# Capítulo 3

# Introdução à Segurança Demonstrável

Rafael Dantas de Castro, Ricardo Dahab e Augusto Jun Devegili
*Instituto de Computação*
*Universidade Estadual de Campinas*
rafael.castro@gmail.com, rdahab@ic.unicamp.br, augusto@devegili.org

***Abstract***

*Provable security is the subject of theoretical cryptography that studies the formal definition of strong security requirements and methods that may be used to analyse the security of cryptographic schemes with regard to these requirements. There are currently two main models in use: the standard model and the random oracle model. We use a chronological approach to present the dawn of modern cryptography and the need of strong security notions, the first provable secure cryptographic schemes and the development of the random oracle model. This text intends to be an introductory reference for those interested in understanding provable security techniques.*

***Resumo***

*Segurança demonstrável é a área de criptografia teórica que estuda a definição formal de requisitos de segurança forte e provê métodos para analisar esquemas criptográficos em relação a esses requisitos. Atualmente há dois modelos principais usados para demonstrar a segurança de tais esquemas: o modelo padrão e o modelo do oráculo aleatório. Usando uma abordagem histórica, descrevemos os primórdios da criptografia moderna e a necessidade de noções de segurança forte, os primeiros esquemas demonstravelmente seguros e o surgimento do modelo do oráculo aleatório. Este texto procura ser uma referência introdutória para os interessados em compreender as técnicas de segurança demonstrável.*

## 3.1. Introdução

A criptografia nasceu da necessidade de pessoas se comunicarem de forma sigilosa através de um canal potencialmente inseguro. Mas ela sempre foi encarada mais como uma arte: a arte de se comunicar secretamente através de canais públicos. Seu desenvolvimento era essencialmente empírico: um "criptossistema" era bom porque ninguém sabia como quebrá-lo, e no dia em que alguém conseguisse quebrá-lo, ele deixaria de ser adequado. Tratamos aqui do processo através do qual, nas últimas décadas, a criptografia vem mudando cada vez mais o seu status de uma arte para o de uma ciência, de maneira que um pesquisador possa avaliar precisamente a segurança provida por um criptossistema e que um usuário possa tranqüilamente confiar seus segredos a um esquema baseado não na intuição dos que o criaram mas sim em evidências matemáticas incontestáveis.

O primeiro grande passo nesta direção foi dado por Shannon nos primeiros anos do pós-guerra. Ele estudou formalmente o que significa exatamente "ser seguro" e, baseado na teoria da informação, estabeleceu os principais alicerces sobre os quais a criptografia, enquanto ciência, se desenvolveria nas próximas décadas. Ele conseguiu definir, entre outras coisas, o que é a "segurança perfeita", quais são os requisitos indispensáveis para se alcançá-la e como analisar quão perto um dado criptossistema está deste ideal de segurança, criando um modelo matemático razoavelmente completo para descrever e analisar sistemas criptográficos. Isto é de extrema importância para o desenvolvimento de qualquer ciência, pois consegue-se assim uma abstração (adequada) da realidade definida por um conjunto de axiomas a partir dos quais se descobrem fatos/verdades (teoremas) sobre a (abstração dessa) realidade.

Talvez o resultado mais célebre obtido por Shannon seja a demonstração matemática de um fato que já era uma crença largamente difundida dentro da comunidade científica: a segurança do *One-time Pad*. A relevância deste resultado não pode ser superestimada. A sua maior importância não está no resultado em si, mas sim em representar o nascimento da segurança demonstrável: pela primeira vez alguém conseguia um argumento matemático que justificasse a segurança de um criptossistema dentro de um certo modelo. A observação de que esta demonstração se baseia em um modelo às vezes escapa a muita gente: Shannon, de alguma maneira, modelou matematicamente a realidade, fez algumas suposições sobre a maneira como o criptossistema seria usado e sobre como potenciais adversários poderiam interagir com ele para tentar quebrá-lo. Somente sob essas circunstâncias ele foi capaz de provar que o *One-time Pad* é perfeitamente seguro.

