x1b[36m"
#define ANSI_COLOR_RESET   "\x1b[0m"


#define MOD_TAM 1200

#define TYPE_PEM 0
#define TYPE_DER 1
#define TIPO_CUSTOM 3

#define TAM_MODULE 1200
#define NUM_DIGITOS_MIRACL 1200
#define BASE_MIRACL 16
#define TIPO_BIN 1
#define TIPO_ASCII 2

#define setDebug 0
```

```
#define NORMAL 0
#define IGUAL 1
#define PERIGO 2
#define NONE 3

#define FX_ini 100
#define FX_fin 400

//site importante
//http//www.mobilefish.com/services/rsa_key_generation/rsa_key_generation.p
//viet spider
//httrack
//compilar com
//gcc -Wall main2_certtest.c -lmiracl -lcrypto -o exe

//warning variavel global
char nomeProOutput[200];


typedef struct t_certificate{
        char filename[300];
        char module[TAM_MODULE];
        int certType;
}t_certificate;




char* mdcComStrings(char* x, char* y,char* result){
        big xd = mirvar(0);
        big yd = mirvar(0);
        big mdcResult = mirvar(0);
        int ret1,ret2;

        if(setDebug){
                printf("mdc\n");
                printf("%s\ninput2\n%s\n\n",x,y);
        }


        big xBig = mirvar(0);
        big yBig = mirvar(0);

        ret1 = cinstr(xBig,x);
        ret2 = cinstr(yBig,y);
```

```
        if(ret1 < 100 || ret2 < 100){
                printf("Cuidado, input de mdc inesperado\n");
                printf("Inputs recebidos\n%s\ninput2\n%s\n\n",x,y);

        }

        if(setDebug){
                printf("Fazendo o mdc com strings\n");
        }

        xgcd(xBig,yBig,xd,yd,mdcResult);

        //printf("Passou do xgcd\n");

        int len = cotstr(mdcResult,result);

        assert(len != 0);

        mirkill(xd);
        mirkill(yd);
        mirkill(xBig);
        mirkill(yBig);
        mirkill(mdcResult);

        //printf("Passou de todo o mdc\n");

        return result;
}


char* readFile(char* filename,char* result)
{
    FILE* file = fopen(filename,"r");

        if(setDebug){
                printf("Filename recebido %s\n",filename);
        }
    if(file == NULL)
    {
                printf("nofile\n");
        return NULL;
    }

    fseek(file, 0, SEEK_END);
    long int size = ftell(file);
```

```
    rewind(file);

    char* content = calloc(size + 1, 1);

    fread(content,1,size,file);

        strcpy(result,content);

        fclose(file);
    return content;
}

//retorna erros para serem analizados
int recebeDiretorio(char* diretorio){
        DIR* dirAtual;
        struct dirent *arquivo;
        int i;
        int numNos;
        char prefix[200];
        int ind = 0;
        strcpy(prefix,diretorio);
        char c;
        int barra = 0;
        while((c = prefix[ind]) != '\0'){
                if(c == '/' && prefix[ind+1] == '\0'){
                        barra = 1;
                }
                ind++;
        }
        if(!barra){
                char minhabarra[2];
                minhabarra[0] = '/';
                minhabarra[1] = '\0';
                strcat(prefix,minhabarra);
        }


        if((dirAtual = opendir(diretorio)) == NULL){
                return 1;
        }

        numNos = 0;
        while((arquivo = readdir(dirAtual)) != NULL){
                if(strcmp(arquivo->d_name,"..") && strcmp(arquivo->d_name,"
                        numNos++;
                }
```

```
}

t_certificate *certificadosVetor = (t_certificate*) calloc(numNos, s
if(certificadosVetor == NULL){
        printf("erro no calloc\n");

}


rewinddir(dirAtual);

char stringModulo[TAM_MODULE];
char straux[100];
i = 0;
while((arquivo = readdir(dirAtual)) != NULL){
        if(strcmp(arquivo->d_name,"..") && strcmp(arquivo->d_name,"
                strcpy(certificadosVetor[i].filename,arquivo->d_nam
                strcpy(straux,prefix);
                strcat(straux,arquivo->d_name);
                readFile(straux,stringModulo);
                strcpy(certificadosVetor[i].module,stringModulo);
                i++;
                if(setDebug){
                        printf("%s\n",stringModulo);
                }
        }
}


