Assinatura digital Rabin-Williams sem randomização e com prova eficiente de segurança

Bernardo Caraponale Magri

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa: Ciência da Computação
Orientador: Prof. Dr. Routo Terada

São Paulo, Abril de 2012

Assinatura digital Rabin-Williams sem randomização e com prova eficiente de segurança

Esta dissertação contém as correções e alterações
sugeridas pela Comissão Julgadora durante a defesa
realizada por Bernardo Caraponale Magri em 13 de abril de 2012.
O original encontra-se disponível no Instituto de
Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

- Prof. Dr. Routo Terada (orientador) — IME-USP
- Prof. Dr. Jeroen Van de Graaf — UFMG
- Prof. Dr. Rodrigo Custódio — UFSC

Agradecimentos

Agradeço ao professor Routo Terada pela confiança em me aceitar como aluno, e pela ajuda durante todo o mestrado. Agradeço também aos meus colegas do IME, em especial aos colegas do laboratório de segurança de dados. Agradeço a minha família pelo apoio e incentivo, e por sempre estarem ao meu lado nos momentos mais difíceis.

Resumo

Assinatura digital Rabin-Williams sem randomização e com prova eficiente de segurança.

Com o surgimento da criptografia de chave pública, muito esforço foi feito para a criação de protocolos de criptografia e de assinatura que fossem comprovadamente seguros contra indivíduos maliciosos. Existem várias definições de segurança, tanto para protocolos de criptografia como para protocolos de assinatura, e também existem vários modelos de adversário, que simulam um indivíduo malicioso tentando corromper o protocolo.

A família de protocolos de assinatura *Rabin* possuí os recordes de velocidade de verificação da assinatura, chegando a ser até 100 vezes mais rápida do que o *RSA*. Este trabalho apresenta uma redução eficiente de segurança no modelo do oráculo aleatório para uma variante do protocolo de assinatura Rabin descrito por Daniel J. Bernstein [Ber08], onde não é necessário o uso de nenhuma função para geração de bits *pseudo-aleatórios*, o que torna o protocolo mais robusto. A redução apresentada é uma redução polinomial e eficiente do problema da fatoração de inteiros para o problema de *quebrar* o protocolo *Principal Rabin-Williams* $B = 0$.

Palavras-chave: Criptografia, Assinatura Digital, Rabin-Williams, RSA.

Abstract

Rabin-Williams digital signature without randomization and with tight security proof.

With the development of public-key cryptography, many efforts were made to build encryption and signature protocols that were provably secure against malicious adversaries. To determine if a protocol is secure or not, one needs first to define what means for a protocol to be secure. There are many definitions of security for encryption and signature protocols, and there are many adversary models to simulate the behaviour of a malicious adversary against a given protocol.

The *Rabin* family of signature protocols has the speed records for verification of signature, being up to 100 times faster than *RSA*. This work presents a tight security proof in the random oracle model for a variant of the Rabin signature protocol presented by Daniel J. Bernstein [Ber08], that does not require the use of *pseudo-random* bits, making the protocol more robust. The proof presented here is a polynomially tight reduction for the problem of integer factorization to the problem of *breaking* the *Principal Rabin-Williams* $B = 0$ protocol.
Keywords: Cryptography, Digital Signature, Rabin-Williams, RSA.

Sumário

Agradecimentos **iii**

Resumo **v**

Abstract **vii**

Lista de Figuras **xi**

Lista de Tabelas **xiii**

1 Introdução **1**
1.1 Motivação e Contribuição 3
1.2 Organização do trabalho 4

2 Preliminares **5**
2.1 Divisibilidade 5
2.1.1 Máximo Divisor Comum 6
2.1.2 Números Primos 7
2.2 Congruências 8
2.2.1 Teorema chinês do resto 9
2.2.2 Função φ de Euler 10
2.2.3 O Teorema de Euler e o Pequeno Teorema de Fermat 10
2.3 Resíduos Quadráticos 12
2.3.1 Resíduos Quadráticos módulo p 12
2.3.2 Resíduos Quadráticos módulo $n = pq$ 16
2.4 Algumas definições 17

3 Trabalhos Relacionados **19**
3.1 Assinaturas digitais 19
3.1.1 Criptanálise de assinaturas 19
3.2 Assinatura RSA 21
3.2.1 O Protocolo de assinatura RSA 21
3.2.2 Prova do algoritmo RSA 21
3.2.3 Propriedades Matemáticas do RSA 22

3.2.4 Ataques ao protocolo de assinatura RSA . . . 22
3.3 Protocolo de assinatura RSA ISO/IEC 9796-2 . . . 23
3.3.1 O Padrão ISO/IEC 9796-2 . . . 24
3.3.2 Ataque ao Padrão . . . 24
3.4 Segurança demonstrável . . . 24
3.5 Protocolo de assinatura PSS . . . 27
3.5.1 Descrição do PSS . . . 28
3.5.2 Segurança do PSS . . . 29
3.5.3 Nova Prova de Segurança do PSS . . . 30
3.6 Variante do PSS com prova eficiente de segurança sem Randomização . . . 30
3.6.1 Permutações *Claw-Free* . . . 31
3.6.2 Descrição do Protocolo . . . 32
3.6.3 Prova de Segurança . . . 32
3.7 Protocolo de Assinatura Rabin . . . 34
3.7.1 Descrição do Protocolo . . . 34
3.7.2 Raiz quadrada principal . . . 35

4 O protocolo *Rabin-Williams* 37
4.1 O protocolo de assinatura Rabin-Williams . . . 38
4.2 Método de Bernstein para provas de segurança . . . 42
4.3 Prova eficiente de segurança . . . 44
4.3.1 Inversão cega . . . 44
4.3.2 Inversão selecionada com uma assinatura . . . 46
4.3.3 Inversão selecionada com muitas assinaturas . . . 47

5 Contribuição 51
5.1 Inversão existencial . . . 51

6 Conclusão 55
6.1 Trabalhos futuros . . . 55

Referências 57

A Implementação 59

Lista de Figuras

3.1 O protocolo OAEP 27
3.2 Protocolo de Assinatura PSS 28

Lista de Tabelas

1.1 Reduções de protocolos de assinatura baseados em raízes modulares. 3

4.1 Recomendação do NIST [BR11] para tamanhos de chave RSA. 39
4.2 Relação entre os parâmetros e, f e o símbolo de *Legendre* 41

Capítulo 1

Introdução

As assinaturas digitais tem o mesmo objetivo que as assinaturas convencionais em papel tentam alcançar, que são, autenticação do remetente, não-repudiação, e garantia da integridade da mensagem. Para um protocolo de assinatura ser utilizado na prática ele precisa ser eficiente em seu tempo de execução (tanto para realizar uma assinatura quanto para verificar a autenticidade de uma assinatura), e precisa garantir um certo nível de segurança contra possíveis falsificações. Alcançar estes dois requisitos citados não é uma tarefa fácil, e normalmente os projetistas dos protocolos estabelecem uma *escolha* (*trade-off*) entre eficiência e segurança, dependendo do cenário em que o protocolo será utilizado.

Depois do início da era da criptografia de chave pública, que começou com o artigo de Diffie e Hellman [DH76], surgiram vários novos problemas criptográficos, e vários protocolos de criptografia e assinatura. Inicialmente, esses protocolos não tinham nenhum tipo de prova de sua segurança, e eram considerados seguros quando resistiam a vários ataques durante alguns anos. O que normalmente ocorria é que esses protocolos eram quebrados (para uma definição de *quebra* de protocolos vide definição 2.4.4 na página 17) depois de alguns anos de tentativas [CR88], deixando todos os usuários deste protocolo vulneráveis a indivíduos maliciosos.

Logo depois começaram a criar as chamadas *provas de segurança* utilizando técnicas da *teoria da complexidade computacional*, onde um problema reconhecidamente difícil era *reduzido* polinomialmente, através de algoritmos de redução, para o problema de se quebrar o protocolo. Se essa redução fosse possível, e fosse polinomial, então poderíamos afirmar que o problema de se quebrar o protocolo é pelo menos tão difícil quanto o problema reconhecidamente difícil. Um questionamento dessa abordagem é que esse fator polinomial pode ser a diferença entre um protocolo ser seguro no *mundo real* ou não.

Bellare e Rogaway apresentaram em 1993 [BR93] um modelo para *facilitar* a criação de reduções de segurança para protocolos, que eles chamaram de *modelo do oráculo aleatório*. Esse modelo assume que a função de hash utilizada possui uma distribuição uniforme. Nesse mesmo artigo, eles também introduziram um protocolo muito simples de assinatura chamado FDH (*Full domain hash*) que é baseado no problema do RSA. O FDH utiliza uma função de *hash* que mapeia mensagens de qualquer tamanho para valores dentro do grupo do módulo n utilizado, e depois aplica a função RSA para *assinar* a saída dessa função de *hash*. Nesse artigo, Bellare e Rogaway conseguem provar

que o FDH está relacionado com o RSA por um fator da ordem de 2^{60}, o que não garante muita segurança.

Em 1996, Bellare e Rogaway [BR96] começaram a tratar de reduções *eficientes* de segurança, onde o fator polinomial que relaciona os dois problemas é *pequeno*, e a probabilidade de se quebrar o problema reconhecidamente difícil é *quase* igual a probabilidade de se quebrar o protocolo, em *quase* o mesmo tempo de execução. Essa linha de pesquisa ficou conhecida como *segurança exata* ou *segurança concreta*. No mesmo artigo de 1996 [BR96] Bellare e Rogaway apresentaram o primeiro protocolo de assinatura com uma redução *eficiente* de segurança, chamado PSS. O PSS é baseado no RSA, e um pouco mais complexo do que o FDH. Apesar de o PSS possuir uma característica muito importante que o FDH não possui (redução *eficiente*), ele introduz uma característica indesejável a protocolos de assinatura, que são bits aleatórios concatenados à mensagem antes de aplicar a função de hash. O que permitiu a redução eficiente do PSS foram os bits aleatórios, então obtivemos uma propriedade bastante desejável (redução eficiente) introduzindo uma não tão desejável (bits aleatórios).

Michael Rabin introduziu em 1979 [Rab79], inspirado pelo RSA, um novo problema criptográfico, que consistia na dificuldade de encontrar raízes quadradas módulo um número composto. O mais impressionante no problema criptográfico *Rabin* é que ele foi provado ser equivalente ao problema da fatoração, isto é, se alguém conseguir obter raízes quadradas módulo um número composto então ele consegue obter a fatoração do módulo composto. O protocolo de assinatura descrito por Rabin em [Rab79] é mais rápido do que o protocolo de assinatura RSA, porque no caso do *Rabin* é necessário apenas uma multiplicação modular para verificar a validade de uma assinatura. A única desvantagem da assinatura *Rabin* em relação à assinatura RSA é que a assinatura *Rabin* consegue assinar *apenas* 1/4 das mensagens possíveis. Isso se deve ao problema em que o protocolo é baseado, que será abordado durante o trabalho.

Dentre os protocolos que estabelecem uma prova de segurança para qualquer nível de segurança aceitável, os protocolos baseados em *Rabin* são os mais eficientes. A exceção é apenas em cenários onde o tamanho em bits de uma assinatura é mais importante do que a velocidade de verificação da assinatura, e nesse caso outros protocolos seriam mais eficientes, como por exemplo o protocolo ElGamal, que é baseado no problema do logaritmo discreto.

Hugh Williams, em 1980 [Wil80] utilizou o problema criptográfico proposto por Rabin e criou uma variação do protocolo de assinatura *Rabin*, que é conhecido como *Rabin-Williams*. O que Williams fez foi estender o protocolo proposto por Rabin, eliminando a limitação de assinar apenas 1/4 das mensagens possíveis.

Em 2003, Daniel J. Bernstein [Ber03] publicou uma prova de segurança eficiente para uma das variações do protocolo de assinatura *Rabin-Williams*, e em 2008, o próprio Bernstein estendeu seu resultado [Ber08] apresentando uma prova de segurança eficiente para uma outra variação do protocolo de assinatura *Rabin-Williams* que não necessita de bits aleatórios concatenados à mensagem. Esse resultado de 2008 é considerado por Bernstein [Ber08] o estado da arte em protocolos de assinatura (em que a velocidade da verificação da assinatura é o fator mais importante).

O objetivo deste trabalho é estender o artigo de Bernstein de 2008 [Ber08], e estabelecer uma relação entre a segurança do protocolo de assinatura *Principal Rabin-Williams* $B = 0$ e o problema da fatoração de números inteiros, mais especificamente, o objetivo é apresentar uma redução *eficiente*

Tabela 1.1: Reduções de protocolos de assinatura baseados em raízes modulares.

	B, quantidade de bits aleatórios na entrada do hash		
	B Grande (≥ 180)	B = 1	B = 0
Não-Estruturado Variável Rabin-Williams	redução eficiente (1996 Bellare e Rogaway)	sem segurança (ataque fácil)	sem segurança (ataque fácil)
Variável Principal Rabin-Williams	redução eficiente (2008 Bernstein)	redução ineficiente	redução eficiente (este trabalho)
Variável RSA	redução eficiente (1996 Bellare e Rogaway)	redução ineficiente (1993 Bellare e Rogaway)	redução ineficiente (1993 Bellare e Rogaway)
Fixo RSA	redução eficiente (1996 Bellare e Rogaway)	redução eficiente (Katz e Wang)	redução ineficiente (1993 Bellare e Rogaway)
Fixo Principal Rabin-Williams	redução eficiente (2008 Bernstein)	redução eficiente (2008 Bernstein)	redução eficiente (este trabalho)
Fixo Não-Estruturado Rabin-Williams	redução eficiente (1996 Bellare e Rogaway)	redução eficiente (2008 Bernstein)	redução eficiente (2008 Bernstein)

do problema da fatoração de números inteiros para o problema de falsificar alguma assinatura do protocolo *Principal Rabin-Williams B = 0*.

A tabela 1 mostra as reduções dos protocolos de assinatura baseados em raízes modulares (i.e RSA e *Rabin*), e quando essas reduções foram alcançadas. Note que as células que correspondem aos protocolos *Fixo/Variável Principal Rabin-Williams B = 0* foram preenchidas por este trabalho, e correspondem ao estado da arte tratando-se de protocolos eficientes (*rápidos*) de assinatura.

1.1 Motivação e Contribuição

A motivação para este trabalho foi encontrar uma redução eficiente para um protocolo que é o estado da arte tratando-se de assinaturas eficientes, podendo assim *fortalecer* sua segurança relacionando o problema de se *quebrar* o protocolo com o problema da fatoração.

As contribuições do trabalho são:

- A prova de segurança eficiente no modelo do oráculo aleatório para o protocolo de assinatura *Principal Rabin-Williams B = 0*.
- Implementação do protocolo *Principal Rabin-Williams B = 0* em uma linguagem de alto nível.

1.2 Organização do trabalho

No capítulo 2 apresentaremos conceitos fundamentais para o desenvolvimento do trabalho, bem como algumas definições. No capítulo 3 iremos discutir alguns trabalhos relacionados ao assunto, e que serviram de base para este trabalho. No capítulo 4 será apresentado o protocolo de estudo deste trabalho e nossos resultados referentes à prova de segurança do protocolo. No capítulo 5 apresentamos algumas conclusões e trabalhos futuros.

Capítulo 2

Preliminares

Neste capítulo serão apresentados ao leitor os conceitos básicos que serão necessários para a compreensão deste trabalho. A maior parte do capítulo é destinada a conceitos da teoria dos números. O leitor que sentir necessidade poderá encontrar um maior aprofundamento do assunto em [Sho06, Sil97, Ter08].

2.1 Divisibilidade

Considere $a, b, z \in \mathbb{Z}$, dizemos que a divide b, ou $a \mid b$ se $a \cdot z = b$ para algum z. Se a não divide b, ou $a \nmid b$ então não existe um $z \in \mathbb{Z}$ que resolva esta equação. Quando $a \mid b$ dizemos que b é um múltiplo de a, ou que a é um divisor de b.

Teorema 2.1.1. *[Sho06, cap. 1] Para todo $a, b, c \in \mathbb{Z}$ temos:*

(i) $a \mid a$, $1 \mid a$, *e* $a \mid 0$.

(ii) $0 \mid a \iff a = 0$.

(iii) $a \mid b \iff -a \mid b \iff a \mid -b$.

(iv) $a \mid b$ *e* $a \mid c$ *implica que* $a \mid (b + c)$.

(v) $a \mid b$ *e* $b \mid c$ *então* $a \mid c$.

Demonstração. Seja $z_i \in \mathbb{Z}$. No item (i) $a \mid a$ porque podemos dizer que $a \cdot 1 = a$, no segundo caso $1 \mid a$ podemos dizer $1 \cdot a = a$, e no último caso $a \mid 0$ porque podemos dizer $a \cdot 0 = 0$. No item (ii o símbolo $\iff$ significa "se e somente se", pois a única forma de representar $0 \cdot z = a$ é se $a = 0$. O item (iii) diz que $a \mid b \iff (-1) \cdot a \mid b \iff a \mid (-1) \cdot b$. No item (iv) temos que se $a \mid b$ e $a \mid c$ então $a \cdot z_1 = b$ e $a \cdot z_2 = c$, logo $a \cdot (z_1 + z_2) = (b + c)$ que pode ser representado como $a \cdot z_3 = (b + c)$. No item (v) temos que se $a \mid b$ e $b \mid c$ então é fácil ver que $a \mid c$, pois $a \cdot z_1 = b$ e $b \cdot z_2 = c$ significa que $(a \cdot z_1) \cdot z_2 = c$, que pode ser representado como $a \cdot z_3 = c$. □

Teorema 2.1.2. *[Sho06, cap. 1] Para todo $a, b \in \mathbb{Z}$, com $b > 0$ existem $q, r \in \mathbb{Z}$ tais que $a = b \cdot q + r$, e $0 \leq r < b$.*

Demonstração. Considere o conjunto S de todos os números da forma $a - b \cdot t$ para algum $t \in \mathbb{Z}$. O conjunto S é não vazio, e vamos chamar o menor de seus elementos de r. O elemento r é da forma $r = a - b \cdot q$ para algum $q \in \mathbb{Z}$. Temos também que $r < b$ pois senão $r - b$ seria o menor elemento de S, o que seria uma contradição. Isso prova a existência de q e r, agora precisamos provar sua unicidade. Suponha que $a = b \cdot q + r$ e $a = b \cdot q' + r'$, e subtraindo as duas equações obtemos $r - r' = b \cdot (q - q')$. Logo, $r - r'$ é um múltiplo de b, mas como por definição $0 \leq r < b$ e $0 \leq r < b'$, então temos que $|r - r'| < b$. Portanto, a única possibilidade é que $r - r' = 0$, logo $r = r'$. Para mostrar que $q = q'$, substituindo na equação anterior temos que $0 = b(q - q')$, e sabemos que $b > 0$, logo temos que $q - q' = 0$, que implica $q = q'$. □

2.1.1 Máximo Divisor Comum

Para $a, b \in \mathbb{Z}$ chamamos de $d \in \mathbb{Z}$ um divisor comum de a e b se $d \mid a$ e $d \mid b$. Quando todos os outros divisores comuns de a e b dividem d chamamos d de máximo divisor comum de a e b, e denotamos como $d = mdc(a, b)$.