Falaremos mais sobre as idiossincrasias deste resultado em §3.3, mas aqui gostaríamos de destacar as duas grandes contribuições de Shannon, implícitas neste resultado, que levaram ao início da transformação da criptografia em uma ciência:

- **Noções de segurança.** Shannon conseguiu definir formalmente os requisitos que um criptossistema deveria apresentar para ser seguro, definindo claramente o que significa "quebrar" o sistema.

- **Modelos de ataque.** Shannon definiu exatamente como ataques ao sistema poderiam ser executados, qual seria o comportamento esperado de um adversário e como analisar isto matematicamente.

Shannon chegou a estes resultados fortemente baseado na intuição geral do que se esperava de um criptossistema, mas o grande passo dele foi formalizar estas intuições e conseguir construir uma teoria utilizando estes formalismos como alicerces.

Na década de 70, com o gradativo aumento da importância dos computadores em diversas áreas e o seu proporcional aumento de capacidade, criou-se simultaneamente a demanda e a oportunidade para o surgimento da criptografia assimétrica. Neste novo paradigma de criptografia as teorias de Shannon, como previamente definidas, eram de pouca valia: a noção importante agora não era mais a de um sigilo perfeito, impossível de ser atingida na criptografia assimétrica, mas sim de um sigilo computacional, que indicasse que o custo envolvido em quebrar um criptossistema seria proibitivo.

Como é comum quando ocorre qualquer mudança grande de paradigma, os primeiros trabalhos encontraram dificuldades para formalizar as noções de segurança devido à falta de um modelo apropriado. Só após alguns anos percebeu-se qual seria o caminho adequado para esta formalização e um trabalho, de certa forma análogo ao feito por Shannon décadas antes para a criptografia simétrica, passou a ser realizado para definir o que significa "ser seguro" neste novo paradigma de criptografia (e, logicamente, definir esquemas que concretizem estas definições).

A questão aqui é naturalmente mais complicada devido à assimetria: enquanto no tratamento dado por Shannon o objetivo é provar que um possível adversário não tem informação suficiente para quebrar o sistema, no caso da criptografia assimétrica isso não é verdade. Por exemplo, uma chave pública sempre tem informação suficiente para que se calcule a sua chave privada correspondente; o que precisa ser provado aqui é o quão difícil é extrair esta informação.

Novamente, a base deste estudo passa pela definição de noções de segurança e modelos de ataque, e é este o principal assunto deste texto. Iniciaremos nosso tratamento do tema na seção §3.2 com uma rápida passagem pelos resultados seminais de Shannon relacionados à teoria da informação e pelos primórdios da criptografia assimétrica, contando com noções ainda ingênuas de segurança. Esta seção é baseada em [Goldreich, 2003] e [Stinson, 2006]. A seguir, na seção §3.3 revisaremos o trabalho de formalização da criptografia feito durante os anos 80, fortalecendo as noções de segurança utilizadas, revisando criptossistemas e esquemas de assinaturas que as concretizavam e apresentando diferentes provas de segurança para sustentar estes resultados. Boa parte da apresentação desta seção é baseada em [Goldreich, 2003] e [Goldreich, 2004]. Em seguida, na seção §3.4, discutiremos o *paradigma do oráculo aleatório* para projeto de protocolos criptográficos, apresentando alguns esquemas cuja segurança será analisada neste modelo, e tecendo alguns comentários sobre a controvérsia que cerca o uso e aceitação deste modelo. Encerramos com algumas considerações finais na seção §3.5, destacando alguns dos tópicos importantes que infelizmente não puderam ser cobertos pelo texto e comentando alguns dos principais desafios enfrentados pelos pesquisadores da área atualmente.