FILE* ptResultado;
ptResultado = fopen(nomeProOutput,"w");

if(ptResultado == NULL){
    printf("Erro ao criar arquivo de output!\n");
    exit(1);
}

int j;
char mdcResultado[TAM_MODULE];
char moduloA[TAM_MODULE];
char moduloB[TAM_MODULE];
int situacaoAtual;
int tamModA;
int tamModResult;
for(i=0;i<numNos;i++){
    if(!setDebug){
```

```
                system("clear");
            }
            printf("Progresso: %.2f%%\n",((float)i/(float)numNos) * 100);
                for(j = i + 1;j<numNos;j++){
                        //fazer mdc entre i e j
                        strcpy(moduloA,certificadosVetor[i].module);
                        strcpy(moduloB,certificadosVetor[j].module);
                        if(setDebug){
                                printf("mdc entre\n%s e %s\n",certificadosV
                        }

                        mdcComStrings(moduloA,moduloB,mdcResultado);

                        tamModA = strlen(moduloA);
                        tamModResult = strlen(mdcResultado);


                        if(!strcmp(mdcResultado,"1")){
                                situacaoAtual = NORMAL;
                        }else if(tamModResult - tamModA >= -3 && tamModRes
            tamModA <= 3){
                                situacaoAtual = IGUAL;
                                fprintf(ptResultado,"**********************
                x %s\nModulos iguais\n%s\n\n",certificadosVetor[i].filenam
                mdcResultado);
                        }else{
                                situacaoAtual = PERIGO;
                                fprintf(ptResultado,ANSI_COLOR_RED"
                ********************************
                ***\nMDC\n%s x %s\nPrimo em comum\n%s\n\n"ANSI_COLOR_RESE
certificadosVetor[j].filename,mdcResultado);
                        }


                }
                fflush(ptResultado);
        }
        fclose(ptResultado);

        return 0;
}
```

```
//recebe 2 ou 3 argumentos, sendo que se receber 2 tem q ser
help, se for 3 eh a execucao padrao em 2 certificados
int main(int argc, char** argv){
        int retDir;
        char diretorioOut[200];

        miracl *mip = mirsys(NUM_DIGITOS_MIRACL, BASE_MIRACL);
        mip->IOBASE = 16;

        if(argc != 3){
                printf("Deve-se passar um diretorio contendo modulos RSA
        e um nome para output gerado automaticamente na pasta outputs/\n");
                return 1;
        }else{
                strcpy(diretorioOut,"outputs/");
                strcat(diretorioOut,argv[2]);
                strcpy(nomeProOutput,diretorioOut);
                retDir = recebeDiretorio(argv[1]);
        }

        if(retDir == 1){
                printf("Diretorio especificado nao existe\n");

        }


        return 0;
}
```

Anexo IV

Resultados do teste CUSTOM

```
**********************************
MDC
teste1  x  teste2igual1
Modulos iguais
B5AFE730A6BFE533CDABE6B0A027FE13A3088AC315F13579F17FE837DEF9F6812A72
5E81889134E8C7D42128335A61C86233191E0AF6F500A72E0186695FADA92F50A42A
4D88E685E660F102089FF44669C25E64ECE1B68CEFB4808B6678F9C69910A5DFEC52
762C834A6333ED6FA264F58C279C9AE736032D949F500BF9A1C5

**********************************
MDC
teste1  x  teste3pricomum
Primo em comum
FC02997968CEBEC5F3CFF85EA3FF1D9E382A7BBB4DB59DC912600E9CDCB6040940D3
55458525EEFEF25E95848385F8484CC78D765272B62DA8FD12765B662B8D

**********************************
MDC
teste2igual1  x  teste3pricomum
Primo em comum
FC02997968CEBEC5F3CFF85EA3FF1D9E382A7BBB4DB59DC912600E9CDCB6040940D3
55458525EEFEF25E95848385F8484CC78D765272B62DA8FD12765B662B8D
```

Anexo V

Instruções para coletar os certificados e realizar o teste

```
Instrucoes

-Abra o diretorio certtest
-faca um make
-Reuna certificados em uma pasta
-caso necessario use o egrep_cola para fazer um grep para reunir os
certificados da icp brasil
-crie um subdiretorio modDir na pasta citada
-transfira os arquivos pegar e spript_getmodulos.sh para a
pasta certificados
-rode o script
-entre na modDir e transfira o script_removebrancos para la, rode-o
-rode o certTest2 com argumentos <pasta onde estao os modulos>
<nome do output>
-um relatorio sera dado dentro da pasta outputs
```