Teorema 2.1.3. *[Sho06, cap. 1] Para todo $a, b \in \mathbb{Z}$ existe um único máximo divisor comum entre a e b chamado de d, e ele pode ser representado como uma combinação linear de a e b da forma*

$$d = as + bt \text{ para } s, t \in \mathbb{Z} \tag{2.1}$$

Demonstração. Suponha que d seja o menor inteiro positivo tal que $d = as + bt$, para $s, t \in \mathbb{Z}$. Agora temos que mostrar que $d \mid a$ e $d \mid b$. Pelo teorema 2.1.2 temos que

$$a = d \cdot q + r \text{ onde } 0 \leq r < d$$

seguindo esse resultado e o de (2.1) temos

$$r = a - dq = a - (sa + tb)q = a - saq - tbq = (1 - qs)a + (-tb)q$$

Isso mostra que r é uma combinação linear de a e b, mas como por definição temos que $0 \leq r < d$ e assumimos que d é o menor inteiro positivo que é uma combinação linear a e b, só podemos ter que $r = 0$, o que mostra que $d \mid a$. A prova que $d \mid b$ é análoga, e o que nos resta provar é que d é o maior divisor comum entre a e b. Assuma por contradição que exista um outro máximo divisor comum de a e b, chamado $d' = mdc(a, b)$, tal que $d' > d$. Por definição $d' \mid a$, $d' \mid b$ e $d = sa + tb$, e pelo item (iv) do teorema 2.1.1 temos que $d' \mid d$, o que implica $d' \leq d$, que é uma contradição. Então isso conclui que d é de fato o máximo divisor comum entre a e b. Para mostrar que d é único, assuma que $e = mdc(a, b)$, então temos que $d \mid e$ e $e \mid d$ e no entanto $d = \pm e$. Como por definição d e e são inteiros positivos temos que $d = e$, provando que existe um único $mdc(a, b)$. □

Teorema 2.1.4. *[Sho06, cap. 1] Sejam $a, b, c \in \mathbb{Z}$, tal que $c \mid ab$ e $mdc(a, c) = 1$ então $c \mid b$.*

Demonstração. Suponha que $c \mid ab$ e que $mdc(a, c) = 1$. Pelo teorema 2.1.3 temos que $as + ct = 1$ para algum $s, t \in \mathbb{Z}$. Multiplicando essa equação por b obtemos

$$bas + bct = b$$

Por hipótese temos que $c \mid ab$, e claramente $c \mid bct$ então c divide o lado esquerdo da equação, o que implica que $c \mid b$. □

Teorema 2.1.5. *[Sho06, cap. 1] Seja p um número primo, e seja $a, b \in \mathbb{Z}$. Se $p \mid a \cdot b$ então $p \mid a$ ou $p \mid b$.*

Demonstração. Se $p \mid a$ então terminamos a prova. Então assuma que $p \nmid a$, agora precisamos mostrar que $p \mid b$. Como sabemos que p é um número primo então temos que $mdc(a, p) = 1$, e pelo teorema 2.1.4 temos que $p \mid b$. □

2.1.2 Números Primos

Seja n um número inteiro positivo. Dizemos que n é um número primo se os únicos números que dividem n sejam 1 e o próprio n. Quando um número não é primo, ele é chamado de número composto. Apesar de ser possível estender o conceito de números primos para incluir os números negativos, não iremos fazer isso neste trabalho. O número 1 não é considerado primo e nem composto.

Teorema 2.1.6 (Teorema Fundamental da Aritmética)**.** *[Sho06, cap. 1] Seja n um número inteiro. Então n pode ser representado por um produto de primos*

$$n = \pm p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_i^{e_i}$$

onde p_i são primos distintos, e_i são inteiros positivos, e tal que $p_1 \leq p_2 \leq \ldots \leq p_i$, e essa representação é única.

Demonstração. Esse teorema diz que todos os números inteiros podem ser representados por produtos de números primos. Se n é primo então a representação seria $n = n \cdot 1$. Essa prova é dividida em duas partes, na primeira parte temos que mostrar que qualquer número inteiro é um produto de um ou mais primos, e na segunda parte basta mostrar que esse produto é único. A primeira parte vamos provar por indução em n. Suponha que $n = 1$, nesse caso a afirmação é verdadeira, pois n é um produto de 0 primos. Agora seja $n > 1$, e assuma que todos os números inteiros positivos[1] menores do que n possam ser representados por um produto de primos. Se n for primo então a afirmação também é verdadeira, pois n é um produto de 1 primo (ele mesmo), agora suponha que n seja composto, isto é, existem $a, b \in \mathbb{Z}$, tal que $1 < a < n$ e $1 < b < n$ e $n = a \cdot b$. Pela hipótese indutiva, tanto a quanto b podem ser representados como um produto de primos (pois são menores do que n), então consequentemente n também pode ser representado por um produto de primos, o

[1] Note que estamos considerando n como sendo positivo, pois se n fosse negativo, poderíamos multiplicar n por -1 e reduzir para o caso em que n é positivo.

que completa a primeira parte de nossa prova. Na segunda parte da prova, precisamos mostrar que se existirem duas formas de se representar o mesmo número através de produtos de primos, estas duas formas são iguais, podendo apenas diferir na ordem dos fatores. Suponha que

$$n = p_1 \cdot p_2 \cdot \ldots \cdot p_i = q_1 \cdot q_2 \cdot \ldots \cdot q_j \tag{2.2}$$

Vamos provar por indução em i. Suponha que $i = 0$ então j também tem que ser 0, essa é a base da indução. Agora suponha que $i > 0$, e que a hipótese é válida para $i - 1$. Como $i > 0$ temos que $j > 0$, e como p_1 obviamente divide o lado esquerdo da equação (2.2) p_1 também tem que dividir o lado direito (pois temos uma igualdade). Logo temos que $p_1 \mid q_1 \cdot q_2 \cdot \ldots \cdot q_j$, e utilizando o resultado do teorema 2.1.5 sabemos que $p_1 \mid q_k$, para algum $k = \{1, 2, 3, \ldots, j\}$. Como todos os q_k são primos, então $p_1 = q_k$,e simplesmente cancelamos p_1 do lado esquerdo da equação (2.2) e q_k do lado direito. Se continuarmos esse processo para todos os i provaremos que $i = j$ e que na equação (2.2) os primos do lado esquerdo são iguais aos primos do lado direito. □

Teorema 2.1.7. *[Sho06, cap. 1] Existem infinitos números primos.*

Demonstração. Vamos provar este fato por contradição. Suponha que existam finitos números primos, e vamos representar esse conjunto de primos por $\{p_1, p_2, p_3, \ldots, p_i\} \forall i \in \mathbb{N}$. Agora vamos chamar de n o produto de todos os números do conjunto mais um, isto é, $n = (p_1 \cdot p_2 \cdot p_3 \cdot \ldots \cdot p_i) + 1$. Sabemos que n pode ser representado com um produto de primos. Se dividirmos n por qualquer um dos p_i primos obteremos um resto de 1, logo, os números primos que compõem n não estão contidos no conjunto inicial, que supostamente continha todos os números primos, o que é uma contradição. □

2.2 Congruências

Seja n um inteiro positivo, e considere $a, b, n \in \mathbb{Z}$, dizemos que a é congruente a b módulo n, ou $a \equiv b \pmod{n}$ se $n \mid (a - b)$. Se $n \nmid (a - b)$ então dizemos que a não é congruente a b módulo n, ou $a \not\equiv b \pmod{n}$. Equivalentemente, dizemos que se $a \equiv b \pmod{n}$ então podemos representar a como $a = b + \gamma \cdot n$, para algum $\gamma \in \mathbb{Z}$.

Teorema 2.2.1. *[Sho06, cap. 2] Seja n um inteiro positivo, para todos $a, b, c \in \mathbb{Z}$ temos:*

(i) $a \equiv a \pmod{n}$.

(ii) $a \equiv b \pmod{n}$ *implica que* $b \equiv a \pmod{n}$.

(iii) $a \equiv b \pmod{n}$ *e* $b \equiv c \pmod{n}$ *implica que* $a \equiv c \pmod{n}$.

Demonstração. O item (i) é fácil de enxergar, pois obviamente $n \mid (a - a = 0)$. No item (ii) perceba que se $n \mid (a - b)$ então $n \mid -(a - b)$, que nada mais é que $n \mid (b - a)$. Para o item (iii) observe que se $n \mid (a - b)$ e $n \mid (b - c)$ então $n \mid (a - b) + (b - c)$, que é equivalente a $n \mid (a - c)$. □

Teorema 2.2.2. *[Sho06, cap. 2] Sejam $a, n \in \mathbb{Z}$ tal que $n > 0$, e seja $d = mdc(a, n)$*

(i) Para todo $b \in \mathbb{Z}$ a congruência $az \equiv b \pmod{n}$ tem uma solução $z \in \mathbb{Z}$ se e apenas se $d \mid b$.

(ii) Para todo $z \in \mathbb{Z}$ temos que $az \equiv 0 \pmod{n}$ se e apenas se $z \equiv 0 \pmod{n/d}$.

(iii) Para todo $z, z' \in \mathbb{Z}$ temos que $az \equiv az' \pmod{n}$ se e apenas se $z \equiv z' \pmod{n/d}$.

Demonstração. No item (i) suponha que $b \in \mathbb{Z}$ seja dado, Então temos

$$\begin{array}{ll} & az \equiv b \pmod{n} \text{ para algum } z \in \mathbb{Z} \\ \iff & az = b + ny \text{ para algum } z, y \in \mathbb{Z} \text{ (definição de congruência)} \\ \iff & az - ny = b \text{ para algum } z, y \in \mathbb{Z} \\ \iff & d \mid b \text{ (pelo teorema 2.1.3)} \end{array}$$

No item (ii) temos

$$n \mid az \iff (n/d) \mid (a/d)z \iff n/d \mid z$$

E no item (iii) temos

$$\begin{array}{lll} az \equiv az \pmod{n} & \iff a(z-z) \equiv 0 & \pmod{n} \\ & \iff z - z \equiv 0 & \pmod{n/d} \text{ (pelo item (ii))} \\ & \iff z \equiv z & \pmod{n/d} \end{array}$$

□

Teorema 2.2.3. *[Sho06, cap. 2] Sejam $a, n \in \mathbb{Z}$, a equação $az \equiv 1 \pmod{n}$ só possuí um resultado $z \in \mathbb{Z}$ se $mdc(a, n) = 1$. Esse resultado é chamado de* inverso multiplicativo *de $a \bmod n$, e é denotado como a^{-1}.*

Demonstração. O item (i) do teorema 2.2.2 mostra que a equação $az \equiv 1 \pmod{n}$ só possuí resultado se $d = mdc(a, n) \mid 1$, e o item (iii) mostra que se existe um inverso multiplicativo de $a \pmod{n}$ então ele é único. □

2.2.1 Teorema chinês do resto

Seja $m = m_1 \cdot m_2$, e suponha que $y \in \mathbb{Z}_m$. Considere as equações

$$a_1 = y \bmod m_1$$

$$a_2 = y \bmod m_2$$

O teorema chinês do resto combina a_1 e a_2 para obter y. O teorema também diz que se $mdc(m_1, m_2) = 1$ esse y é único.

Teorema 2.2.4 (Teorema Chinês do Resto)**.** *[Sho06, cap. 2] Sejam $m_1, m_2, \ldots, m_k$ relativamente primos dois a dois, isto é, $mdc(m_i, m_j) = 1$ para $1 \leq i < j \leq k$. Seja $a_i \in \mathbb{Z}_{m_i}$ para $1 \leq i \leq k$ e seja $m = m_1 \cdot m_2 \cdot \ldots \cdot m_k$. Então existe um único $y \in \mathbb{Z}_m$ tal que $y \equiv a_i \bmod m_i$ para $i = 1 \ldots k$.*

Demonstração. Para cada i, seja $n_i = \left(\frac{m}{m_i}\right) \in \mathbb{Z}$. Isto quer dizer que n_i é o produto de todos os m_j tal que $j \neq i$. Pelo teorema 2.1.3 sabemos que $mdc(m_i, n_i) = 1$, logo $\exists b_i \in \mathbb{Z}_{m_i}$ tal que $n_i \cdot b_i \equiv 1 \bmod m_i$. Seja $c_i = n_i \cdot b_i$. Então $c_i = \begin{cases} 1 \bmod m_i \\ 0 \bmod m_j \text{ para } j \neq i \end{cases}$

Seja $y = \sum_{i=1}^{k} (c_i \cdot a_i) \bmod m$. Então $y \equiv a_i \bmod m_i$ para todo i. □

Os Conjuntos $\mathbb{Z}_n$ e $\mathbb{Z}_n^*$. O conjunto dos números inteiros é chamado de $\mathbb{Z}$, e o conjunto dos números inteiros módulo n é chamado de $\mathbb{Z}_n$. O conjunto dos números inteiros módulo n e que são relativamente primos a n é chamado de $\mathbb{Z}_n^*$.

2.2.2 Função φ de Euler

A função φ de Euler é definida para todos os inteiros positivos, sendo

$$\varphi(n) = |\mathbb{Z}_n^*|$$

Em outras palavras, a função φ de Euler representa a quantidade de números em $\mathbb{Z}_n^*$ (de 0 até $n-1$) que possuem inverso multiplicativo módulo n, isto é, que sejam primos em relação a n.

Teorema 2.2.5. *[Sho06, cap. 2] Seja $n \in \mathbb{Z}$ um número primo. A função φ de Euler é definida como $\varphi(n) = |\mathbb{Z}_n^*| = n - 1$.*

Demonstração. Como n é um número primo, temos por definição que $mdc(a, n) = 1$ para todo $a \in \{1, 2, 3, \ldots, n-1\}$ □

Teorema 2.2.6. *[Sho06, cap. 2] Seja $n \in \mathbb{Z}$ um número composto pelo produto dos primos distintos $\{n_1, n_2, \ldots, n_k\}$, para algum $k \in \mathbb{Z}$. A função φ de Euler é definida como*

$$\varphi(n) = |\mathbb{Z}_n^*| = \prod_{i=1}^{k} \varphi(n_i)$$

Demonstração. Podemos provar esse teorema utilizando diretamente o resultado do teorema 2.2.5 e o teorema chinês do resto (2.2.4), pois os componentes de n são números primos. □

2.2.3 O Teorema de Euler e o Pequeno Teorema de Fermat

Seja n um inteiro positivo, e seja $a \in \mathbb{Z}_n^*$. Considere a sequência de potências de a

$$a^0, a^1, a^2, \ldots \tag{2.3}$$

Como o conjunto $\mathbb{Z}_n^*$ é finito, todos os elementos da sequência (2.3) $\in \mathbb{Z}_n^*$. Os elementos de (2.3) começarão a se repetir em algum momento.

Teorema 2.2.7. *[Sho06, cap. 2] Seja n um inteiro positivo, e seja $a \in \mathbb{Z}_n^*$. Existe um valor k, tal que $a^k = a^i$, para algum $i \in \{0, 1, 2, \ldots, k-1\}$, e chamamos o menor k de ordem multiplicativa de a. Mais ainda, afirmamos que $a^k = 1$.*

Demonstração. Por contradição, suponha que $a^k = a^i$ para algum $i \in \{0, 1, 2, \ldots, k-1\}$, então poderíamos dividir ambos os lados da equação por a, obtendo $a^{k-1} = a^{i-1}$, o que contradiz a minimalidade de k. □

Teorema 2.2.8 (Teorema de Wilson)**.** *[Sho06, cap. 2] Seja p um primo ímpar, então*

$$\prod_{b \in \mathbb{Z}_p^*} b = -1$$

Demonstração. Agruparemos os elementos de $\mathbb{Z}_p^*$ em pares, tal que seus produtos sejam igual a 1. Seja $D = \{\{a, b\} \in \mathbb{Z}_p^* \ : \ ab = 1\}$. Sabemos que para todo $a \in \mathbb{Z}_p^*$ existe um $b \in \mathbb{Z}_p^*$ tal que $a \cdot b = 1$. Temos que $a \not\equiv b \bmod p$ em todos os casos, exceto quando $a = \pm 1$, pois $(1) \cdot (1) = 1$ e $(-1) \cdot (-1) = 1$. Então todos os elementos de $\mathbb{Z}_p^*$ pertencem a exatamente um par de D, exceto 1 e -1. Podemos representar esse produto como

$$\prod_{b \in \mathbb{Z}_p^*} b = (1) \cdot (-1) \cdot \prod_{\{a,b\} \in D} (a \cdot b)$$

$$\prod_{b \in \mathbb{Z}_p^*} b = (1) \cdot (-1) \cdot \prod_{\{a,b\} \in D} (1) = -1$$

□

Teorema 2.2.9 (Teorema de Euler)**.** *[Sho06, cap. 2] Seja n um inteiro positivo, e seja $a \in \mathbb{Z}_n^*$. Então $a^{\varphi(n)} = 1$.*

Demonstração. Pelo teorema de Wilson 2.2.8 sabemos que

$$\prod_{b \in \mathbb{Z}_n^*} b = -1$$

Da mesma forma, pelo teorema 2.2.8 temos que

$$\prod_{b \in \mathbb{Z}_n^*} ab = -1$$

Logo, podemos dizer que

$$\prod_{b \in \mathbb{Z}_n^*} ab = a^{\varphi(n)} \cdot \prod_{b \in \mathbb{Z}_n^*} b$$

$$(-1) = a^{\varphi(n)} \cdot (-1)$$

$$a^{\varphi(n)} = 1$$

□

Uma decorrência direta do teorema 2.2.9 é o *Pequeno Teorema de Fermat*, que será descrito a seguir.

Teorema 2.2.10 (Pequeno Teorema de Fermat)**.** *[Sho06, cap. 2] Seja p um número primo, e seja $a \in \mathbb{Z}_p$. Então $a^p = a$.*

Demonstração. Se $a = 0$ a afirmação está correta, então vamos assumir que $a \neq 0$. Como p é um número primo, então temos que $mdc(a, p) = 1$, e logo $a \in \mathbb{Z}_n^*$. Pelo teorema 2.2.9 temos que $a^{p-1} = 1$, e se multiplicarmos ambos os lados da equação por a teremos $a^p = a$. □

2.3 Resíduos Quadráticos

Nesta seção estudaremos um tipo especial de congruência, que é da forma $z^2 \equiv a \pmod{n}$. Esse tipo de congruência é a base do esquema de criptografia e assinatura *Rabin* [Rab79] e *Rabin-Williams* [Wil80] que será tratado neste trabalho.

Definição 2.3.1. Definimos o conjunto de resíduos quadráticos módulo n como

$$(\mathbb{Z}_n^*)^2 = \{b^2 \ : \ b \in \mathbb{Z}_n^*\}$$

Teorema 2.3.2. *[Sho06, cap. 2] Seja n um inteiro positivo, e sejam $a, b \in \mathbb{Z}_n^*$.*

(i) Se $a \in (\mathbb{Z}_n^*)^2$ então $a^{-1} \in (\mathbb{Z}_n^*)^2$.

(ii) Se $a \in (\mathbb{Z}_n^*)^2$ e $b \in (\mathbb{Z}_n^*)^2$ então $ab \in (\mathbb{Z}_n^*)^2$.

(iii) Se $a \in (\mathbb{Z}_n^*)^2$ e $b \notin (\mathbb{Z}_n^*)^2$ então $ab \notin (\mathbb{Z}_n^*)^2$.