## 3.2. Primeiras Noções de Segurança

Nesta seção abordamos brevemente dois trabalhos de suma importância para o surgimento da criptografia moderna: o tratamento dado por Shannon para a segurança de

criptossistemas, baseado na teoria da informação; e o surgimento da criptografia assimétrica em [Diffie e Hellman, 1976], acompanhado das primeiras, ainda ingênuas, definições de segurança.

### 3.2.1. Shannon e a Criptografia Simétrica

Formalmente, um *criptossistema* $\mathfrak{C}$ é uma tupla $(\mathscr{P},\mathscr{C},\mathscr{K},\mathscr{E},\mathscr{D})$ onde:

1. $\mathscr{P}$ é um conjunto enumerável de possíveis *textos em claro* com uma distribuição de probabilidade discreta associada e um algoritmo de amostragem $A_{\mathscr{P}}$;
2. $\mathscr{C}$ é um conjunto enumerável de possíveis *textos cifrados* com uma distribuição de probabilidade discreta induzida por $\mathscr{P}$ e $\mathscr{K}$ e um algoritmo de amostragem $A_{\mathscr{C}}$;
3. $\mathscr{K}$ é um conjunto enumerável de possíveis *chaves* com uma distribuição de probabilidade discreta associada e um algoritmo de amostragem $A_{\mathscr{K}}$;
4. Para cada $K \in \mathscr{K}$, existe uma regra de ciframento $e_K \in \mathscr{E}$ e uma regra de deciframento $d_K \in \mathscr{D}$ correspondente. Cada $e_K : \mathscr{P} \to \mathscr{C}$ e $d_K : \mathscr{C} \to \mathscr{P}$ são funções tais que $d_K(e_K(x)) = x$ para todo texto em claro $x \in \mathscr{P}$.

Perceba que esta definição não trata da segurança do criptossistema e até esquemas trivialmente inseguros se encaixam. Abordaremos o assunto da segurança dos esquemas após uma rápida revisão de conceitos de probabilidade necessários à discussão.

Uma *variável aleatória discreta* $\mathbf{X}$ consiste em um conjunto enumerável $X$ e uma distribuição de probabilidade discreta definida em $X$. A probabilidade de que a variável aleatória $\mathbf{X}$ tenha o valor $x$ é denotada por $\Pr[\mathbf{X} = x]$ ou então $\Pr[x]$ caso a variável $\mathbf{X}$ esteja fixada. Seja $E \subseteq X$. A probabilidade de que $\mathbf{X}$ tenha um valor em $E$ é dada por $\Pr[x \in E] = \sum_{x \in E} \Pr[x]$; $E$ é chamado de *evento*. A *probabilidade condicional* $\Pr[x|y]$ é a probabilidade de que $\mathbf{X}$ tenha o valor $x$ dado que $\mathbf{Y}$ tem o valor $y$. Pelo teorema de Bayes, $\Pr[x|y] = (\Pr[x]\Pr[y|x])/\Pr[y]$; duas variáveis $\mathbf{X}$ e $\mathbf{Y}$ são *variáveis aleatórias independentes* se e somente se $\Pr[x|y] = \Pr[x]$ para todo $x \in X$ e $y \in Y$. Se $F(\cdot)$ é uma função com domínio $X$, então $F(\mathbf{X})$ denota a aplicação da função $F$ a um elemento de $X$ escolhido de forma aleatória conforme a distribuição de probabilidade definida em $X$.

Com estas definições de criptossistema e probabilidade elementar, podemos começar a descrever propriedades da segurança de criptossistemas. Suponha que Alice e Beto queiram usar um criptossistema $\mathfrak{C}$ para se comunicar de forma confidencial. Alice e Beto escolhem uma chave $K \in \mathscr{K}$ conforme o algoritmo de amostragem definido em $\mathscr{K}$. Podemos assumir que as variáveis aleatórias referentes a $\mathscr{P}$ e $\mathscr{K}$ são independentes. Alice cifra seu texto $x \in X$ com a chave $K$ e envia $y = e_K(x) \in \mathscr{C}$ para Beto. Suponha que Eva, a adversária, intercepte $y$. O quanto de informação ela consegue obter sobre $x$ a partir de $y$?

Em um mundo ideal, Eva não deveria conseguir informação alguma sobre $x$ a partir de $y$. Neste caso, dizemos que o criptossistema $\mathfrak{C}$ possui *sigilo perfeito*. Definindo formalmente sigilo perfeito, dizemos que a probabilidade *a posteriori* de se saber $x$ a partir de $y$ deve ser igual à probabilidade *a priori* de $x$. Pelo teorema de Bayes,

$$\Pr[x|y] = \frac{\Pr[x]\Pr[y|x]}{\Pr[y]}$$

onde $\Pr[x|y]$ é a probabilidade *a posteriori* da mensagem $x$ caso o texto cifrado $y$ seja interceptado, $\Pr[x]$ é a probabilidade *a priori* de $x$, $\Pr[y|x]$ é a probabilidade do texto cifrado $y$ caso o texto em claro $x$ tenha sido escolhido, ou seja, a soma das probabilidades de todas as chaves que produzem $y$ a partir de $x$, e $\Pr[y]$ é a probabilidade de se obter o texto cifrado $y$.

Para sigilo perfeito precisamos de que $\Pr[x|y] = \Pr[x]$, i.e., ou $\Pr[x] = 0$, o que não deve acontecer[1], ou então $\Pr[y|x] = \Pr[y]$. Temos então uma caracterização de sigilo perfeito conforme o teorema a seguir.

**Teorema 1** (Sigilo perfeito)**.** *Um criptossistema possui sigilo perfeito se e somente se* $\Pr[y|x] = \Pr[y]$ *para todo* $x \in \mathscr{P}$ *e* $y \in \mathscr{C}$*, i.e.,* $\Pr[y|x]$ *deve ser independente de x.*

O teorema de sigilo perfeito pode ser interpretado da seguinte forma: para todo $x, w \in \mathscr{P}$ e $y \in \mathscr{C}$, a probabilidade total de todas as chaves que transformam $x$ em $y$ deve ser igual à de todas as chaves que transformam $w$ no mesmo $y$. Note que ao se fixar uma chave $K \in \mathscr{K}$ existe um e apenas um $y = e_K(x)$ para um determinado texto em claro $x$. Por conseguinte, o espaço de textos cifrados precisa ter pelo menos a mesma cardinalidade que o espaço de textos em claro. Note também que, para obter sigilo perfeito, $\Pr[y|x] = \Pr[y] \neq 0$ para quaisquer valores de $x, y$. Como a probabilidade *a posteriori* de $y$ independe da probabilidade de $x$, existe pelo menos uma chave que transforma qualquer texto em claro $x$ em qualquer um dos textos cifrados $y$. Entretanto, cada chave que transforma um dado $x$ em um dado $y$ precisa ser diferente, e portanto chegamos a uma importante conclusão de Shannon:

> *Para se obter sigilo perfeito, o número de chaves diferentes precisa ser pelo menos tão grande quanto o número de textos em claro:* $|\mathscr{K}| \geqslant |\mathscr{P}| \leqslant |\mathscr{C}|$.

Podemos medir a quantidade de informação produzida quando se escolhe um texto em claro através da entropia: $H(\mathscr{P}) = -\sum_{x \in \mathscr{P}} \Pr[x] \log \Pr[x]$, e de forma similar para a incerteza associada à escolha de uma chave, $H(\mathscr{K})$. Se todos os textos em claro são eqüiprováveis, então $H(\mathscr{P}) = \log |\mathscr{P}|$ e este é o limite superior para a entropia de um texto em claro. Caso a entropia da chave seja menor do que $\log |\mathscr{P}|$, a entropia do texto em claro diminui. A quantidade de informação de um texto em claro só pode ser escondida completamente caso a incerteza do espaço de chaves seja pelo menos igual à dos textos em claro: $H(\mathscr{P}) \leqslant H(\mathscr{K})$, o que nos leva a outra observação importante de Shannon:

> *Existe um limite para o que se consegue obter dada a incerteza da chave: a quantidade de incerteza que se consegue introduzir na solução de um criptossistema não pode ser maior do que a incerteza da chave.*

Suponha que haja um gerador seguro de chaves e que, para um texto em claro $x \in \mathscr{P}$ com tamanho $L_x$, seja necessária uma chave $K \in \mathscr{K}$ de tamanho $L_K$ para cifrá-lo. Sejam $R_x, R_K$ os logaritmos dos comprimentos dos alfabetos de $\mathscr{P}, \mathscr{K}$. Para se obter sigilo perfeito, é necessário que $R_x L_x \leqslant R_K L_K$. Esta desigualdade tem duas conseqüências diretas:

[1] Pois a igualdade deve ser verdadeira independentemente do valor de $x$ escolhido.

- se os alfabetos de $\mathscr{P}, \mathscr{K}$ forem iguais, a condição simplifica para $L_x \leqslant L_K$;
- se o espaço de textos em claro possui cardinalidade infinita, então não é possível obter sigilo perfeito com um espaço de chaves finito.

O exemplo canônico de criptossistema com sigilo perfeito é o *One-time Pad*:

**Criptossistema 1.** *One-time Pad* (`OTP`)

**Geração de parâmetros.** Seja $k$ um inteiro e $\mathscr{P} = \mathscr{C} = \mathscr{K} = \{0,1\}^k$.

**Geração de chave.** Seja $K \in \mathscr{K}$ uma chave escolhida conforme o algoritmo de amostragem definido em $\mathscr{K}$.

**Ciframento.** Calcule $e_K(x)$ como o resultado da operação de ou-exclusivo bit-a-bit de $K$ e $x$.

**Deciframento.** Calcule $d_K(y)$ como o resultado da operação de ou-exclusivo bit-a-bit de $K$ e $y$.

Apesar de ser uma cifra perfeita e extremamente eficiente, os requisitos de que a chave precise ser pelo menos do mesmo tamanho do texto em claro e que cada chave deva ser usada uma única vez tornam o *One-time Pad* impraticável.

Shannon também formalizou o fato de que a probabilidade de um criptoanalista decifrar mensagens aumenta quanto maior a quantidade de textos cifrados que ele obtém. Seja a *entropia condicional* dos textos em claro em relação aos textos cifrados definida como $H(\mathscr{P}|\mathscr{C}) = \sum_{x\in\mathscr{P}, y\in\mathscr{C}} \Pr[x \cap y] \log \Pr[x|y]$; é possível definir, de forma similar, a entropia respectiva do espaço de chaves. Intuitivamente, a entropia condicional é uma medida da incerteza de um texto em claro (ou chave) dado que se conhece um texto cifrado: uma entropia condicional próxima de zero indica incerteza próxima de zero, ou seja, a probabilidade de um texto em claro (ou uma chave) ser correto é próxima de um.

Como visto anteriormente, não é possível obter sigilo perfeito em um esquema onde o espaço de textos em claro seja infinito e o espaço de chaves seja finito. Um criptossistema com *sigilo ideal* é aquele em que as entropias condicionais $H(\mathscr{P}|\mathscr{C})$ e $H(\mathscr{K}|\mathscr{C})$ não se aproximam de zero quando o número de mensagens interceptadas tende ao infinito. Nestes criptossistemas, não importa quantos textos em claro um criptoanalista obtenha — não haverá uma solução única para os textos cifrados mas sim várias soluções com probabilidade similar, mesmo que o espaço de chaves seja finito. Em um criptossistema *fortemente ideal*, a entropia condicional $H(\mathscr{K}|\mathscr{C})$ mantém o valor constante $H(\mathscr{K})$.