Demonstração. No item (i) temos que $a = \gamma^2$, então $a^{-1} = (\gamma^{-1})^2 \in (\mathbb{Z}_n^*)^2$. No item (ii) temos que $a = \gamma^2$, e $b = \delta^2$ então $ab = (\gamma\delta)^2 \in (\mathbb{Z}_n^*)^2$.
Por último, no item (iii) suponha que $a \in (\mathbb{Z}_n^*)^2$, $b \notin (\mathbb{Z}_n^*)^2$ e que $ab \in (\mathbb{Z}_n^*)^2$. Pelo item (i) temos que $a^{-1} \in (\mathbb{Z}_n^*)^2$, e pelo item (ii) temos que $b = a^{-1}(ab) \in (\mathbb{Z}_n^*)^2$, o que é claramente uma contradição[2]. □

2.3.1 Resíduos Quadráticos módulo p

Primeiro iremos estudar os resíduos quadráticos módulo um primo ímpar p.

Teorema 2.3.3. *[Sho06, cap. 2] Seja p um primo ímpar, e seja $b \in \mathbb{Z}_p$. $b^2 = 1$ se e somente se $b = \pm 1 \bmod p$.*

[2]Note que podemos estender este teorema para qualquer potência $m \in \mathbb{N}$.

Demonstração. Se $b = \pm 1$ então claramente $b^2 = 1$. Agora suponha que $b^2 = 1$, que significa que $b^2 \equiv 1 \pmod{p}$

$$p \mid b^2 - 1$$

$$p \mid (b+1)(b-1)$$

Como p é primo, então pelo teorema 2.1.5 temos que $p \mid (b+1)$ ou $p \mid (b-1)$, o que implica que $b = \pm 1 \bmod p$. □

Teorema 2.3.4. *[Sho06, cap. 2] Seja p um primo ímpar, e sejam $a, b \in \mathbb{Z}_p$. $a^2 = b^2$ se e somente se $a = \pm b$.*

Demonstração. A prova é resultante do teorema 2.3.3.

$$a^2 = b^2 \iff (ab^{-1})^2 = 1$$

$$\sqrt{(ab^{-1})^2} = \sqrt{1} \iff (ab^{-1})^2 = \pm 1$$

$$a = \pm b$$

□

Teorema 2.3.5. *[Sho06, cap. 2] Seja p um primo ímpar, então $|(\mathbb{Z}_p^*)^2| = \frac{p-1}{2}$.*

Demonstração. Pelo teorema 2.3.4 temos que $(a)^2 = (-a)^2$ e $a \neq -a$. Em outras palavras, cada resíduo quadrático módulo um primo ímpar p possui duas raízes quadradas módulo p (a e $-a$). Logo, apenas a metade dos elementos de $\mathbb{Z}_n^*$ são resíduos quadráticos módulo p, isto é, $\frac{p-1}{2}$ elementos. □

Teorema 2.3.6 (Critério de Euler)**.** *[Sho06, cap. 2] Seja p um primo ímpar, e seja $a \in \mathbb{Z}_p^*$.*

(i) $a^{\frac{p-1}{2}} = \pm 1$.

(ii) Se $a \in (\mathbb{Z}_p^*)^2$ *então* $a^{\frac{p-1}{2}} = 1$.

(iii) Se $a \notin (\mathbb{Z}_p^*)^2$ *então* $a^{\frac{p-1}{2}} = -1$.

Demonstração. No item (i) seja $\gamma = a^{\frac{p-1}{2}}$, pelo teorema de Euler (2.2.9) temos que $\gamma^2 = a^{p-1} = 1$, e pelo teorema 2.3.3 temos que $\sqrt{1} = \pm 1$. No item (ii) suponha que $a = b^2$, novamente pelo teorema 2.2.9 temos

$$a^{\frac{p-1}{2}} = (b^2)^{\frac{p-1}{2}} = b^{p-1} = 1$$

Para o item (iii) seja $a \in \mathbb{Z}_p^* \setminus (\mathbb{Z}_p^*)^2$, e considere o produto

$$\prod_{b \in \mathbb{Z}_p^*} b$$

Mostraremos que

$$\prod_{b\in\mathbb{Z}_p^*} b = a^{\frac{p-1}{2}}$$

Agruparemos os elementos de $\mathbb{Z}_p^*$ em pares tal que seus produtos sejam igual a a, isto é, seja o conjunto $C = \{\{x, y\} \in \mathbb{Z}_p^* \ : \ xy = a\}$, e note que para cada $x \in \mathbb{Z}_p^*$ temos que existe um único $y \in \mathbb{Z}_p^*$ tal que $xy = a$, este sendo $y = a/x$. Temos a restrição que $x \neq y$ pois senão teríamos $x^2 = a$, o que violaria a hipótese de que $a \notin (\mathbb{Z}_p^*)^2$. Então cada elemento de $\mathbb{Z}_p^*$ pertence a um único par de C, sendo assim

$$\prod_{b\in\mathbb{Z}_n^*} b = \prod_{\{x,y\}\in C} (x \cdot y)$$

$$\prod_{b\in\mathbb{Z}_n^*} b = \prod_{\{x,y\}\in C} (a) = a^{\frac{p-1}{2}}$$

pois temos $(p-1)/2$ pares em C. E utilizando o teorema de Wilson (2.2.8) chegamos à conclusão que

$$\prod_{b\in\mathbb{Z}_n^*} b = a^{\frac{p-1}{2}} = -1$$

□

Símbolo de Legendre

Definição 2.3.7. Para um $a \in \mathbb{Z}_p^*$, o símbolo de Legendre $\left(\frac{a}{p}\right)$ determina se $a \in (\mathbb{Z}_p^*)^2$ ou se $a \notin (\mathbb{Z}_p^*)^2$. Em outras palavras, determina se a é ou não um resíduo quadrático módulo um primo ímpar p.

Se $a \in (\mathbb{Z}_p^*)^2$ então $\left(\frac{a}{p}\right) = 1$, se $a \notin (\mathbb{Z}_p^*)^2$ então $\left(\frac{a}{p}\right) = -1$, e se $p \mid a$ então $\left(\frac{a}{p}\right) = 0$

Teorema 2.3.8. *[Sho06, cap. 12] Seja p um primo ímpar, e sejam $a, b \in \mathbb{Z}$. Então temos*

(i) $\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}} \pmod p$

(ii) $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right) \cdot \left(\frac{b}{p}\right)$

(iii) $a \equiv b \pmod p$ *implica que* $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$

(iv) $\left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}}$

Demonstração. O item (i) é o teorema 2.3.6. Os itens (ii) e (iii) são uma consequência direta do item (i) e da definição do símbolo de Legendre respectivamente. O item (iv) é o mesmo que dizer que 2 é um resíduo quadrático módulo p se e somente se $p \equiv \pm 1 \bmod 8$. A prova pode ser encontrada em [Sho06, cap. 12].

□

Teorema 2.3.9. *[Sho06, cap. 2] Seja p um primo ímpar, e sejam $a, b \notin (\mathbb{Z}_p^*)^2$. Então $ab \in (\mathbb{Z}_p^*)^2$.*

Demonstração. Como temos que $a, b \notin (\mathbb{Z}_p^*)^2$ então pelo teorema 2.3.6 temos que $a^{\frac{p-1}{2}} = -1$ e $b^{\frac{p-1}{2}} = -1$, logo

$$(ab)^{\frac{p-1}{2}} = a^{\frac{p-1}{2}} \cdot b^{\frac{p-1}{2}} = (-1) \cdot (-1) = 1$$

□

Teorema 2.3.10. *[Sho06, cap. 2] Seja p um primo ímpar, então -1 é um resíduo quadrático módulo p se e somente se $p \equiv 1 \pmod 4$.*

Demonstração. Pelo critério de Euler (2.3.6) sabemos que $-1 \in (\mathbb{Z}_p^*)^2 \iff (-1)^{\frac{p-1}{2}} = 1$. Se $p \equiv 1 \pmod 4$ temos que $p = 4m + 1$ para algum $m \in \mathbb{Z}$, logo

$$(-1)^{\frac{(4m+1)-1}{2}} = (-1)^{2m} = 1$$

□

Como p é um primo ímpar, os únicos valores possíveis para p mod 4 são 1 e 3, isto é, se dividirmos p por 4 obteríamos resto de 1 ou 3.

Teorema 2.3.11. *[Sho06, cap. 2] Existem infinitos números primos da forma $p \equiv 3 \pmod 4$.*

Demonstração. Suponha que exista finitos números primos com esse formato, sendo $p_1, p_2, \ldots, p_k$. Seja $M = \prod_{i=1}^{k} p_i$, e seja $N = 4M - 1$. Como $N \equiv 3 \pmod 4$ então tem que existir um $p \equiv 3 \pmod 4$ que divida N, caso contrário N seria da forma $N \equiv 1 \pmod 4$. É fácil ver que esse p não é nenhum dos p_i do conjunto inicial, pois se fosse, então $p \mid 4M$ e $p \mid N$, o que implicaria $p \mid 1$. Isso contradiz a hipótese de que $p_1, p_2, \ldots, p_k$ são os únicos primos com esse formato. □

Teorema 2.3.12. *[Sho06, cap. 2] Existem infinitos números primos da forma $p \equiv 1 \pmod 4$.*

Demonstração. Essa prova é um pouco mais complexa do que a prova anterior. Suponha que exista finitos números primos com esse formato, sendo $p_1, p_2, \ldots, p_k$. Seja $M = \prod_{i=1}^{k} p_i$, e seja $N = 4M^2 + 1$. O motivo de elevarmos M ao quadrado é que não temos como saber se algum fator p de N é da forma $p \equiv 1 \pmod 4$. Da mesma forma como na prova anterior temos que $p_i \nmid N$, e temos um $p \mid N$, então $4M^2 \equiv -1 \pmod p$ pela definição de congruência. Sabemos pelo teorema 2.3.10 que -1 é um resíduo quadrático módulo p se e somente se $p \equiv 1 \pmod 4$. Então existe um outro primo $p \equiv 1 \pmod 4$ que não foi considerado no conjunto inicial, o que é uma contradição. □

Teorema 2.3.13. *Seja p um primo ímpar da forma $p \equiv 3 \bmod 4$, e seja $a \in \mathbb{Z}_p^*$. Se $a^{(p-1)/2} = -1$ então $(-a)^{(p-1)/2} = 1$.*

Demonstração. Suponha que $a^{(p-1)/2} = -1$.

Agora com $(-a)$ temos que $(-a)^{(p-1)/2} = (-1)^{(p-1)/2}(a)^{(p-1)/2} = (-1) \cdot (-1) = 1$, pois p é da forma $p \equiv 3 \bmod 4$, o que resulta em um expoente ímpar. □

2.3.2 Resíduos Quadráticos módulo $n = pq$

Até agora estávamos considerando apenas resíduos quadráticos módulo um primo ímpar p. Agora vamos considerar a classe de resíduos quadráticos módulo um número composto $n = pq$, onde p e q são dois números primos ímpares.

Quando consideramos um módulo composto, por exemplo $n = pq$, trabalhamos com cada fator primo de n separadamente, como na seção anterior, e depois utilizamos o Teorema Chinês do Resto (2.2.4) para combinar os resultados módulo p e q e encontrar o resultado módulo n.

Símbolo de Jacobi

O símbolo de Jacobi $\mathbf{J}_n(a)$ é o equivalente ao símbolo de Legendre para módulos compostos $n = p_1 \cdot p_2 \dots p_k$.

Definição 2.3.14. Sejam $a, n \in \mathbb{Z}$, onde n é positivo e ímpar tal que $n = p_1 \cdot p_2 \cdot \ldots \cdot p_k$, sendo $p_1, p_2, \dots, p_k$ primos ímpares não necessariamente distintos. O símbolo de Jacobi é definido como

$$\mathbf{J}_n(a) = \left(\frac{a}{p_1}\right) \cdot \left(\frac{a}{p_2}\right) \dots \left(\frac{a}{p_k}\right)$$

Por definição, $\mathbf{J}_1(a) = 1$ para todo $a \in \mathbb{Z}$, e $\mathbf{J}_n(a) = 0 \iff mdc(a, n) > 1$.

Teorema 2.3.15. *[Sho06, cap. 12] Sejam m, n inteiros positivos ímpares, e sejam $a, b \in \mathbb{Z}$. Então temos*

(i) $\mathbf{J}_n(ab) = \mathbf{J}_n(a) \cdot \mathbf{J}_n(b)$

(ii) $\mathbf{J}_{mn}(a) = \mathbf{J}_m(a) \cdot \mathbf{J}_n(a)$

(iii) $a \equiv b \pmod{p}$ *implica que* $\mathbf{J}_n(a) = \mathbf{J}_n(b)$

(iv) $\mathbf{J}_n(-1) = (-1)^{\frac{n-1}{2}}$

(v) $\mathbf{J}_n(2) = (-1)^{\frac{n^2-1}{8}}$

(vi) $\mathbf{J}_n(m) = (-1)^{\frac{m-1}{2} \cdot \frac{n-1}{2}} \cdot \mathbf{J}_m(n)$

Demonstração. Seja $n = p_1 \cdot p_2 \dots p_k$, então no item (i) temos que

$$\mathbf{J}_n(ab) = \mathbf{J}_n(a) \cdot \mathbf{J}_n(b) = \left(\frac{ab}{p_1}\right) \dots \left(\frac{ab}{p_k}\right) = \left(\frac{a}{p_1}\right) \dots \left(\frac{a}{p_k}\right) \cdot \left(\frac{b}{p_1}\right) \dots \left(\frac{b}{p_k}\right)$$

Os itens (ii) e (iii) são consequências diretas da definição do símbolo de Jacobi. Os itens (iv-vi) podem ser encontrados em [Sho06, cap. 12]. Note que utilizando o item (vi) é possível calcular $\mathbf{J}_n(m)$ sem conhecer a fatoração de n. □

2.4 Algumas definições

Nesta seção vamos introduzir alguns conceitos de assinaturas digitais que será utilizado durante o trabalho.

Definição 2.4.1 (Protocolo de assinatura)**.** Um protocolo de assinatura é uma tripla de algoritmos *(Gen,Sign,Verify)* e um parâmetro de segurança K. O algoritmo *Gen* é o gerador de chaves, que recebe como entrada 1^K e retorna um par de chaves (pK, sK) que correspondem a chave pública e a chave privada, respectivamente. O algoritmo *Sign* é o algoritmo de assinatura, ele recebe como entrada uma mensagem m a ser assinada, a chave privada sK, e retorna a assinatura $x = Sign_{sK}(m)$. O algoritmo *Verify* recebe como entrada uma mensagem m, uma suposta assinatura x, e a chave pública pK. Ele retorna um bit $Verify_{pK}(m, x)$ que será 1 caso a assinatura seja verdadeira, e 0 caso contrário. É exigido que se $x = Sign_{sK}(m)$ então $Verify_{pK}(m, x) = 1$.

Definição 2.4.2 (Adversário)**.** Chamaremos de adversário um indivíduo mal-intencionado que usa suas habilidades para atacar protocolos de assinatura, e com isso obter falsificações de assinaturas.

Definição 2.4.3 (Vítima)**.** Chamaremos de vítima o dono da chave privada que está sendo o alvo de algum ataque. A vítima é um usuário de um protocolo de assinatura que está sendo atacado por um adversário.

Definição 2.4.4 (Quebra do protocolo de assinatura)**.** Informalmente, dizemos que um adversário *quebra* um protocolo de assinatura se ele consegue gerar pelo menos um par *(mensagem,assinatura)* que não foi gerado previamente pela vítima.

Capítulo 3

Trabalhos Relacionados

Neste capítulo iremos discutir alguns trabalhos relacionados ao assunto, e que serviram de base para o desenvolvimento deste trabalho.

3.1 Assinaturas digitais

As assinaturas digitais tem o mesmo objetivo que as assinaturas convencionais em papel tentam alcançar, que são, autenticação do remetente, não-repudiação, e garantia da integridade da mensagem.

Autenticação do remetente significa que o destinatário (quem recebeu a mensagem) poderá verificar facilmente (utilizando apenas informações públicas) se a mensagem foi de fato assinada pelo remetente.

Não-repudiação significa que o remetente da mensagem (quem assinou a mensagem) não poderá negar (por exemplo, em um tribunal) que de fato assinou aquela mensagem, pois apenas ele conhece a chave secreta necessária para realizar uma operação de assinatura, assim como uma pessoa não pode negar ter assinado um documento de papel, pois sua assinatura poderá ser identificada facilmente através de testes.

Integridade da mensagem significa que o destinatário (quem recebeu a mensagem) tem como verificar (utilizando apenas informações públicas) se a mensagem não foi alterada antes de ser entregue a ele.

3.1.1 Criptanálise de assinaturas

Criptanálise é o termo utilizado para descrever qualquer tentativa de comprometimento da segurança (i.e *ataques*) de um protocolo de criptografia ou assinatura.

Tipos de ataques

Existem quatro tipos básicos de ataque [GMR88].

Ataque de chave pública (*Key-Only attack*). O ataque de chave pública é o ataque mais fraco, no sentido de que o adversário (quem está atacando o protocolo) possui menos informação. Neste caso, o adversário conhece somente a chave pública da vítima.

Ataque de assinatura conhecida (*Known signature attack*). Nesse tipo de ataque, o adversário conhece alguns pares $(mensagem, assinatura)$ que foram gerados previamente pela vítima (provavelmente foram interceptados pelo adversário).

Ataque de mensagem escolhida (*Chosen message attack*). Neste tipo de ataque, o adversário, com a cooperação da vítima, consegue obter assinaturas de mensagens selecionadas previamente.

Ataque adaptativo de mensagem escolhida (*Adaptive chosen message attack*). Neste tipo de ataque, o adversário, com a cooperação da vítima, consegue gerar assinaturas de mensagens escolhidas adaptativamente, isto é, o adversário consegue escolher mensagens para serem assinadas depois de ter obtido assinaturas de outras mensagens.

Objetivos do adversário

O adversário, quando realiza um ataque, pode ter quatro objetivos, que são listados a seguir.

Quebra total (*Total break*). É o nível mais alto de sucesso. Neste caso, o adversário consegue calcular a chave secreta do adversário (e.g obter a fatoração de n). Para ilustrar melhor, imagine que um adversário roubou o *smartcard* de sua vítima (que contém sua chave privada). Se o adversário conseguir alcançar este nível de sucesso em seu ataque, ele conseguirá extrair a chave privada do *smartcard*.

Falsificação universal (*Universal forgery*). O adversário não tem conhecimento da chave secreta da vítima, mas consegue gerar uma assinatura válida (falsificada) para qualquer mensagem.

Falsificação selecionada (*Selective forgery*). O adversário consegue gerar uma assinatura válida (falsificada) para algumas mensagens escolhidas.

Falsificação existencial (*Existential forgery*). O adversário consegue gerar uma assinatura válida (falsificada), mas ele não tem controle sobre a mensagem cuja assinatura foi falsificada. É o nível mais baixo de sucesso.

3.2 Assinatura RSA

O protocolo de assinatura RSA foi introduzido por Rivest, Shamir e Adleman em 1978 [RSA78], sendo o primeiro protocolo de criptografia e assinatura inspirado pelas idéias descritas em Diffie e Hellman, 1978 [DH76]. O RSA é baseado no problema de calcular raízes modulares, que é um problema considerado difícil, mesmo não se conhecendo nenhuma *prova* matemática que comprove essa dificuldade. Os ataques aos protocolos de assinatura RSA tiveram início em 1982 com a publicação de George I. Davida [Dav82], onde foi apresentado um ataque de mensagem escolhida, que resultava em uma falsificação universal. Desde então, vários artigos estenderam o trabalho realizado de Davida, tornando o ataque mais eficiente.