À medida que um criptoanalista intercepta mais exemplares de texto cifrado, a entropia condicional diminui. Parametrizando a função de entropia condicional pela quantidade de texto cifrado interceptado, o teorema seguir apresenta algumas propriedades da entropia condicional.

**Teorema 2.** *Seja n a quantidade de texto em claro interceptado. A entropia condicional de uma chave $H(\mathscr{K}|\mathscr{C},n)$ é uma função decrescente em n. A entropia condicional das primeiras m letras do texto em claro é uma função decrescente em n e é menor ou igual à entropia condicional da chave.*

As definições de entropia, sigilo perfeito e entropia condicional são um subconjunto do trabalho de Shannon para analisar matematicamente o que significam e como analisar informação, comunicação e confidencialidade. Note que, embora Gilbert Vernam haja descrito o *One-time Pad* em 1917 e este ter sido informalmente considerado "inquebrável" por muitos anos, a primeira demonstração matemática de sua segurança incondicional surgiu com Shannon mais de 30 anos depois. Além do apresentado neste texto, conceitos como distância de unicidade, criptossistemas produto, álgebra de criptossistemas, criptossistemas puros e mistos, e métodos estatísticos aplicados a criptossistemas (incluindo confusão e difusão) foram desenvolvidos por Shannon e explorados para analisar criptossistemas simétricos.

### 3.2.2. Criptografia Assimétrica

Em 1976, Diffie e Hellman publicaram o artigo seminal que introduziu o conceito de criptografia de chave pública, ou criptografia assimétrica [Diffie e Hellman, 1976]. Diferentemente dos criptossistemas projetados até então, nos quais uma mesma chave era usada tanto para cifrar quanto para decifrar mensagens, a criptografia de chave pública usa um *par de chaves*: uma chave pública para ciframento e uma chave privada para deciframento. Como a chave pública é de conhecimento de todos, deixa de existir a necessidade dos criptossistemas simétricos de distribuir uma chave de forma segura apenas entre os participantes envolvidos em uma comunicação sigilosa. Mais do que isso, tornam-se possíveis esquemas simples de estabelecimento e distribuição de chaves, bem como esquemas de assinaturas digitais. Conforme a notação definida em §3.1, em um criptossistema de chave pública $\mathfrak{C} = (\mathscr{P}, \mathscr{C}, \mathscr{K}, \mathscr{E}, \mathscr{D})$ temos que:

- cada entidade possui um par de chaves $(SK, PK) \in \mathscr{K}$; $SK$ é dita chave privada e $PK$ é dita chave pública;
- para um par de chaves $(SK, PK)$, a regra de ciframento é $e_{PK} \in \mathscr{E}$ e a regra de deciframento é $d_{SK} \in \mathscr{D}$.

Uma outra definição comumente utilizada é a seguinte: um criptossistema de chave pública é uma tupla de algoritmos $(Gen, \mathscr{E}, \mathscr{D})$ tais que

- $Gen$ é o algoritmo de geração de chaves: recebe como parâmetro $1^k$, onde $k$ é um parâmetro de segurança, e devolve um par de chaves $(SK, PK)$;
- $\mathscr{E}$ é o algoritmo de ciframento: dado um texto em claro $x$ pertencente ao espaço de textos em claro $\mathscr{P}$, o algoritmo $\mathscr{E}_{PK}(x)$ devolve um texto cifrado $y$ com tamanho polinomial $p(k)$;
- $\mathscr{D}$ é o algoritmo de deciframento: $\mathscr{D}_{SK}(y)$ devolve o texto em claro $x$ ou o símbolo $\perp$ que representa um deciframento incorreto;
- o criptossistema precisa satisfazer o requisito de *correção*: para todo $x \in \mathscr{P}$ e todos os possíveis pares $(PK, SK)$ devolvidos por $Gen$, tem-se que $\mathscr{D}_{SK}(\mathscr{E}_{PK}(x)) = x$.