3.2.1 O Protocolo de assinatura RSA

Seja n o módulo da vítima, tal que $n = pq$, sendo p e q primos grandes (por exemplo 512 bits, tornando n 1024 bits); e seja e e d os expoentes público e secreto, respectivamente, tal que $ed \equiv 1 \bmod \varphi(n)$. O expoente secreto d é usado para assinar a mensagem, e o expoente público e é usado para verificar a validade da assinatura. Para assinar uma mensagem M, o remetente deve calcular $C = M^d \pmod n$ e enviar C para o destinatário, o destinatário então deve calcular $M \stackrel{?}{=} C^e \pmod n$; se a igualdade for verdadeira então o destinatário poderá ter certeza que a mensagem foi assinada pelo remetente (ou por alguém com acesso à chave privada do remetente).

A segurança desse esquema se deve à dificuldade de um adversário conseguir inverter a função RSA, que se acredita ser um problema difícil, embora não exista nenhuma prova matemática que suporte este fato.

3.2.2 Prova do algoritmo RSA

Primeiro são escolhidos dois números primos grandes p e q com cerca de $\frac{|n|}{2}$ bits, e então é definido o módulo $n = pq$.

Calcula-se a função $\varphi(n) = (p-1)(q-1)$ e seleciona-se um valor aleatório $e < n$, tal que $mdc(e, \varphi(n)) = 1$.

Utilizando o algoritmo de Euclides estendido é fácil encontrar o valor d, tal que $ed \equiv 1 \pmod{\varphi(n)}$.

Os valores (e, n) formam a chave pública, e os valores (d, n) formam a chave secreta. Para provar que o esquema é válido, é necessário estabelecer a seguinte igualdade

$$M \equiv (M^d)^e \pmod n, \quad \forall\, M \in \mathbb{Z}_n$$

Para o caso em que $M \in \mathbb{Z}_n^*$ temos que

$$ed \equiv 1 \pmod{\varphi(n)}$$

logo

$$ed = k\varphi(n) + 1$$

substituindo os expoentes temos

$$(M^d)^e = M^{ed} = M^{k\varphi(n)+1} \pmod{n}$$

pelo teorema de Euler 2.2.9,

$$\forall a \in \mathbb{Z}_n^*, a^{\varphi(n)} = 1 \pmod{n}$$

logo, chegamos ao resultado desejado

$$M(M^{\varphi(n)})^k \equiv M \pmod{n}$$

3.2.3 Propriedades Matemáticas do RSA

Os ataques ao protocolo de assinatura RSA utilizam suas propriedades matemáticas para explorar *fraquezas* no protocolo [Mis98], que são apresentadas a seguir.

Propriedade 1. Seja P a função pública do RSA.

$$P(x) = x^e \pmod{n}$$

Propriedade 2. Seja S a função secreta do RSA.

$$S(x) = x^d \pmod{n}$$

Propriedade 3. A propriedade recíproca do RSA se deve ao fato que:

$$P \circ S = S \circ P = Identidade$$

Propriedade 4. A propriedade multiplicativa (ou homomorfismo multiplicativo) se deve ao fato que:

$$\forall x, y \quad S(xy) = S(x)S(y)$$

3.2.4 Ataques ao protocolo de assinatura RSA

A seguir serão apresentados dois ataques de mensagem escolhida (*chosen message attack*) no protocolo simples de assinatura RSA, que resultam em uma falsificação universal (*universal forgery*).

Ataque de Davida. O ataque de Davida [Dav82] utiliza a propriedade multiplicativa do RSA, e é descrito a seguir.

1. Seja m a mensagem que se quer falsificar uma assinatura.
2. Sejam $m_1, ..., m_t$ os fatores de m. Note que a fatoração não precisa ser completa (pode haver alguns fatores compostos).
3. Seja $sig = S(m')$ a assinatura da mensagem m'.
4. Suponha que o adversário consiga obter as assinaturas $sig_1, ..., sig_t$ das mensagens $m'.m_1, ..., m'.m_t$.
5. Então ele consegue calcular a assinatura s da mensagem m:

$$s = (\prod_{i=1}^{t} sig_i) sig^{-t} \pmod{n}$$

Ataque de Moore. O Ataque de Moore [Moo88] é uma simplificação do ataque de Davida, necessitando, com a cooperação da vítima, de apenas uma assinatura para gerar uma falsificação de uma mensagem escolhida m.

1. Seja m a mensagem que se quer falsificar a assinatura s.
2. Seja s_1 um valor aleatório.
3. Calcule m_1, tal que, (m_1, s_1) seja um par $(mensagem, assinatura)$ válido:

$$m_1 = s_1^e \pmod{n}$$

4. Obtenha (com a cooperação da vítima) a assinatura de $m_1 m$.

$$s_2 = S(m_1 m) = (m_1 m)^d \pmod{n}$$

5. Calcule $s_1^{-1} s_2$ para obter a assinatura s da mensagem m:

$$s = s_1^{-1} s_2 = s_1^{-1} (m_1 m)^d = s_1^{-1} s_1 m^d = m^d \pmod{n}$$

6. Note que se s_1^{-1} não existir, então s_1 possui pelo menos um fator em comum com o módulo n, e como $n = p \cdot q$ então $mdc(s_1, n) = p$ ou $mdc(s_1, n) = q$, que resulta em uma quebra total do protocolo. Esse caso ocorre com uma probabilidade negligenciável.

3.3 Protocolo de assinatura RSA ISO/IEC 9796-2

O padrão de assinatura ISO/IEC 9796-2 foi estabelecido em 2002 como o padrão oficial da ISO para assinatura digital. Esse protocolo foi criado com a abordagem *ad hoc*, isto é, com o objetivo de evitar certos tipos de ataques (i.e os ataques da seção anterior). O que normalmente acontece com protocolos *ad hoc* é que eles eventualmente são atacados por algum tipo de ataque que não havia sido previsto durante a fase de criação do protocolo, o que muitas vezes resulta em uma *quebra* do protocolo.

3.3.1 O Padrão ISO/IEC 9796-2

O padrão define uma função de *encoding* definida a seguir

$$\mu(m) = 6A_{16} \parallel m[1] \parallel H(m) \parallel BC_{16}$$

depois basta aplicarmos a função RSA para obtermos a assinatura desejada

$$\sigma = \mu(m)^d \pmod{n}$$

A função H é uma função de *hash* definida como $H : \{0,1\}^* \to \{0,1\}^{k_h}$. $m[1]$ denota os primeiros (mais a esquerda) $k - k_h - 16$ bits de m, e $m[2]$ são os bits restantes. k é o tamanho em bits da mensagem m. O valor k_h é o tamanho (em bits) da função de *hash* utilizada (não há um tamanho pré-definido).

3.3.2 Ataque ao Padrão

No artigo de Coron et al. de 2009 [CNTW09, pg. 5] é descrito um ataque prático ao padrão ISO/IEC 9796-2 onde se obtém uma falsificação existencial, utilizando um ataque de mensagem escolhida. Esse resultado obtido é uma extensão do resultado de Coron et al. de 1999 [CNS99].

Esse ataque se baseia no ataque de Desmedt e Odlyzko de 1985 [DO85], que será descrito sucintamente a seguir:

1. Escolha um limite B, e seja $\mathcal{B} = \{p_1, ..., p_\ell\}$ o conjunto de todos os números primos menores que B.

2. Represente um $\mu(m_j)$ como uma combinação multiplicativa dos outros $\mu(m_i)$ resolvendo um sistema linear representado pelo vetor de expoentes de $\mu(m_j)$ em relação aos primos em $\mathcal{B}$.

3. Obtenha as assinaturas de m_i para $i \neq j$ e falsifique a assinatura de m_j.

Como a função $\mu(m)$ gera uma saída tão grande quanto n (o módulo do RSA) esse ataque fica inviável para ser realizado nesse protocolo. Em [CNTW09, pg. 5] é apresentada uma técnica que torna o ataque aplicável ao protocolo, mesmo sendo o tamanho de $\mu(m)$ igual ao tamanho de n.

O ISO/IEC 9796-2 é um exemplo de protocolo que não possui prova de segurança, e foi atacado de uma forma prática [CNTW09]. Isso apenas reforça a importância de provas *eficientes* de segurança para protocolos em geral.

3.4 Segurança demonstrável

Depois do início da era da criptografia de chave pública, que começou com o artigo de Diffie e Hellman [DH76], surgiram vários novos problemas criptográficos, e vários protocolos de criptografia e assinatura. Inicialmente, esses protocolos não tinham nenhum tipo de prova de sua segurança, e

eram considerados seguros quando resistiam a vários ataques durante alguns anos. O que normalmente ocorria é que esses protocolos eram quebrados depois de alguns anos [CR88] de tentativas. Logo depois começaram a criar as chamadas *provas de segurança* utilizando técnicas da *teoria da complexidade computacional*, onde um problema reconhecidamente difícil era *reduzido* polinomialmente, através de algoritmos de redução, para o problema de se quebrar o protocolo. Se essa redução fosse possível, e fosse polinomial, então poderíamos afirmar que o problema de se quebrar o protocolo é pelo menos tão difícil quanto o problema reconhecidamente difícil (por um fator polinomial).

Segurança exata

O termo *segurança exata* foi criado por Bellare e Rogaway em seu artigo de 1993 [BR93] para descrever uma redução de segurança que tenha um valor prático, isto é, uma redução em que partindo de um *ataque* podemos criar um algoritmo que resolva o problema difícil com praticamente a mesma probabilidade, e praticamente no mesmo tempo.

Muitas vezes, para se alcançar esse nível de segurança sem prejudicar a eficiência do protocolo é necessário utilizar alguns modelos que foram criados justamente para lidar com provas de segurança de protocolos eficientes. Esses modelos transformam objetos reais em objetos *ideais*, como por exemplo o *modelo do oráculo aleatório*, onde a função de hash é transformada em uma função uniforme e aleatória.

Modelo do Oráculo Aleatório

O modelo do oráculo aleatório foi introduzido por Bellare e Rogaway em [BR93] com o objetivo de abstrair o protocolo, assumindo a distribuição da função de hash utilizada no protocolo como sendo uma distribuição uniforme, o que permite um foco maior no protocolo como um todo, e assim encontrar reduções de segurança mais eficientes.

Nesse modelo, assume-se que todas as partes envolvidas no protocolo tem acesso a um oráculo aleatório e uniforme. Para se desenvolver um protocolo nesse modelo deve-se seguir os seguintes passos:

1. Assuma que existe um problema Π.
2. Encontre uma definição formal para Π no modelo em que todos os participantes compartilhem um oráculo aleatório R.
3. Desenvolva um protocolo P para Π nesse modelo do oráculo aleatório.
4. Prove que P satisfaz a definição de Π.
5. Substitua os acessos ao oráculo R por uma função de hash H.

Acredita-se [BR93] que esse modelo é uma ferramenta útil para o desenvolvimento de protocolos seguros e ao mesmo tempo eficientes.

Existe uma dúvida grande em relação à validade do modelo, pois existem protocolos que foram provados seguros no modelo do oráculo aleatório, mas nenhuma implementação desses protocolos se mostrou segura na prática [CGH04].

Optimal Asymmetric Encryption Padding (OAEP)

Apesar de ser um protocolo de cifração, e não de assinatura, que é o foco deste trabalho, o OAEP é muito interessante, pois foi um dos primeiros protocolos criados com uma prova eficiente de segurança[1] no modelo do oráculo aleatório.

É um protocolo probabilístico de cifração, que dada uma função f *trapdoor one-way*[2] de k bits para k bits e uma função de hash H, encripta qualquer sequência de bits x, de comprimento um pouco menor do que k *bits* como $f(r_x)$, onde r_x é um *padding*[3] probabiliístico de x, que depende da função de hash H. O protocolo foi provado IND-CCA2[4] seguro [BR94], assumindo que a função de hash H seja aleatória e uniforme (i.e modelo do oráculo aleatório).

Descrição do Protocolo

O OAEP é um protocolo probabilístico de cifração, o que significa que se uma mesma mensagem m for cifrada mais de uma vez utilizando o OAEP e com a mesma chave, o resultado não será o mesmo, exceto por uma probabilidade muito pequena.

O protocolo é muito eficiente, e é comprovadamente *IND-CCA2* seguro contra ataques de texto cifrado escolhido. As definições de segurança atuais utilizadas para demonstrações de segurança de protocolos podem ser encontradas em [Mao03].

No artigo original [BR94], o protocolo é descrito como sendo possível substituir f por qualquer função *trapdoor one-way*, mas aqui consideraremos apenas o caso em que f é a função RSA.

O protocolo de encriptação é definido a seguir:

1. Seja n o módulo RSA, tal que $n = pq$.
2. Sejam k_0 e k_1 dois parâmetros de segurança.
3. Seja M a mensagem a ser encriptada, tal que $m = |M| = |n| - k_0 - k_1$
4. Sejam H e G duas funções de hash, tal que $H : \{0,1\}^{m+k_1} \rightarrow \{0,1\}^{k_0}$ e $G : \{0,1\}^{k_0} \rightarrow \{0,1\}^{m+k_1}$.

[1] Depois de nove anos de uso do OAEP em muitas aplicações, Shoup [Sho02] mostrou que a prova do OAEP continha uma falha grave que invalidava a sua segurança. No mesmo artigo, Shoup mostrou uma forma de recuperar (parcialmente) a segurança do protocolo no caso OAEP-RSA. Fujisaki et al. [FOPS04] estendeu o trabalho de Shoup e recuperou totalmente a segurança do protocolo OAEP-RSA.

[2] função *alçapão* de uma via

[3] *padding* é uma sequência de bits concatenada à mensagem.

[4] *Indistinguishable adaptive chosen-ciphertext attack* ou *IND-CCA2* é a definição de segurança mais forte para protocolos de cifração.

5. Selecionar um número aleatório r, tal que $|r| = k_0$.
6. Calcular $s = m \parallel 0^{k_1} \oplus G(r)$ e $t = r \oplus H(s)$.
7. O texto cifrado é $c = f(s \parallel t)$.

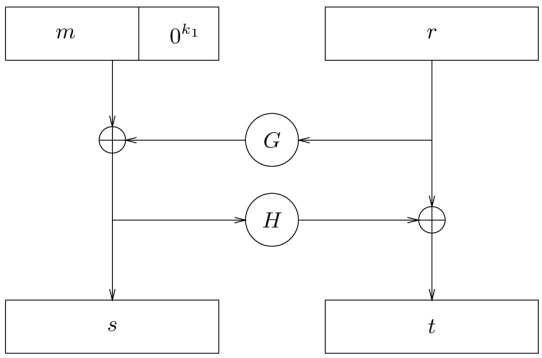

Figura 3.1: O protocolo OAEP

Para decriptografar c, deve ser realizado os seguintes passos:

1. Calcular $s \parallel t = f^{-1}(c)$.
2. Calcular $r = t \oplus H(s)$ e depois $m \parallel 0^{k_1} = s \oplus G(r)$.
3. Se os últimos k_1 bits de $m \parallel 0_1^k$ não forem 0 então rejeita a mensagem.
4. Senão, m é a mensagem original.

Prova eficiente de segurança

A técnica utilizada para se provar a segurança de protocolos criptográficos é realizar a redução de um problema reconhecidamente difícil para o problema de quebrar o protocolo.

No caso do OAEP-RSA é realizada a redução do problema do RSA (que se acredita ser intratável) para o problema de quebrar o protocolo OAEP-RSA. É mostrado que se um adversário $\mathcal{A}$ consegue quebrar o protocolo em tempo polinomial, com uma probabilidade ϵ, então $\mathcal{A}$ consegue quebrar o RSA em tempo polinomial com uma probabilidade ϵ' (relacionada a ϵ).

A prova detalhada de segurança é descrita em [Mao03].

3.5 Protocolo de assinatura PSS

Em aplicações práticas, a *eficiência* da redução da demonstração de segurança de um protocolo deve ser considerada importante. Uma demonstração de segurança considerada *eficiente*, é uma

em que, na existência de um algoritmo A, que *quebre* o protocolo de assinatura em questão, com probabilidade ϵ, permite a construção de um outro algoritmo I, que utilizando o algoritmo A, consegue resolver um problema reconhecidamente difícil (e.g RSA) com um probabilidade $\epsilon' \simeq \epsilon$. Nesse caso, dizemos que o protocolo de assinatura é *pelo menos* tão seguro quanto o problema reconhecidamente difícil em questão, mas no caso contrário, em que a redução não é eficiente, a garantia de segurança do protocolo não será muito boa, e na prática, teríamos que aumentar o tamanho dos parâmetros de segurança para obter uma segurança semelhante, o que reduz muito a eficiência do protocolo.

O Protocolo de assinatura digital PSS foi proposto por Bellare e Rogaway em [BR96], e foi o primeiro protocolo de assinatura eficiente a possuir um prova eficiente de segurança no modelo do oráculo aleatório.

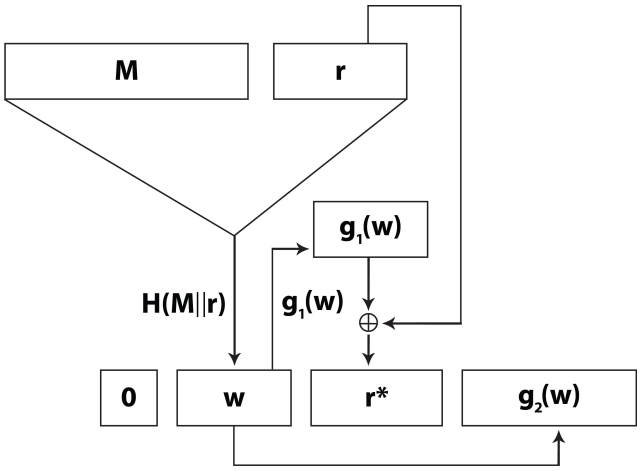

Figura 3.2: Protocolo de Assinatura PSS

3.5.1 Descrição do PSS

O Protocolo PSS$[k_0, k_1]$ = *(GenPSS, SignPSS, VerifyPSS)* é parametrizado por k_0 e k_1, que são números entre 1 e k, e que satisfaçam a restrição $k_0 + k_1 \leq k - 1$. O valor k representa o tamanho em *bits* do módulo n.

O algoritmo de geração de chaves *GenPSS* é descrito como: recebe como entrada 1^k e executa *RSA*(1^k) para obter o vetor (N, e, d), a saída do algoritmo é *(pk,sk)*, onde $pk = (N, e)$ e $sk = (N, d)$.

Os algoritmos de assinatura *SignPSS*, e de verificação *VerifyPSS* utilizam duas funções de hash. A primeira, chamada de compressora é definida como $h : \{0,1\}^* \to \{0,1\}^{k_1}$, e a segunda, chamada de geradora é definida como $g : \{0,1\}^{k_1} \to \{0,1\}^{k-k_1-1}$. A prova de segurança assume que essas funções de hash são *ideias* (i.e possuem distribuição uniforme).

Agora descreveremos os algoritmos *SignPSS* e *VerifyPSS*:

Algoritmo 3.5.1 (SignPSS(M))**.** Recebe a mensagem M como entrada.

1. $r \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$
2. $w \leftarrow h(M \parallel r)$
3. $r^* \leftarrow g_1(w) \oplus r$
4. $y \leftarrow 0 \parallel w \parallel r^* \parallel g_2(w)$
5. **retorna** $y^d \bmod N$

Algoritmo 3.5.2 (VerifyPSS(M, x))**.** Recebe como entrada a mensagem M e a suposta assinatura x

1. $y \leftarrow x^e \bmod N$
2. represente y como $b \parallel w \parallel r^* \parallel \gamma$ (isto é, o primeiro bit de y é b, e assim por diante.)
3. $r \leftarrow r^* \oplus g_1(w)$
4. **se** $h(M \parallel r) = w$ e $g_2(w) = \gamma$ e $b = 0$ então retorna **verdadeiro** senão retorna **falso**

O passo $r \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$ indica que o valor r (de k_0 bits) é escolhido aleatoriamente.

A função g_1 é a função que recebe de entrada $w \in \{0,1\}^{k_1}$ e retorna os primeiros k bits de $g(w)$, e a função g_2 é a função que recebe de entrada $w \in \{0,1\}^{k_1}$ e retorna os $k - k_0 - k_1 - 1$ bits de restantes de $g(w)$.

3.5.2 Segurança do PSS

Foi realizada uma redução eficiente da segurança do protocolo PSS para o problema do RSA [BR96].

Suponha que o problema do RSA seja (t', ϵ')-seguro. Então para qualquer quantidade de q_{sig}, q_{hash}, o protocolo PSS$[k_0, k_1]$ é $(t, q_{sig}, q_{hash}, \epsilon)$-seguro, onde

$$t = t' - [q_{sig} + q_{hash} + 1] \cdot k_0 \cdot \Theta(k^3), \ \ e$$

$$\epsilon = \epsilon' + [3(q_{sig} + q_{hash})^2] \cdot (2^{-k_0} + 2^{-k_1})$$

As variáveis q_{sig} e q_{hash} correspondem a quantidade de consultas aos oráculos de assinatura e de hash, respectivamente (normalmente são considerados $q_{sig} = 2^{30}$ e $q_{hash} = 2^{60}$ [BR96]). As variáveis t e t' representam o tempo de execução necessário para *quebrar* o protocolo PSS$[k_0, k_1]$ e o RSA respectivamente. E por último, as variáveis ϵ e ϵ' representam a probabilidade que um algoritmo *quebre* o protocolo PSS$[k_0, k_1]$ e o RSA, respectivamente.

3.5.3 Nova Prova de Segurança do PSS

Pelo resultado obtido na prova original do PSS [BR96], podemos concluir que, para termos uma redução eficiente de segurança (i.e $\epsilon \simeq \epsilon'$), precisamos que $(q_{sig} + q_{hash})^2 \cdot (2^{-k_0} + 2^{-k_1}) < \epsilon'$, isto é, precisamos que $k_0 \geq k_{min}$ e $k_1 \geq k_{min}$, onde

$$k_{min} = 2 \cdot \log_2(q_{hash} + q_{sig}) + \log_2 \frac{1}{\epsilon'}$$

Se considerarmos $q_{sig} = 2^{30}$, $q_{hash} = 2^{60}$ e $\epsilon' = 2^{-60}$, veremos que para a redução de segurança original do PSS ser eficiente, é necessário que o tamanho de k_0 e k_1 sejam maiores do que k_{min}, isto é, maiores do que 180 bits.

Em [Cor02] é demonstrado que o protocolo PSS pode ter uma redução eficiente para valores de k_0 bem menores do que os considerados inicialmente.

Suponha que o problema do RSA seja (t', ϵ')-seguro. Então para qualquer quantidade de q_{sig}, q_{hash} o protocolo PSS$[k_0, k_1]$ é $(t, q_{sig}, q_{hash}, \epsilon)$-seguro, onde

$$t = t' - q_{sig} + q_{hash} \cdot k_1 \cdot \mathcal{O}(k^3), \ \ e$$

$$\epsilon = \epsilon' \cdot (1 + 6 \cdot q_{sig} \cdot 2^{-k_0}) + 2 \cdot (q_{sig} + q_{hash})^2 \cdot 2^{-k_1}$$

Com essa nova redução de segurança, podemos ver que o tamanho de k_0 , isto é, o tamanho da variável aleatória r pode ser $k_0 = \log_2 q_{sig}$, e ainda assim obtemos uma redução eficiente (note que $k_1 > k_{min}$ ainda se aplica).

No caso considerado ($q_{sig} = 2^{30}$), será necessário gerar um número aleatório de apenas 30 bits, e se obterá o mesmo nível de segurança da prova anterior (i.e uma redução eficiente, onde $\epsilon \simeq \epsilon'$). Também é mostrado que se $k_0 < \log_2 q_{sig}$, então a redução de segurança não pode ser eficiente, o que mostra que o resultado obtido em [Cor02] é ótimo.

Um resultado importante mostrado em [Cor02], é que é *impossível* obter uma redução eficiente da *inversão* do RSA para a *quebra* de protocolos do tipo *hash-and-sign* (e.g FDH) com assinaturas únicas para cada mensagem (i.e uma mensagem m sempre irá gerar a mesma assinatura s). Esse teorema [Cor02, teorema 5] mostra que estamos *limitados* a uma redução ineficiente se utilizarmos a função RSA em um protocolo de assinatura sem randomização (i.e com assinaturas únicas para cada mensagem), o que nos fez buscar alternativas, como a função *Rabin*.

3.6 Variante do PSS com prova eficiente de segurança sem Randomização

O PSS utiliza um valor aleatório r para possiblitar uma redução eficiente de segurança para um problema reconhecidamente difícil, no caso, RSA. Coron mostrou em [Cor02] que é possível diminuir o tamanho desse parâmetro de segurança (para cerca de 30 bits) e ainda assim obter uma redução eficiente de segurança.

Será apresentado na seção 3.6.2 uma variação do protocolo PSS, que não necessita aleatorizar a entrada para obter uma redução eficiente de segurança, tornando o protocolo determinístico, e mais eficiente. Esse protocolo não contradiz a afirmação de Coron [Cor02] de que é *impossível* obter uma redução eficiente da *inversão* do RSA para a *quebra* de protocolos *hash-and-sign* de assinatura única, pois no protocolo que será apresentado, existem exatamente *duas* assinaturas para cada mensagem. Apesar de existirem *duas* assinaturas válidas para cada mensagem, apenas uma delas será gerada pelo protocolo, mesmo se a mensagem for assinada múltiplas vezes. Isso é realizado por um algoritmo determinístico, que apesar de parecer, não necessita que o protocolo mantenha estado para saber qual das assinaturas foi gerada anteriormente.

3.6.1 Permutações *Claw-Free*

O protocolo PSS [BR96] possui uma prova (redução) de segurança eficiente quando baseado na função *RSA*, mas não quando baseado em uma função de permutação *trapdoor genérica*, o que ocorre também com vários outros protocolos [Cor02]. Como foi notado em [DR02], o que torna a redução de segurança de protocolos (e.g PSS) eficiente quando baseados em funções de permutação *trapdoor* específicas (e.g RSA) não é *apenas* a propriedade de *trapdoor* da função, e sim quando a função de permutação *trapdoor* em questão vêm de uma família de funções de permutação *claw-free* (como no caso do RSA e Rabin). Nesse caso, a redução de segurança fica *muito* mais eficiente.

Informalmente, um par de permutações *claw-free* (f_0, f_1) possui a propriedade de que cada uma das funções f_0 e f_1 são, individualmente, funções de permutação sob um mesmo domínio. Outra propriedade é que sem a *trapdoor* de uma das funções, fica muito difícil de encontrar uma *claw*, isto é, x_0 e x_1, tal que, $f_0(x_0) = f_1(x_1)$.

Definição 3.6.1. Uma família de permutações *claw-free* é uma tupla de algorítmos probabilísticos de tempo polinomial (cf-Gen, F, G, F^{-1}, G^{-1}) tal que:

- cf-Gen gera um índice aleatório i e um número *trapdoor* td.
- $F(i, \cdot)$ e $G(i, \cdot)$ são permutações sob o mesmo domínio D_i.
- Existe um algoritmo que recebe como entrada o índice i e retorna um valor em D_i, distribuído uniformemente.
- se (i, td) foi o resultado de cf-Gen, então $F^{-1}(td, \cdot)$ é o inverso de $F(i, \cdot)$, e $G^{-1}(td, \cdot)$ é o inverso de $G(i, \cdot)$.

É dito que um algoritmo $\mathcal{A}$ (t, ϵ)-*quebra* uma família de permutações *claw-free* se $\mathcal{A}$ executa (no máximo) em tempo t, e retorna uma *claw* (x_0 e x_1, tal que, $f_0(x_0) = f_1(x_1)$) com probabilidade maior que ϵ.

Uma permutação *claw-free* é considerada (t, ϵ)-*segura* se nenhum algoritmo consegue (t, ϵ) quebrá-la.

3.6.2 Descrição do Protocolo

Como já foi mencionado, Coron [Cor02, teorema 5] afirma que é *impossível* obter reduções de segurança eficientes da função RSA para a *quebra* de protocolos de assinatura do tipo *hash-and-sign* que possuem assinaturas únicas para cada mensagem. Devido a esse resultado, o protocolo apresentado possui *duas* assinaturas diferentes para cada mensagem, mas apenas uma assinatura será gerada para cada mensagem. O esquema apresentado pode ser considerado como PSS$[k_0 = 1, k_1]$.

O protocolo [KW03] assume uma função de permutação *trapdoor one-way* induzida por uma permutação *claw-free* (f, g). H é um oráculo aleatório, que mapeia cadeias de bit de tamanho aleatório para cadeias de bit de um tamanho determinado.

A seguir serão descritos os três algoritmos que compõem o protocolo, $KeyGen$ para a geração das chaves, $Sign_{SK}(m)$ para realizar a assinatura da mensagem m utilizando a chave secreta SK, e $Verify_{PK}(m, \sigma)$ para verificar a validade da assinatura σ sobre a mensagem m, utilizando a chave pública PK.

Algoritmo 3.6.2 (KeyGen)**.** Algoritmo de geração de chaves.

1. executa cf-Gen para obter (f, g) e a informação *trapdoor* td.
2. A chave pública é $PK = f$, e a chave secreta é a *trapdoor* td.

Algoritmo 3.6.3 ($Sign_{SK}(M)$)**.** Algoritmo de assinatura.

1. verifica se a mensagem m já foi assinada anteriormente.
2. se sim, **retorna** a assinatura gerada anteriormente.
3. senão, escolhe um bit aleatório b e **retorna** $\sigma = f^{-1}(H(b \parallel m))$.

Algoritmo 3.6.4 ($Verify_{PK}(m, \sigma)$)**.** Algoritmo de verificação da assinatura.

1. se $f(\sigma) = H(0 \parallel m)$ ou $f(\sigma) = H(1 \parallel m)$ **retorna** verdadeiro.
2. senão **retorna** falso.

Uma maneira de evitar a necessidade de manter estado para armazenar todos os pares (m, σ) gerados, é com o bit b sendo uma função aleatória de m. Como estamos no modelo do oráculo aleatório, uma solução simples seria fazer $b = H'(SK \parallel m)$, onde $H' : \{0,1\}^* \to \{0,1\}$ é um oráculo aleatório.

3.6.3 Prova de Segurança

Teorema 3.6.5. *Se a permutação claw-free do protocolo é (t', ϵ')-segura, e o tempo para se computar f ou g é, no máximo, t_f, então o protocolo acima é $(t, q_{hash}, q_{sig}, \epsilon)$-seguro (no modelo do oráculo aleatório), onde:*

$$t \leq t' - (q_{hash} + q_{sig}) \cdot t_f$$

$$\epsilon \geq 2\epsilon'$$

Demonstração. Se existir um algoritmo A, que faça (q_{hash}, q_{sig}) consultas aos oráculo de *hash* e de assinatura respectivamente, e consiga obter uma falsificação válida de um par (m, σ), com probabilidade pelo menos ϵ, então é possível construir um algoritmo I, que executa em tempo (no máximo) t', e que encontre uma *claw* com probabilidade pelo menos ϵ'.

O algoritmo I *simula* os oráculos de *hash* e de *assinatura*, tentando se passar por oráculos de *hash* e de *assinatura* verdadeiros. O algoritmo A interage com o algoritmo I, e tenta obter uma falsificação válida (m, σ).

O algoritmo I recebe como entrada o par (f, g), e seu objetivo é obter uma *claw* (i.e x_0, x_1 tal que $f(x_0) = g(x_1)$). Ele então define $PK = f$, e executa A com o parâmetro de entrada PK. □

Estado global. Toda vez que uma consulta ao oráculo de *hash* $H(b \parallel m)$, ou ao oráculo de assinatura $Sign_{SK}(m)$ for feita para uma mensagem *nova* m, o algoritmo I escolhe um bit aleatório $b_m \in \{0, 1\}$.

Consulta ao Oráculo de *Hash*. I mantém uma lista H_L de tuplas de uma forma que será descrita a seguir. Em resposta a uma consulta de *hash* $H(b \parallel m)$ o algoritmo I verifica se existe em H_L uma tupla (b, m, x, y), se existir, então I retorna y, senão, I escolhe um valor aleatório x (no domínio apropriado), e se $b = b_m$ então I retorna $y = f(x)$, senão retorna $y = g(x)$. Em todos os casos, I armazena a tupla (b, m, x, y) em H_L.

Consulta ao Oráculo de Assinatura. Quando o algoritmo A pede a assinatura de uma mensagem m para o algoritmo I, o algoritmo I precisa responder a essa requisição de uma forma que o algoritmo A acredite que esteja interagindo com o verdadeiro oráculo de assinatura, e não com um simulador. O algoritmo I não pode simplesmente *assinar* a mensagem m, uma vez que o algoritmo I não conhece a chave secreta SK. O que o algoritmo I faz é computar $y = H(b_m \parallel m)$, e encontrar a tupla (b_m, m, x, y) em H_L (note que essa tupla sempre irá existir, pois será uma consulta ao oráculo de *hash* que acabou de ser feita). I retorna $\sigma = x$ que é uma assinatura válida de m, pois $f(\sigma) = H(b_m \parallel m)$.

A simulação que o algoritmo I proporciona é perfeita, pois todas as requisições feitas por A ao oráculo de assinatura são respondidas com assinatura válidas. Em algum momento, o algoritmo A termina sua execução e retorna uma falsificação (m', σ') (uma restrição feita a A é que não pode ter sido feita uma consulta ao oráculo de assinatura para a mensagem m'). Se $Verify_{PK}(m', \sigma') = 1$ então a falsificação é verdadeira, e existem dois casos possíveis:

Caso 1: $f(\sigma') = H(b_{m'} \parallel m')$. Nesse caso, I não consegue encontrar uma *claw* e aborta.

Caso 2: $f(\sigma') = H(1-b_{m'} \parallel m')$. Nesse caso, I encontra uma *claw* da seguinte maneira: encontra uma tupla $(1-b_{m'}, m', x, y)$ em H_L e retorna a *claw* (σ', x) (note que $g(x) = y = H(1 - b_{m'} \parallel m')$).

Como o algoritmo A não conhece o valor de $b_{m'}$, a falsificação válida de A irá possibilitar que I encontre uma *claw* com probabilidade $\frac{1}{2}$. Então se A retorna uma falsificação válida com probabilidade ϵ, I irá encontrar uma *claw* com probabilidade $\epsilon' = \frac{\epsilon}{2}$.

3.7 Protocolo de Assinatura Rabin

Em 1979 Rabin [Rab79] estendeu o artigo do RSA [RSA78] e criou uma função de permutação *trapdoor* mais simples e mais *eficiente* do que o RSA. Ele conseguiu provar que *inverter* essa função é equivalente ao problema da fatoração.

A vantagem da função Rabin sobre o RSA é que a verificação da assinatura pode ser até cem vezes mais rápida[Ber03]. Uma desvantagem é que Rabin é uma função 1 para 4, e não uma permutação, o que se torna necessário uma forma de *contornar* essa ambiguidade.

3.7.1 Descrição do Protocolo

O protocolo Rabin é descrito a seguir:

1. Escolher dois números primos grandes p e q tal que $|p| = |q|$, e calcular $n = p \cdot q$.
2. A chave pública é n, e a chave secreta é (p, q).
3. Seja $H : \{0,1\}^* \rightarrow \{0,1\}^n$ uma função de *hash* resistente à colisão.
4. Para assinar uma mensagem m, calcular um número aleatório r, e calcular $h = H(m \parallel r)$.
5. Se $h \notin (\mathbb{Z}_n^*)^2$ voltar ao passo anterior.
6. $\sigma \stackrel{R}{=} \sqrt{h} \mod n$ (σ recebe uma raiz quadrada aleatória).
7. A tupla (m, r, σ) é enviada para a parte que irá verificar a validade da assinatura.
8. Para verificar se a tupla (m, r, σ) é válida, calcular $H(m \parallel r) \stackrel{?}{=} \sigma^2 \mod n$. Se a igualdade for verdadeira a assinatura é válida, caso contrário, a assinatura é falsa.

No passo 5, se h não for um *resíduo quadrático* módulo n precisamos escolher um outro valor aleatório r até que h seja um *resíduo quadrático* módulo n, e possua raiz quadrada módulo n [Ter08, cap. 2].

Para sabermos se $h \in (\mathbb{Z}_n^*)^2$ precisamos calcular o símbolo de Jacobi [Ter08] de h, se $\mathbf{J}_n(h) = 1$ então $h \in (\mathbb{Z}_n^*)^2$, caso contrário $h \notin (\mathbb{Z}_n^*)^2$.

No passo 6, σ recebe *aleatóriamente* uma das *quatro* raizes quadradas de h módulo n. O motivo de h possuir *quatro* raízes quadrada módulo n se deve ao fato de $n = p \cdot q$ ser o produto de dois números primos [Ter08, cap. 2].

O número aleatório r é necessário pois caso $H(m) \notin (\mathbb{Z}_n^*)^2$ não poderíamos assinar a mensagem m, então a função do r é aleatorizar a entrada de H, para tentar gerar um valor $h \in (\mathbb{Z}_n^*)^2$. No artigo original [Rab79], Rabin conjecturou $|r| \geq 60$ bits, pois apenas 1/4 dos números que $\in \mathbb{Z}_n$ são resíduos quadráticos, e 60 bits *praticamente* garante a assinatura de todas as mensagens. A probabilidade de não ser possível gerar uma assinatura quando $r = 60$ é $< 2^{-58}$.

3.7.2 Raiz quadrada principal

No passo 1 da descrição do protocolo, devemos escolher p e q tal que $p \equiv q \equiv 3 \mod 4$. Isso transforma a função Rabin em uma permutação, e ainda facilita os cálculos da raizes quadradas no módulo n.

A raiz quadrada principal de $h \mod n$ é a *única* raiz quadrada de h, que também é um *resíduo quadrático* de n.

A função é transformada em uma função $Rabin : (\mathbb{Z}_n^*)^2 \to (\mathbb{Z}_n^*)^2$, onde $Rabin(x) = \sqrt{x} \mod n$. Dessa forma, evita-se o risco de retornar duas raizes quadradas diferentes (exceto com uma probabilidade muito pequena) de uma mesma mensagem, o que poderia levar à fatoração do módulo n.

Como o domínio da função Rabin é $(\mathbb{Z}_n^*)^2$ podemos aplicar a função apenas às mensagens $m \in (\mathbb{Z}_n^*)^2$, o que torna *impossível* eliminar a variável aleatória r sem correr o risco de existir mensagens que não poderão ser assinadas (3/4 de todas as mensagens).

Isso torna a função Rabin (sem a randomização r) inadequada para a maioria das aplicações práticas, pois possibilitaria a assinatura de apenas uma fração de todas as mensagens possíveis, forçando-nos a recorrer a uma variação da função Rabin, a função Rabin-Williams, que será apresentada no próximo capítulo.

Capítulo 4

O protocolo *Rabin-Williams*

Neste capítulo será apresentado o protocolo de estudo deste trabalho, bem como nossos resultados referentes à prova de segurança do protocolo.

A idéia por trás do protocolo de assinatura *Rabin-Williams* é estender o domínio das mensagens que podem ser assinadas. Isso é possível quando selecionamos os valores p e q de uma forma especial. Uma assinatura *Rabin-Williams* de uma mensagem m é uma tripla (e, f, s) chamada de *raiz quadrada modificada*[1] de $H(m) \bmod pq$, tal que $efs^2 \equiv H(m) \pmod{pq}$. Diferentemente da assinatura *Rabin*, todas as mensagens possuem uma assinatura válida. O motivo de estudarmos o protocolo *Rabin-Williams* neste trabalho é que ele possui diversas características interessantes, que o tornam o protocolo mais eficiente conhecido em termos de velocidade de verificação de assinaturas. Vamos citar algumas características que o tornam tão especial.

Extensão do domínio de mensagens assináveis. Em contraste com a assinatura *Rabin* [Rab79], que permite assinar apenas 1/4 das mensagens do domínio de mensagens, tornando-se necessário concatenar um valor aleatório à mensagem antes de assinar, na esperança de se obter uma mensagem que seja *assinável*, a assinatura *Rabin-Williams* permite que toda mensagem do domínio de mensagens seja assinável, mesmo sem precisar concatenar qualquer bit aleatório à mensagem, o que a torna mais indicada ao uso em aplicações *reais*.

Função de hash. O protocolo *Rabin-Williams* não assina diretamente uma mensagem m. Ele assina um valor $h = H(m)$, que é o resultado da mensagem m aplicada a uma função de hash $H : \{0,1\}^* \to \{0,1\}^K$. Basicamente, o papel dessa função de hash é reduzir o tamanho da mensagem m para um número de bits K especificado. A instanciação de funções de hash com domínio e imagem de tamanhos arbitrários é discutida por Bellare e Rogaway em [BR93, BR96]. A função de hash é essencial para a segurança do protocolo, e se não utilizarmos uma função de hash, o protocolo pode ser facilmente quebrado por um ataque de mensagem escolhida, o que resultaria em uma falsificação existencial. Note que o protocolo de assinatura *Rabin* do artigo original [Rab79] utiliza uma função de hash e continua sem ser quebrado até hoje, enquanto que o protocolo *RSA*

[1] do termo em inglês *tweaked square root*

[RSA78] não utilizava função de hash e foi quebrado trivialmente pelo ataque de Davida, como descrito na seção 3.2.4. O hash também é uma ferramenta importante para conseguirmos assinar mensagens de tamanho arbitrário sem nos preocuparmos em como lidar com mensagens que excedem o tamanho do módulo n utilizado.

Expoente 2. O protocolo *Rabin* foi o primeiro a considerar o expoente 2 como chave pública, ao invés de expoentes grandes e aleatórios escolhidos pelo usuário do protocolo. Bernstein [Ber08] alega que isso torna o processo de verificação de assinatura até cem vezes mais rápido do que o *RSA*, pelo fato de a verificação necessitar de uma simples multiplicação modular. O motivo de muitos optarem por outros expoentes ao invés do 2 é acreditarem de que são necessários cálculos de símbolos de Jacobi no processo de assinatura, o que não é necessariamente verdade se usarmos uma técnica mostrada no artigo [Ber08], e que será utilizada no nosso protocolo *Rabin-Williams*. As provas de segurança necessitam de um esforço a mais pois uma distribuição uniforme de $s \bmod n$ não corresponde a uma distribuição uniforme de $s^2 \bmod n$ pelo fato dessa função ser $4 \to 1$ e não $1 \to 1$.

Assinaturas determinísticas. A mesma mensagem sempre irá gerar a mesma assinatura. Sempre houve uma crença entre os pesquisadores da área de que para obtermos uma prova eficiente de segurança para um protocolo de assinatura deveríamos *randomizar* a mensagem de alguma forma antes de aplicarmos o hash a essa mensagem, então era escolhido um valor aleatório r de tamanho B bits, que era concatenado à mensagem m antes de ser aplicado o hash. Inicialmente acreditava-se que esse B deveria ser grande, 180 bits no caso do *PSS* [BR96]. Posteriormente foi mostrado que níveis altos de segurança são possíveis mesmo com um B pequeno [Cor02, KW03, Ber03]. Bernstein mostrou em 2008 [Ber08] como obter uma prova eficiente de segurança para um $B = 0$ bits apenas no caso em que a assinatura *Rabin-Williams* é escolhida de uma forma uniforme e aleatória, porém determinística (i.e a mesma assinatura é devolvida toda vez que a mesma mensagem for assinada). Isso implica em uma maneira de o protocolo *lembrar* das assinaturas que foram geradas para cada mensagem, sendo necessário manter algum tipo de estado. A forma que Bernstein encontrou para o protocolo não precisar *lembrar* das assinaturas que foram geradas, foi fazer com que o dono da chave privada (i.e assinante) utilizasse de uma outra chave privada (i.e z), que é concatenada a mensagem m a ser assinada, e depois aplicada a uma outra função de hash $H_2 : \{0,1\}^* \to \{0,1\}^2$, que gera como saída um valor de 0 a 3, que determina qual *raiz quadrada modificada* deverá ser utilizada como assinatura de m.

4.1 O protocolo de assinatura Rabin-Williams

Nessa seção iremos especificar o protocolo que será analisado neste artigo, que é o protocolo *Principal Rabin-Williams B=0* do artigo de 2008 de Bernstein [Ber08], que será explicado a seguir.

Parâmetros. Os usuários do protocolo compartilham três parâmetros, que serão descritos abaixo.

(i) um inteiro $K \geq 1024$ (vide tabela 4.1).

Tabela 4.1: Recomendação do NIST [BR11] para tamanhos de chave RSA.

Tamanho da chave	Usar até
$1024 < \|n\| < 2048$	Aceitável até 2012
$\|n\| \geq 2048$	Aceitável após 2013

(ii) uma função de hash que mapeia mensagens $H: \{0,1\}^* \to \{0,1\}^K$.

(iii) uma distribuição D de pares de primos p e q, sendo $p \equiv 3 \bmod 8$, $q \equiv 7 \bmod 8$ e $2^K < pq < 2^{K+1}$, tal que $|p| \approx |q|$.

Geração de chaves. A geração das chaves do protocolo baseia-se em encontrar dois números primos aleatórios p e q (com uma distribuição D), tal que $p \equiv 3 \bmod 8$ e $q \equiv 7 \bmod 8$. A chave pública será $n = pq$, e a chave privada será o par (p, q).

Distribuição de assinaturas Sabemos que todo $h = H(m)$ possui quatro raízes quadradas módulo $n = pq$. O protocolo precisa determinar qual das quatro raízes será devolvida como sendo a assinatura de mensagem m. Existem 3 formas utilizadas na literatura.

(i) **Não-estruturado:** O assinante escolhe uma *raiz quadrada modificada* de $h = H(m)$ uniforme e aleatória, independente de qualquer escolha anterior.

(ii) **Principal:** O assinante escolhe a *raiz quadrada principal modificada* de $h = H(m)$.

(iii) **|Principal|:** O assinante escolhe a *raiz quadrada principal modificada* de $h = H(m)$. Se a *raiz quadrada principal modificada* de $h = H(m)$ é (e, f, s), então a *raiz quadrada principal modificada* é $(e, f, min\{s, pq - s\})$. A vantagem é que $min\{s, pq - s\}$ ocupa um bit a menos de espaço do que na distribuição *Principal*.

Assinaturas fixas e assinaturas variáveis Outro ponto importante é estabelecer um comportamento adequado quando uma mesma mensagem m for assinada mais de uma vez. Existem duas escolhas para o protocolo nesse caso.

(i) **Fixa:** Dada uma mesma mensagem m, o assinante devolve sempre a mesma assinatura s.

(ii) **Variável:** Dada uma mesma mensagem m, o assinante devolve uma nova assinatura escolhida uniforme e aleatoriamente, independente das escolhas anteriores.

No caso do protocolo de estudo deste trabalho (*Principal Rabin-Williams B=0*) essa escolha é irrelevante, pois como não há *randomização* na mensagem, sempre obteremos a mesma assinatura para a mesma mensagem. Portanto omitiremos o parâmetro *Fixa* ou *Variável* do nosso protocolo.

Definição 4.1.1 (Assinatura Principal)**.** A assinatura Principal *Rabin-Williams* de uma mensagem m é a tripla (e, f, s), sendo que:

(i) $h = H(m)$.

(ii) $efs^2 \equiv h \pmod{pq}$.

(iii) $e = 1$ se h é um resíduo quadrático módulo q, caso contrário $e = -1$.

(iv) $f = 1$ se eh é um resíduo quadrático módulo p, caso contrário $f = 2$.

(v) s é um resíduo quadrático módulo $n = pq$.

No caso da assinatura |Principal|, o inteiro s estará no intervalo $\{0..(pq-1)/2\}$, portanto a assinatura pode ser codificada em $K + 2$ bits. A assinatura Principal pode ser codificada em $K + 3$ bits. O termo assinatura principal modificada foi criado por Bernstein em 2008 [Ber08]. Bernstein chama de assinatura |Principal| o menor valor entre $\{s, n - s\}$, sendo que a única vantagem é que a assinatura |Principal| ocupa menos espaço do que a assinatura principal. A vantagem das distribuições de assinatura principal e |Principal| sobre a distribuição não-estruturada é que as duas primeiras não necessitam guardar nenhuma informação sobre as mensagens previamente assinadas, e também não necessitam de nenhum tipo de randomização (para $B = 0$), o que torna o protocolo mais eficiente e mais seguro contra ataques específicos.

Algoritmo 4.1.2. [Ber08, p. 19] Assinatura Principal *Rabin-Williams* da mensagem m sob a chave privada (p, q):

1. Calcular $h \leftarrow H(m)$;
2. Calcular $u \leftarrow h^{(q+1)/8} \bmod q$;
3. Se $u^4 - h \bmod q = 0$ então $e \leftarrow 1$, senão $e \leftarrow -1$;
4. Calcular $v \leftarrow (eh)^{(p-3)/8} \bmod p$;
5. Se $v^4(eh)^2 - (eh) \bmod p = 0$ então $f \leftarrow 1$, senão $f \leftarrow 2$;
6. Calcular $w \leftarrow f^{(3q-5)/8} \cdot u \bmod q$;
7. Calcular $x \leftarrow f^{(9p-11)/8} \cdot v^3(eh) \bmod p$;
8. Calcular $y \leftarrow w + q(q^{p-2}(x - w) \bmod p)$;
9. Calcular $s \leftarrow y^2 \bmod pq$;
10. Retorna (e, f, s);

Agora vamos provar que esse algoritmo retorna a *assinatura principal* de uma mensagem m.

Tabela 4.2: Relação entre os parâmetros e, f e o símbolo de *Legendre*

	$\left(\frac{h}{q}\right) = 1$	$\left(\frac{h}{q}\right) = -1$
$\left(\frac{eh}{p}\right) = 1$	$e = 1$ $f = 1$	$e = -1$ $f = 1$
$\left(\frac{eh}{p}\right) = -1$	$e = 1$ $f = 2$	$e = -1$ $f = 2$

Teorema 4.1.3. *O Algoritmo 4.1.2 retorna a* assinatura principal *de m sob a chave privada (p, q).*

Demonstração. A prova será construída de uma forma que seja possível provar todos os itens que determinam a validade de uma *assinatura principal*. Se h é um resíduo quadrático módulo q então $h^{(q-1)/2} = 1$. Calculando

$$u^4 \equiv h^{\left(\frac{q+1}{8}\right)4} \equiv h^{(q+1)/2} \equiv h \cdot h^{(q-1)/2} \bmod q$$

Obtemos $u^4 \equiv h \cdot h^{(q-1)/2} \bmod q$, portanto, se h é um resíduo quadrático módulo q temos que $h^{(q-1)/2} = 1$, e portanto $u^4 \equiv h \bmod q$ e de acordo com o algoritmo atribuímos $e = 1$. Agora, se h não é um resíduo quadrático módulo q temos que $h^{(q-1)/2} = -1$ e $u^4 \equiv -h \bmod q$, e de acordo com o algoritmo atribuímos $e = -1$. Note que de qualquer forma temos que $u^4 \equiv eh \bmod q$, e portanto

$$u^4 \equiv eh \bmod q$$

Pelo item (iv) do teorema 2.3.8 temos que $2^{(q-1)/2} = 1 \bmod q$ pois o teorema diz que 2 é resíduo quadrático módulo q se q for da forma $q \equiv 7 \bmod 8$. Então podemos afirmar que

$$f^{3(q-1)/2} u^4 \equiv eh \bmod q$$

pois não importa o valor de f (1 ou 2); sempre teremos $f^{3(q-1)/2} = 1$.

Expandindo $v^4(eh)^2$ obtemos

$$(eh)^{\left(\frac{p-3}{8}\right)4}(eh)^2 \equiv (eh)^{(p-3)/2}(eh)^2 \equiv (eh)(eh)^{(p-1)/2} \bmod p$$

Se eh é um resíduo quadrático módulo p então $(eh)^{(p-1)/2} = 1$ e $f = 1$, logo $v^4(eh)^2 \equiv eh \bmod p$ e se eh não é um resíduo quadrático módulo p então $(eh)^{(p-1)/2} = 1$ e $f = 2$, logo $v^4(eh)^2 \equiv -eh \bmod p$. Pelo teorema 2.3.8 temos que $2^{(p-1)/2} = -1 \bmod p$, pois $p \equiv 3 \bmod 8$. Então podemos afirmar

$$f^{9(p-1)/2} v^4 (eh)^2 \equiv eh \bmod p$$

pois se $f = 1$ temos que $f^{9(p-1)/2} = 1$ e eh é um resíduo quadrático módulo p, e se $f = 2$ então $f^{9(p-1)/2} = -1$, o que resulta em $-v^4(eh)^2 \equiv -eh \bmod p$, que nada mais é que $v^4(eh)^2 \equiv eh \bmod p$.

No algoritmo, y é definido como $y = w+q(q^{p-2}(x-w) \bmod p)$, e considere $z = q^{p-2}(x-w) \bmod p$. Agora temos que $y = w + qz$, e se reduzirmos o módulo q temos que $y = w \bmod q$. Temos também que $s = y^2$, então fs^2 pode ser representado como $fy^4 \equiv fw^4 \bmod q$. Substituindo w obtemos $f(f^{(3q-5)/8}u)^4 \equiv f^{3(q-1)/2}u^4 \bmod q$. Como demonstramos acima $f^{3(q-1)/2}u^4 \equiv eh \bmod q$, portanto

$$fs^2 \equiv eh \bmod q$$

Da mesma forma que fizemos acima, temos que $y = w + q^{p-1}(x - w) \equiv w + x - w \equiv x \bmod p$. Portanto $fs^2 \equiv fy^4 \equiv fx^4 \bmod p$, e substituindo da mesma forma que fizemos anteriormente, obtemos $fx^4 \equiv f(f^{(9p-11)/8}v^3(eh))^4 \equiv f^{9(p-1)/2}v^4(eh)^{p-3}(eh)^4 \equiv f^{9(p-1)/2}v^4(eh)^2$, e utilizando nosso resultado anterior

$$fs^2 \equiv eh \bmod p$$

Provamos que $fs^2 \equiv eh \bmod p$ e provamos que $fs^2 \equiv eh \bmod q$, portanto temos que

$$fs^2 \equiv eh \bmod pq$$

que é equivalente a dizer

$$efs^2 \equiv h \bmod pq$$

Obviamente s é um resíduo quadrático módulo pq pois $s = y^2 \bmod pq$, logo a tripla (e, f, s) é a *raiz quadrada principal modificada* de m módulo pq, finalizando a demonstração. □

4.2 Método de Bernstein para provas de segurança

Para desenvolver a prova de segurança do protocolo de assinatura *Principal Rabin-Williams* $B = 0$ vamos utilizar um método que foi introduzido por Bernstein em 2008 [Ber08]. O método de Bernstein é baseado em 5 níveis de problemas difíceis, que tentam abstrair a intuição de um adversário que está atacando o protocolo em questão. Bernstein sugere que a melhor forma para se criar uma prova (redução) de segurança é de uma forma estruturada em níveis, onde cada nível expõe uma simples *interface* que é *acessada* por outros níveis. Isso possibilita uma *composição* de algoritmos, que reduzem de um problema reconhecidamente *difícil* para um problema supostamente mais *fácil*, eventualmente reduzindo até o problema de se quebrar o protocolo. Os cinco níveis de problemas difíceis são descritos a seguir:

Inversão cega. É o problema de falsificar uma assinatura de uma mensagem qualquer m, sem nenhuma informação a mais, exceto a chave pública. Equivale ao adversário que tenta gerar uma falsificação tendo conhecimento apenas da chave pública da vítima a ser atacada. É considerado o ataque mais difícil, por ser o que permite menos *flexibilidade* ao adversário. É o **ataque de chave pública** descrito na seção 3.1.1. Normalmente é a primeira opção de ataque de um adversário analisando algum protocolo.

Inversão selecionada com uma assinatura. É o problema de falsificar uma assinatura de uma mensagem qualquer m, podendo utilizar um *oráculo* de assinatura que irá gerar um par (uniforme e aleatório) *(mensagem, assinatura)* válido. Equivale ao adversário que consegue obter um par de *(mensagem, assinatura)* válido, e tenta obter uma falsificação a partir dessa informação. Normalmente é a segunda opção de ataque de um adversário analisando algum protocolo.

Inversão selecionada com várias assinaturas. É o problema de falsificar uma assinatura de uma mensagem qualquer m, podendo utilizar um *oráculo* de assinatura que irá gerar uniforme e aleatoriamente Q pares *(mensagem, assinatura)* válidos. Equivale ao adversário que consegue obter vários pares de *(mensagem, assinatura)* válidos, e tenta obter uma falsificação a partir dessa informação. Sendo o valor Q determinado pelo protocolo. Normalmente é a terceira opção de ataque de um adversário analisando algum protocolo.

Inversão existencial. É o problema de falsificar uma assinatura de uma mensagem m, mas agora podendo escolher alguma mensagem m que seja mais fácil de ser falsificada, podendo utilizar um *oráculo* de assinatura que irá gerar uniforme e aleatoriamente Q pares *(mensagem, assinatura)* válidos. Equivale ao adversário que consegue obter vários pares de *(mensagem, assinatura)* válidos, e tenta obter uma falsificação de uma mensagem mais *fácil* de ser falsificada a partir dessa informação. Este é o *modelo* de ataque que permite uma maior *flexibilidade* por parte do adversário.

Ataques genéricos. Este último nível trata de ataques que exploram fragilidades de algum componente específico do protocolo. Para citar um exemplo, podemos pensar no caso em que algum usuário do protocolo utilize uma chave de 512 bits ao invés de utilizar uma chave do tamanho recomendado (a partir de 2048 bits). Nesse caso o adversário pode utilizar o algoritmo *Number-field sieve* para fatorar (em um tempo viável) o módulo, e obter a chave privada do usuário, podendo falsificar assinaturas para qualquer mensagem que ele quiser. Note que esse fato não é considerado uma fragilidade do protocolo, e sim mal uso por parte do usuário do protocolo. Outro exemplo interessante seria explorar vulnerabilidades da função de *hash* utilizada pelo protocolo, tentando encontrar colisões, e assim gerar falsificações. Neste trabalho não iremos considerar este último nível de *segurança*, pois estamos preocupados somente com a segurança do protocolo como um *todo*, e não com algum componente específico (i.e *hash*).

Essa abordagem com as funções de hash foi *popularizada* por Bellare e Rogaway em 1993 [BR93] sob o nome de *modelo do oráculo aleatório*, onde eles assumem que a função de hash sendo utilizada pelo protocolo gera uma saída uniforme e aleatória. Essa abordagem é muito útil para a construção de provas *genéricas* de segurança, onde o que importa é a segurança do protocolo em si, pois se alguma vulnerabilidade for *descoberta* em algum componente do protocolo, basta substituir esse componente por outro mais seguro, mantendo o protocolo seguro novamente.

4.3 Prova eficiente de segurança

Vamos provar a segurança do protocolo de assinatura *Principal Rabin-Williams* $B = 0$ utilizando o método de Bernstein.

4.3.1 Inversão cega

Suponha que temos uma chave pública $n = pq$ e um inteiro $h' \in \{1, 2, ..., 2^K\}$. Queremos calcular a *raiz quadrada principal modificada* de h' mod n de uma forma eficiente. Sabemos que um método seria fatorar n, mas estamos interessados em métodos mais eficientes.

Definição 4.3.1 (Inversão cega)**.** Suponha que existe um algoritmo $A(pq, h')$ que retorne uma tripla $(e', f', s') \in \{-1, 1\} \times \{1, 2\} \times \{0, 1, 2, ..., pq - 1\}$ tal que $e'f'(s')^2 \equiv h' \pmod{pq}$ com uma probabilidade, que chamaremos de *PrInversaoCega(A)*. Os parâmetros para esse algoritmo A são:

- Uma chave pública pq selecionada aleatoriamente com uma distribuição uniforme;
- Um inteiro $h' \in \{1, 2, ..., 2^K\}$ selecionado aleatoriamente com uma distribuição uniforme.

sendo n independente de h'.

Qualquer algoritmo A de probabilidade 1 pode gerar um algoritmo de probabilidade 1 para falsificar uma assinatura *Principal Rabin-Williams* $B = 0$, dado acesso público a função de *hash* H. O adversário simplesmente escolhe uma mensagem m' da qual ele quer falsificar a assinatura, calcula $h' = H(m')$ utilizando a função pública H, e utiliza o algoritmo A para gerar a raiz quadrada principal de h' mod pq, obtendo a tripla (e', f', s') que é a assinatura válida de m'.

Redução eficiente do problema da fatoração. Vamos apresentar abaixo uma redução que fatora um inteiro $n = pq$ utilizando o algoritmo $A(pq, h')$. A reduçao abaixo e sua respectiva prova foram retirados do artigo de Bernstein [Ber08].

Redução 4.3.2 (*Fatorar(A)*)**.** Recebe como entrada o valor n a ser fatorado. [Ber08, p. 6]

1. Gerar aleatoriamente com uma distribuição uniforme a tripla $(e, f, s) \in \{-1, 1\} \times \{1, 2\} \times \{0, 1, 2, ..., pq - 1\}$.
2. Calcular $h' = efs^2 \bmod n$.
3. Volte ao passo 2 se $h' \notin \{1, 2, ..., 2^K\}$.
4. Calcular $(e', f', s') = A(n, h')$.
5. Se $mdc(n, s' - s) \notin \{1, n\}$ imprima e pare.
6. Se $mdc(n, s') \notin \{1, n\}$ imprima e pare.

O tempo de execução da redução *Fatorar(A)* é praticamente igual ao tempo de execução do algoritmo A, sendo que a diferença é algumas operações módulo n. O passo 3 irá retornar verdadeiro se $h' \notin \{1, 2, ..., 2^K\}$ apenas $n/2^K < 2$ vezes em média, tornando o algoritmo A menos do que duas vezes mais rápido do que a redução *Fatorar(A)*.

Teorema 4.3.3. *[Ber08, p. 7] A redução 4.3.2 fatora o valor de entrada n com probabilidade $Pr(Fatorar(A)) \geq PrInversaoCega(A)/2$.*

Demonstração. Sejam (p, q) valores aleatórios com uma distribuição uniforme. O valor $h' \in \{1, 2, ..., 2^K\}$ é um inteiro aleatório e uniforme, que pode ser produzido por *exatamente* quatro escolhas diferentes de e, f, s. Portanto o evento $e'f'(s')^2 \equiv h' \pmod{pq}$ ocorre com probabilidade igual a $PrInversaoCega(A)$. Agora vamos mostrar que caso esse evento ocorra, a redução 4.3.2 fatora n com uma probabilidade de pelo menos $1/2$. Vamos citar quatro casos que podem ocorrer:

Caso 1: $mdc(h', pq) = pq$. Este caso não pode ocorrer, pois $1 \leq h' \leq 2^K < pq$.

Caso 2: $mdc(h', pq) = p$. Nesse caso, $mdc(s', pq) = p$, e n é fatorado.

Caso 3: $mdc(h', pq) = q$. Nesse caso, $mdc(s', pq) = q$, e n é fatorado.

Caso 4: $mdc(h', pq) = 1$. Note que $e'f'(s')^2 \equiv efs^2 \pmod{pq}$, logo $e'f'(s')^2 \equiv efs^2 \pmod{p}$ e $e'f'(s')^2 \equiv efs^2 \pmod{q}$. Lembrando que os primos p e q são da forma $p \equiv 3 \bmod 8$ e $q \equiv 7 \bmod 8$. Ambas as possibilidades de f (1 ou 2) são quadrados módulo q, então $f'(s')^2$ e fs^2 são quadrados módulo q, e ambos são diferentes de zero pois $mdc(h', q) = 1$. Como $e'f'(s')^2 \equiv efs^2 \pmod{q}$ temos que e'/e é um resíduo quadrático módulo q, portanto $e'/e \neq -1$, logo $e' = e$. Com isso temos que $f'(s')^2 = fs^2 \bmod pq$, e ambos $(s')^2$ e s^2 são quadrados módulo p, e ambos são diferentes de zero pois $mdc(h', p) = 1$. Da mesma forma que antes, temos que f'/f é um quadrado módulo p, e portanto não pode ser 2, logo $f' = f$ e finalmente

$$(s')^2 \equiv s^2 \pmod{pq}$$

Todas as quatro escolhas de e, f, s que levam a h' tem o mesmo valor e, f como foi mostrado, portanto apenas duas dessas escolhas tem que $s \equiv s'$ ou $s \equiv -s'$. As outras duas escolhas ocorrem com probabilidade $1/2$, e possibilitam a fatoração do módulo n, pois temos que $(s')^2 \equiv s^2 \pmod{pq}$, o que é o mesmo que $n|(s')^2 - s^2$, que fatorando resulta em $n|(s' + s)(s' - s)$. Observe que se $s \not\equiv s'$ ou $s \not\equiv -s'$ nenhum dos fatores será igual a zero, e n divide o produto, portanto cada um dos fatores de n divide cada um dos produtos $(s' + s)(s' - s)$. E se calcularmos o $mdc(s' - s, pq)$ obteremos alguns dos fatores p ou q, e analogamente se calcularmos $mdc(s' + s, pq)$ obteremos o fator que restou p ou q, assim obtendo a fatoração de n. □

Com esse resultado, já podemos afirmar que o protocolo de assinatura *Principal Rabin-Williams $B = 0$ é existencialmente seguro contra ataques de chave pública.*

4.3.2 Inversão selecionada com uma assinatura

Suponha que temos uma chave pública $n = pq$ e dois inteiros $h, h' \in \{1, 2, ..., 2^K\}$ e uma raiz quadrada principal modificada (e, f, s) de h módulo pq. Queremos calcular a *raiz quadrada principal modificada* de h' mod n de uma forma eficiente.

Definição 4.3.4. Suponha que existe um algoritmo $A(pq, h, e, f, s, h')$ que retorne uma tripla $(e', f', s') \in \{-1, 1\} \times \{1, 2\} \times \{0, 1, 2, ..., pq - 1\}$ tal que $e'f'(s')^2$ seja a raiz quadrada modificada de h' com uma probabilidade, que chamaremos de *PrInversaoSelecionada1(A)*. Os parâmetros para esse algoritmo A são:

- Uma chave pública pq selecionada aleatoriamente com uma distribuição D.
- Um inteiro $h \in \{1, 2, ..., 2^K\}$ selecionado aleatoriamente com uma distribuição uniforme.
- (e, f, s) é a raiz quadrada principal modificada de h módulo pq.
- Um inteiro $h' \in \{1, 2, ..., 2^K\}$ selecionado aleatoriamente com uma distribuição uniforme.

sendo todas essas escolhas independentes.

Qualquer algoritmo A de probabilidade 1 pode gerar um algoritmo de probabilidade 1 para falsificar uma assinatura *Principal Rabin-Williams* $B = 0$, dado acesso público a função de *hash* H. O adversário simplesmente recebe $h = H(m)$ e (e, f, s) de uma mensagem m assinada (i.e por um oráculo de assinatura), e escolhe uma mensagem $m' \neq m$ da qual ele quer falsificar a assinatura, calcula $h' = H(m')$ utilizando a função pública H, e utiliza o algoritmo A para gerar a *raiz quadrada principal* de h' mod pq, obtendo a tripla (e', f', s') que é a assinatura válida de m'.

Redução eficiente do problema da inversão cega

Vamos criar uma redução chamada *Fatorar2(A)* que utiliza o algoritmo $A(pq, h, e, f, s, h')$ e tenta reduzir o problema da *inversão cega* para o problema da *inversão selecionada com uma assinatura*, o que implica em uma redução do problema da fatoração para o problema da *inversão selecionada com uma assinatura*. A idéia dessa redução é mostrar que um adversário que tenta atacar o protocolo não obtém vantagem nenhuma se ele tiver acesso a um par *(mensagem, assinatura)* válido. A redução *Fatorar2* tem que criar uma *simulação* perfeita de um oráculo de assinatura, de uma forma que o algoritmo A não possa distinguir se está se comunicando com o simulador ou com o oráculo de assinatura real.

Redução 4.3.5 (*Fatorar2(A)*)**.** Recebe como entrada n e h' [Ber08, p. 10].

1. Gerar aleatoriamente com uma distribuição uniforme a tripla $(e', f', x) \in \{-1, 1\} \times \{1, 2\} \times \{0, 1, 2, ..., n - 1\}$.
2. Calcular $g = mdc(x, n)$.

3. Se $g \equiv 7 \bmod 8$ então $e = 1$, senão $e = e'$.
4. Se $g \equiv 3 \bmod 8$ então $f = 1$, senão $f = f'$.
5. Calcular $s = x^2 \bmod n$.
6. Calcular $h = efs^2 \bmod n$.
7. Volte ao passo 1 se $h \notin \{1, 2, ..., 2^K\}$.
8. Imprima $A(n, h, e, f, s, h')$.

O tempo de execução da redução *Fatorar2(A)* é quase idêntico ao tempo de execução do algoritmo A, sendo a única diferença algumas operações módulo n que geram h, e que são repetidas apenas $n/2^K < 2$ vezes.

Teorema 4.3.6. *[Ber08, p. 8] A redução 4.3.5 reduz o problema da inversão cega com a probabilidade*

$$PrInversaoCega(Fatorar2(A)) = PrInversaoSelecionada1(A)$$

Demonstração. Se $e = 1$ então $h \equiv efs^2 = fs^2$ é um resíduo quadrático módulo q, pois 2 é um resíduo quadrático módulo q. Se $e = -1$ então $h \equiv efs^2 = -fs^2$, que não é um resíduo quadrático módulo q, caso contrário q divide s, e s divide x, então o $mdc(x, n) \in \{n, q\}$, então $g = n$ ou $g \equiv 7 \bmod 8$, e $e = 1$, o que é uma contradição. Igualmente, se $f = 1$ então $eh \equiv s^2$ é um quadrado módulo p, e se $f = 2$ então $eh \equiv 2s^2$ que não é um quadrado módulo p, caso contrário p divide s e p divide x, e o $mdc(x, n) \in \{n, p\}$, então $g = n$ ou $g \equiv 3 \bmod 8$, e $f = 1$, o que é outra contradição. Sabemos que s é um resíduo quadrático módulo n, e portanto temos que (e, f, s) é a raiz quadrada principal modificada de h. Agora precisamos mostrar que a distribuição de h é uniforme. Seja (e, f, s) a raiz quadrada principal modificada de h. Se $mdc(h, n) = 1$ então $mdc(s, n) = 1$, logo $mdc(x, n) = 1$, e então $e = e'$, $f = f'$ e x é uma das *quatro* raízes quadradas modificadas de $s \bmod n$. Se $mdc(h, n) = p$ então $mdc(s, n) = p$, logo $mdc(x, n) = p$ e então $e = e'$, $f \in \{1, 2\}$, e x é umas das *duas* raízes quadradas modificadas de $s \bmod n$. Se $mdc(h, n) = q$ então $mdc(s, n) = q$, logo $mdc(x, n) = q$ e então $e \in \{-1, 1\}$, $f = f'$, e x é umas das *duas* raízes quadradas modificadas de $s \bmod n$. Se $mdc(h, n) = n$ então $mdc(s, n) = n$, logo $mdc(x, n) = n$ e então $e \in \{-1, 1\}$, $f \in \{1, 2\}$, e $x = 0$. Cada inteiro h é gerado por exatamente *quatro* escolhas de (e', f', x). □

Com esse resultado, já podemos afirmar que o protocolo de assinatura *Principal Rabin-Williams $B = 0$ é existencialmente seguro contra ataques de mensagem escolhida com uma assinatura.*

4.3.3 Inversão selecionada com muitas assinaturas

Suponha que temos uma chave pública $n = pq$ e inteiros $h_1, h_2, ..., h_Q, h' \in \{1, 2, ..., 2^K\}$ e uma raiz quadrada principal modificada para cada h_i módulo pq. Queremos calcular a *raiz quadrada principal modificada* de $h' \bmod n$ de uma forma eficiente.

Definição 4.3.7. Suponha que existe um algoritmo $A(pq, h_1, e_1, f_1, s_1, ..., h_Q, e_Q, f_Q, s_Q, h')$ que retorne uma tripla $(e', f', s') \in \{-1, 1\} \times \{1, 2\} \times \{0, 1, 2, ..., pq - 1\}$ tal que $e'f'(s')^2$ seja a raiz quadrada principal de h' com uma probabilidade, que chamaremos de *PrInversaoSelecionada2(A)*. Os parâmetros para esse algoritmo A são:

- Uma chave pública pq selecionada aleatoriamente com uma distribuição uniforme.
- Cada inteiro $h_i \in \{1, 2, ..., 2^K\}$ selecionado aleatoriamente com uma distribuição uniforme.
- (e_i, f_i, s_i) é a raiz quadrada principal modificada de h_i módulo pq.
- Um inteiro $h' \in \{1, 2, ..., 2^K\}$ selecionado aleatoriamente com uma distribuição uniforme.
- Um valor Q que representa a quantidade de assinaturas válidas que o adversário é capaz de obter (aproximadamente 2^{30} [BR96]).

sendo todas essas escolhas independentes.

Qualquer algoritmo A de probabilidade 1 pode gerar um algoritmo de probabilidade 1 para falsificar uma assinatura *Principal Rabin-Williams* $B = 0$, dado acesso público a função de *hash* H. O adversário simplesmente recebe $h_i = H(m_i)$ e (e_i, f_i, s_i) de Q mensagens m_i assinadas (i.e por um oráculo de assinatura), e escolhe uma mensagem $m' \neq m_i$, tal que $1 < i \leq Q$ da qual ele quer falsificar a assinatura, calcula $h' = H(m')$ utilizando a função pública H, e utiliza o algoritmo A para gerar a *raiz quadrada principal* de h' mod pq, obtendo a tripla (e', f', s') que é a assinatura válida de m'.

Redução eficiente do problema da inversão cega

Vamos criar uma redução chamada *Fatorar3(A)* que utiliza o algoritmo $A(pq, h_1, e_1, ...)$ e tenta reduzir o problema da *inversão cega* para o problema da *inversão selecionada com muitas assinaturas*, o que implica em uma redução do problema da fatoração para o problema da *inversão selecionada com muitas assinaturas*. O parâmetro Q representa a quantidade de assinaturas que o adversário poderá obter, que na literatura [BR96] é considerado aproximadamente 2^{30}. A idéia dessa redução é mostrar que um adversário que tenta atacar o protocolo não obtém vantagem nenhuma se ele tiver acesso a muitos pares *(mensagem, assinatura)* válidos. A redução *Fatorar3* tem que criar uma *simulação* perfeita de um oráculo de assinatura, de uma forma que o algoritmo A não possa distinguir se está se comunicando com o simulador ou com o oráculo real de assinatura.

Redução 4.3.8 (*Fatorar3(A)*)**.** Recebe como entrada n e h'.

1. Para cada $i \in \{1, 2, ..., 2^K\}$:
2. Gerar aleatoriamente com uma distribuição uniforme a tripla $(e'_i, f'_i, x_i) \in \{-1, 1\} \times \{1, 2\} \times \{0, 1, 2, ..., n - 1\}$.
3. Calcular $g_i = mdc(x_i, n)$.

4. Se $g_i \equiv 7 \bmod 8$ então $e_i = 1$, senão $e_i = e'_i$.

5. Se $g_i \equiv 3 \bmod 8$ então $f_i = 1$, senão $f_i = f'_i$.

6. Calcular $s_i = x_i^2 \bmod n$.

7. Calcular $h_i = e_i f_i s_i^2 \bmod n$.

8. Volte ao passo 2 se $h_i \notin \{1, 2, ..., 2^K\}$.

9. Imprima $A(n, h_1, e_1, f_1, s_1, ..., h_Q, e_Q, f_Q, s_Q, h')$.

O tempo de execução da redução *Fatorar3(A)* é quase igual ao tempo de execução do algoritmo A, sendo a única diferença algumas operações módulo n que geram h, e que são repetidas apenas $n/2^K < 2$ vezes a cada iteração do laço, sendo assim, o tempo é em média $2Q$ o tempo de A.

Teorema 4.3.9. *A redução 4.3.8 reduz o problema da inversão cega com a probabilidade*

$$PrInversaoCega(Fatorar3(A)) = PrInversaoSelecionada2(A)$$

Demonstração. A prova deste teorema é exatamente igual a prova do teorema 4.3.6, sendo a única diferença que esta redução está respondendo a Q consultas de assinatura, ao invés de apenas uma como é feito na redução 4.3.5 □

Com esse resultado, já podemos afirmar que o protocolo de assinatura *Principal Rabin-Williams* $B = 0$ *é existencialmente seguro contra ataques de mensagem escolhida.*

Capítulo 5

Contribuição

5.1 Inversão existencial

Nesta seção, iremos apresentar a nossa nova idéia, que possibilita uma prova eficiente de segurança para o protocolo *Principal Rabin-Williams B = 0*. Essa idéia é uma variação do que foi apresentado por Bernstein em [Ber08].

Definição 5.1.1 (*Inversão Existencial Principal*)**.** Suponha que temos uma chave pública $n = pq$ com uma distribuição uniforme, e $Q + 1$ elementos $h_1, h_2, ..., h_{Q+1} \in \{1, 2, 3, ...2^K\}$ aleatórios, com uma distribuição uniforme, e independentes uns dos outros. Podemos fazer Q_i consultas a um oráculo de assinatura, que irá nos responder com a *raiz quadrada principal modificada* (e_i, f_i, s_i) de $h_i \bmod n$. Nossa tarefa é gerar uma saída (e_i, f_i, s_i) tal que $e_i f_i s_i^2 \equiv h_i \pmod n$, e (e_i, f_i, s_i) seja a *raiz quadrada principal modificada* de h_i, e h_i não foi uma das consultas feitas ao oráculo de assinatura.

Note que a diferença entre o problema da definição 5.1.1 e o problema da definição 4.3.7 é que, agora podemos *escolher* qual dos h_i é mais *fácil* de falsificar, podendo tornar o ataque mais *fácil*.

Vamos assumir que existe um algoritmo eficiente $A(n, h_1, h_2, ..., h_{Q+1})$ que resolve o problema da *Inversão Existencial Principal* com uma probabilidade $Pr(A)$

Abaixo vamos descrever um outro problema semelhante.

Definição 5.1.2 (*Inversão Existencial*)**.** Suponha que temos uma chave pública $n = pq$ com uma distribuição uniforme, e $Q + 1$ elementos $h_1, h_2, ..., h_{Q+1} \in \{1, 2, 3, ...2^K\}$ aleatórios, com uma distribuição uniforme, e independentes uns dos outros. Podemos fazer Q_i consultas a um oráculo de assinatura, que irá nos responder com uma *raiz quadrada principal modificada* (e_i, f_i, s_i) de $h_i \bmod n$. Nossa tarefa é gerar uma saída (e_i, f_i, s_i) tal que $e_i f_i s_i^2 \equiv h_i \pmod n$, e h_i não foi uma das consultas feitas ao oráculo de assinatura.

Vamos assumir que existe um algoritmo eficiente $B(n, h_1, h_2, ..., h_{Q+1})$ que resolve o problema da *Inversão Existencial* com uma probabilidade $Pr(B)$

Esse problema difere do problema da*Inversão Existencial Principal* apenas na saída esperada, que agora é uma *raiz quadrada modificada* aleatória de algum h_i que não foi consultado ao oráculo de assinaturas.

Usando o algoritmo A com $Pr(A) = 1$ podemos construir um algoritmo B com $Pr(B) = 1$ da maneira descrita a seguir.

Redução 5.1.3. Execute o algoritmo A respondendo as consultas de assinatura de cada mensagem m_i com a *raiz quadrada principal modificada* de $h_i = H(m_i) \bmod n$. Eventualmente A irá retornar (e, f, s) que é a *raiz quadrada principal modificada* de algum h_i que não tenha sido assinado.

Usando o algoritmo B com $Pr(B) = 1$ podemos construir um algoritmo A com $Pr(A) = 1/2$ da maneira descrita a seguir.

Redução 5.1.4. Execute o algoritmo B respondendo as consultas de assinatura de cada mensagem m_i com a *raiz quadrada principal modificada* de $h_i = H(m_i) \bmod n$. Eventualmente B irá retornar (e, f, s) que é alguma *raiz quadrada modificada* aleatória de algum h_i que não tenha sido assinado. Como sabemos que cada h_i possui exatamente *quatro* raízes quadradas modificadas módulo n, a probabilidade de (e, f, s) ser a *raiz quadrada principal modificada* de h_i é *apenas* uma em quatro, mas também podemos obter a *raiz quadrada principal modificada* no caso em que for retornado sua inversa, portanto a probabilidade é 1/2.

Abaixo é apresentado um diagrama mostrando as reduções para a prova de segurança. A flecha à direita significa que o problema da esquerda é reduzido para o problema da direita, utilizando a redução especificada entre parênteses.

$$\text{InversãoExistencialPrincipal(A)} \longrightarrow \text{InversãoExistencial(B)}$$

$$\text{InversãoExistencial(B)} \longrightarrow \text{InversãoExistencialPrincipal(A)}$$

$$\text{Fatoração(B)} \longrightarrow \text{InversãoExistencial(B)InversãoExistencialPrincipal(A)}$$

Com essas reduções, mostramos que a existência de um algoritmo A que resolve o problema da *Inversão Existencial Principal* implica na existência de um algoritmo B que resolve o problema da *Inversão Existencial*, da mesma forma que a existência de um algoritmo B que resolve o problema da *Inversão Existencial* implica na existência de um algoritmo A que resolve o problema da *Inversão Existencial Principal*.

Reduzindo o problema da fatoração para qualquer um dos problemas acima, de uma forma eficiente, estaremos demonstrando que o problema de gerar uma falsificação de assinatura no protocolo *Principal Rabin-Williams $B = 0$* está fortemente relacionado ao problema da fatoração.

A maior contribuição deste trabalho é esta redução eficiente, do problema da fatoração para o problema da *Inversão Existencial Principal*.

Vamos construir abaixo uma redução chamada *Fatorar4*, que utiliza o algoritmo B para obter uma fatoração do módulo n com uma probabilidade $Pr(Fatorar4(B)) \geq PrInversaoExistencial(B)/2$.

O parâmetro Q representa a quantidade de assinaturas que o adversário poderá obter, que na literatura [BR96] é considerado aproximadamente 2^{30}.

Redução 5.1.5 (*Fatorar4(B)*)**.** Recebe com entrada o módulo n.

1. Para cada $i \in \{1, 2, ..., Q+1\}$
2. Gerar aleatoriamente com uma distribuição uniforme a tripla $(e'_i, f'_i, x_i) \in \{-1, 1\} \times \{1, 2\} \times \{0, 1, 2, ..., n-1\}$.
3. Calcular $g_i = mdc(x_i, n)$.
4. Se $g_i \equiv 7 \bmod 8$ então $e_i = 1$, senão $e_i = e'_i$.
5. Se $g_i \equiv 3 \bmod 8$ então $f_i = 1$, senão $f_i = f'_i$.
6. Calcular $s_i = x_i^2 \bmod n$.
7. Calcular $h_i = e_i f_i s_i^2 \bmod n$.
8. Volte ao passo 2 se $h_i \notin \{1, 2, ..., 2^K\}$.
9. Calcular $(j, e', f', s') = B(n, h_1, h_2, ..., h_{Q+1})$ usando (e_i, f_i, s_i) para responder as consultas de assinatura do algoritmo B
10. Se $mdc(n, s' - s_j) \notin \{1, n\}$ imprima e pare.
11. Se $mdc(n, s') \notin \{1, n\}$ imprima e pare.

Demonstração. O tempo de execução da redução *Fatorar4(B)* é quase igual ao tempo de execução do algoritmo B, sendo necessário algumas operações módulo n para gerar cada h_i, que são repetidas em média 2 vezes cada iteração do laço, portanto o tempo da redução *Fatorar4(B)* é $< 2(Q+1)$ o tempo do algoritmo B.

Na seção 4.3.2 já foi demonstrado que o simulador utilizado na redução *Fatorar4(B)* gera a *raiz quadrada principal modificada* de cada h_i aleatório com distribuição uniforme, para poder responder as consultas de assinatura do algoritmo B. O que ocorre no passo 9 é que o algoritmo B irá retornar (j, e', f', s') que é uma *raiz quadrada modificada* aleatória de h_j. Como o algoritmo já havia gerado a *raiz quadrada principal modificada* de h_j temos duas *raízes quadradas modificadas* de h_j, sendo possível obter uma fatoração no frequente caso em que $s' \not\equiv s_j$ e $s' \not\equiv -s_j$, o que ocorre com uma probabilidade 1/2, como foi demonstrado no teorema 4.3.6.

Com a redução apresentada, podemos afirmar que uma redução do problema da fatoração para o problema da *Inversão Existencial Principal* ocorre com probabilidade 1/4, pois temos que o algoritmo *Fatorar4(B)* obtém uma fatoração com probabilidade 1/2, e temos que uma redução do problema da *Inversão Existencial* para o problema da *Inversão Existencial Principal* tem uma probabilidade de 1/2. □

Devido ao resultado obtido, podemos afirmar que o protocolo *Principal Rabin-Williams B = 0* é existencialmente seguro contra ataques adaptativos de mensagem escolhida.

Capítulo 6

Conclusão

Este trabalho apresenta uma prova eficiente (*tight*) de segurança no modelo do oráculo aleatório para o protocolo *Principal Rabin-Williams B = 0*.

Obtivemos essa prova eficiente de segurança demonstrando que o problema de encontrar a *raiz quadrada principal modificada* de um h aleatório módulo n é equivalente ao problema de encontrar qualquer *raiz quadrada modificada* de uma h aleatório módulo n.

O protocolo *Principal Rabin-Williams B = 0* é considerado [Ber08] o *estado-da-arte* de protocolos de assinatura eficientes na velocidade de verificação. Esse resultado fortalece a escolha do protocolo *Principal Rabin-Williams B = 0* para o uso em aplicações que necessitam de velocidade para a verificação de assinaturas.

Portanto, devido ao resultado obtido, podemos afirmar que o protocolo *Principal Rabin-Williams B = 0* é existencialmente seguro contra ataques adaptativos de mensagem escolhida.

6.1 Trabalhos futuros

Uma extensão óbvia para o trabalho seria eliminar a hipótese do *oráculo aleatório*, e com isso melhorar a segurança do protocolo, deixando o ambiente da prova mais próximo ao ambiente *real*.

Uma sugestão [Ber08] é em relação a compressão do tamanho da assinatura, podendo reduzir bastante o tamanho da assinatura, com apenas uma pequena perda de eficiência, o que talvez fosse interessante para algumas aplicações específicas.

Da mesma forma que pode ser feita a compressão, pode ser feita uma expansão da assinatura, com isso ganhando eficiência na verificação da assinatura.

Referências

[Ber03] Daniel J. Bernstein. Proving tight security for standard rabin-williams signatures, 2003.

[Ber08] Daniel J. Bernstein. Proving tight security for rabin-williams signatures. Em *EUROCRYPT*, páginas 70–87, 2008.

[BR93] Mihir Bellare e Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. Em *ACM Conference on Computer and Communications Security*, páginas 62–73, 1993.

[BR94] Mihir Bellare e Phillip Rogaway. Optimal asymmetric encryption. Em *EUROCRYPT*, páginas 92–111, 1994.

[BR96] Mihir Bellare e Phillip Rogaway. The exact security of digital signatures - how to sign with rsa and rabin. Em *EUROCRYPT*, páginas 399–416, 1996.

[BR11] Elaine Barker e Allen Roginsky. Transitions: recommendation for transitioning the use of cryptographic algorithms and key lengths. Relatório técnico, NIST, january 2011.

[CGH04] Ran Canetti, Oded Goldreich e Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[CNS99] Jean-Sébastien Coron, David Naccache e Julien P. Stern. On the security of rsa padding. Em *CRYPTO*, páginas 1–18, 1999.

[CNTW09] Jean-Sébastien Coron, David Naccache, Mehdi Tibouchi e Ralf-Philipp Weinmann. Practical cryptanalysis of iso/iec 9796-2 and emv signatures. Em *CRYPTO*, páginas 428–444, 2009.

[Cor02] Jean-Sébastien Coron. Optimal security proofs for pss and other signature schemes. Em *EUROCRYPT*, páginas 272–287, 2002.

[CR88] Benny Chor e Ronald L. Rivest. A knapsack-type public key cryptosystem based on arithmetic in finite fields. *IEEE Transactions on Information Theory*, 34(5):901–909, 1988.

[Dav82] G.I. Davida. *Chosen signature cryptanalysis of the RSA (MIT) public key cryptosystem.* TR-CS. Dep. of Electrical and Computer Science, College of Engineering and Applied Science, Univ. of Wisc., 1982.

[DH76] Whitfield Diffie e Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DO85] Yvo Desmedt e Andrew M. Odlyzko. A chosen text attack on the rsa cryptosystem and some discrete logarithm schemes. Em *CRYPTO*, páginas 516–522, 1985.

[DR02] Yevgeniy Dodis e Leonid Reyzin. On the power of claw-free permutations. Em *SCN*, páginas 55–73, 2002.

[FOPS04] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval e Jacques Stern. Rsa-oaep is secure under the rsa assumption. *J. Cryptology*, 17(2):81–104, 2004.

[GMR88] Shafi Goldwasser, Silvio Micali e Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, Abril 1988.

[KW03] Jonathan Katz e Nan Wang. Efficiency improvements for signature schemes with tight security reductions. Em *ACM Conference on Computer and Communications Security*, páginas 155–164, 2003.

[Mao03] Wenbo Mao. *Modern cryptography: theory and practice*. Prentice Hall Professional Technical Reference, 2003.

[Mis98] Jean-François Misarsky. How (not) to design rsa signature schemes. Em *Public Key Cryptography*, páginas 14–28, 1998.

[Moo88] Judy H. Moore. Protocol failures in cryptosystems. Em *Proceedings of the IEEE*, volume 76, páginas 594–602, 1988.

[Rab79] Michael O. Rabin. *Digitalized signatures and public-key functions as intractable as factorization*. MIT Laboratory for Computer Science, 1979.

[RSA78] Ronald L. Rivest, Adi Shamir e Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[Sho02] Victor Shoup. Oaep reconsidered. *J. Cryptology*, 15(4):223–249, 2002.

[Sho06] Victor Shoup. *A computational introduction to number theory and algebra*. Cambridge University Press, 2006.

[Sil97] Joseph H. Silverman. *A friendly introduction to number theory*. Prentice Hall, 1997.

[Ter08] Routo Terada. *Segurança de dados: criptografia em redes de computadores*. Edgard Blucher, 2a. edição, 2008.

[Wil80] Hugh C. Williams. A modification of the rsa public-key encryption procedure (corresp.). *IEEE Transactions on Information Theory*, 26(6):726–729, 1980.

Apêndice A

Implementação

Neste apêndice apresentaremos alguns algoritmos desenvolvidos na plataforma Maple 13 que corroboram a redução de segurança apresentada neste trabalho.

Note que não utilizamos nenhuma função de hash na redução abaixo, que apesar de ser essencial para a segurança do protocolo na prática, não impede a simulação da redução.

Abaixo, o algoritmo para geração de números *aleatórios*.

```
randomValue := overload(
  [
   proc (startVal::integer, endVal::integer) option overload;
    return RandomTools[Generate](integer(range = startVal .. endVal));
   end,
   proc (s::set) option overload;
    local endVal::integer, val::integer;
    endVal := nops(s);
    val := RandomTools[Generate](integer(range = 1 .. endVal));
    return op(val, s) end proc]);
   end
  ]
);
```

Abaixo, o algoritmo que simula o oráculo de assinatura

```
generateRandomPrincipalSignature := proc (n::integer)
 local e, e2, f, f2, s, x, g, h, K;
 K := ceil(log[2](n));
 while(true) do
  e2 := randomValue({-1, 1});
```

```
    f2 := randomValue({1, 2});
    x := randomValue(1, n-1);
    g := gcd(x, n);
    if g = n or g = 7 mod 8 then e := 1 else e := e2 end if;
    if g = n or g = 3 mod 8 then f := 1 else f := f2 end if;
    s :=  x&^2 mod n;
    h := e*f*S&^2 mod n;
    if h < 2^K then break end if;
   end do;
   return [h, [e, f, s]]
  end proc;
```

Abaixo, o algoritmo que gera assinatura Rabin-Williams não estruturada.

```
AnySignature := proc (p::integer, q::integer, lista::list)
 local m, u, v, e, f, w, invp, invq, x, raiz1, raiz2, raiz3, raiz4, j, raizes;
 j := randomValue(1, nops(lista)-1);
 m := op(j, lista);
 u := m&^((q+1)/4) mod q;
 if (u&^2 - m) mod q = 0 then e := 1 else e := -1 end if;
 v := (e*m)&^((p+1)/4) mod p;
 if v&^2 - (e*m) mod p = 0 then f := 1 else f := 2 end if;
 w := f&^((q-3)/4)*u mod q;
 x := f&^((p-3)/4)*v mod p;
 invp := 1/p mod q;
 invq := 1/q mod p;
 raiz1 := (invp*p*w+invq*q*x) mod(p*q);
 raiz2 := (invp*p*w-invq*q*x) mod(p*q);
 raiz3 := p*q-raiz1;
 raiz4 := p*q-raiz2;
 raizes := [raiz1, raiz2, raiz3, raiz4];
 return [j, e, f, op(randomValue(1, 4), raizes)];
end proc;
```

Abaixo, o algoritmo que verifica se a assinatura sign é a assinatura principal

```
isSquare := proc (p::integer, q::integer, sign::integer)
 local u, v;
 u := sign&^((q+1)/4) mod q;
 v := sign&^((p+1)/4) mod p;
 return evalb((u&^2 - sign mod q = 0) and (v&^2 - sign mod p = 0));
end proc;
```

Abaixo, o algoritmo que verifica a frequência de assinaturas principais geradas pelo algoritmo de assinatura não estruturada. A saída esperada é de aproximadamente 1/4.

```
simulate := proc (p::integer, q::integer, Q::integer)
 local random1, random2, signature, counter, total, s;
 total := 0;
 for counter from 1 to Q do
 random1 := randomValue(1, p*q-1);
 random2 := randomValue(1, 4);
 signature := op(4, AnySignature(p, q, [random1]));
 s := op(random2, signature);
 if isSquare(p, q, s) = true then total := total+1 end if end do;
 print(total/Q);
end proc;
```

Abaixo, o algoritmo que realiza a redução do problema da fatoração para o problema da *Inversão Existencial*. Esse algoritmo corrobora o resultado deste trabalho. A saída do algoritmo é a probabilidade de se obter uma fatoração do módulo n.

```
reduction := proc (Q::integer, n::integer, p::integer, q::integer, times)
 local listaH, listaAss, ass, i, h, temp, z, total, forgery, oldSign, mdc, j;
 total := 0;
 for z from 1 to times do
  for i from 1 to Q+1 do
   temp := generateRandomPrincipalSignature(n, k);
   h := op(1, temp);
   ass := op(2, temp);
   listaH := [h, op(listaH)];
   listaAss := [ass, op(listaAss)]
  end do;
  forgery := AnySignature(p, q, listaH);
  j := op(1, forgery);
  oldSign := op(3, op(j, listaAss));
  mdc := gcd(n, oldSign-op(4, forgery));
  if mdc <> 1 and mdc <> n then total := total+1 end if;
  mdc := gcd(n, oldSign);
  if mdc <> 1 and mdc <> n then total := total+1 end if
 end do;
 print(total/times);
end proc;
```

Abaixo, o algoritmo de geração de assinaturas Rabin-Williams principal.

```
signRabinWilliams := proc (m, p, q)
 local u, v, w, x, y, s, e, f;
 u := m&^((q+1)/4) mod q;
 if (u&^2 - m) mod q = 0 then e := 1 else e := -1 end if;
 v := (e*m)&^((p+1)/4) mod p;
 if v&^2 - (e*m) mod p = 0 then f := 1 else f := 2 end if;
 w := f&^((3*q-5)/4)*u mod q;
 x := f&^((9*p-11)/4)*(e*m)&^((p-3)/4)*v&^2 mod p;
 y := w+q*(q&^(p-2)*(x-w) mod p);
 s:=y;
 return [e, f, s];
end proc;
```

Para simularmos uma redução, devemos executar a função *reduction* passando como parâmetro o valor Q que seria a quantidade de assinaturas que um adversário poderia obter do oráculo de assinatura, a chave pública n, a chave secreta p e q (pois sem a chave secreta não poderíamos simular), e o parâmetro *times* que representa o número de vezes que queremos executar essa redução. O resultado obtido será a probabilidade de obtermos uma fatoração de n utilizando um algoritmo (*AnySignature*) que quebra o problema da *Inversão Existencial*. O resultado esperado é aproximadamente 1/2.