Criptossistemas de chave pública requerem a existência de dois tipos de função: funções unidirecionais e funções (unidirecionais) com segredo. Informalmente, uma função é unidirecional se é fácil calculá-la mas difícil invertê-la. Uma função é com segredo se ela é uma função unidirecional em que a inversão é fácil quando se conhece um segredo. Quando a função induz uma permutação no domínio, usamos as expressões permutação unidirecional e permutação com segredo.

**Definição 1.** *Uma função* $f : \{0,1\}^* \to \{0,1\}^*$ *é dita* unidirecional *se as seguintes condições valem:*

1. ***Eficiência.*** *Existe um algoritmo A de tempo polinomial tal que* $A(x) = f(x)$*;*
2. ***Dificuldade de inversão.*** *Para todo algoritmo probabilístico* $A'$ *de tempo polinomial, todo polinômio positivo* $p(\cdot)$ *e n suficientemente grande,*

$$\Pr[A'(f(X_n), 1^n) \in f^{-1}(f(X_n))] < \frac{1}{p(n)}.$$

A condição de eficiência indica a facilidade de calcular a função $f$. A condição de dificuldade de inversão indica que todo algoritmo probabilístico de tempo polinomial tem chance desprezível[2] de inverter $f$.

Definiremos agora funções (unidirecionais) com segredo.

**Definição 2.** *Uma função (unidirecional)* $f_r : \{0,1\}^* \to \{0,1\}^*$ *é dita* com segredo *se existe um par* $(r,s) \in \{0,1\}^* \times \{0,1\}^*$ *tal que as seguintes condições valem:*

1. ***Eficiência.*** *Existe um algoritmo F de tempo polinomial tal que* $F(x,r) = f_r(x)$*;*
2. ***Dificuldade de inversão.*** *Para todo algoritmo probabilístico* $A'$ *de tempo polinomial, todo polinômio positivo* $p(\cdot)$ *e n suficientemente grande,*

$$\Pr[A'(r, f_r(X_n), 1^n) \in f_r^{-1}(f_r(X_n))] < \frac{1}{p(n)}.$$

3. ***Facilidade de inversão com segredo.*** *Existe um algoritmo polinomial determinístico, denotado por* $F^{-1}$*, tal que* $F^{-1}(F(x,r),s) = x$*.*

A seguir damos alguns exemplos dos primeiros criptossistemas de chave pública eas permutações com segredo por eles utilizadas.

**Criptossistema 2.** *RSA*

**Geração de chaves.** Selecione aleatoriamente dois números primos $p,q$ tais que $p \neq q$. Seja $N = pq$ e $\phi(N) = (p-1)(q-1)$. Escolha aleatoriamente $d,e$ tais que $de \equiv 1 \pmod{\phi(N)}$. A chave privada é o par $SK = (N,d)$ e a chave pública é o par $PK = (N,e)$.

**Ciframento.** Calcule $e_{PK}(x) = x^e \pmod{N}$.

**Deciframento.** Calcule $d_{SK}(y) = y^d \pmod{N}$.

O RSA é uma família de permutações com segredo indexada pelos pares $(N,e)$ conforme a definição de criptossistema acima. O algoritmo $F_{\mathrm{RSA}}$, ao receber como entrada $((N,e),x)$, devolve

$$\mathrm{RSA}_{N,e}(x) \stackrel{\mathrm{def}}{=} x^e \pmod{N}.$$

O algoritmo $F_{\mathrm{RSA}}^{-1}$ é idêntico ao algoritmo $F_{\mathrm{RSA}}$.

[2]Uma função $\mu : \mathbb{N} \to \mathbb{R}$ é *desprezível* em $n$ se para todo polinômio positivo $p(\cdot)$ e $n$ suficientemente grande, $\mu(n) < 1/p(n)$.

Caso um adversário consiga fatorar $N$, é trivial calcular $d = e^{-1} \pmod{\phi(N)}$. Não se sabe se a fatoração de números inteiros pode ser reduzida à inversão de $\mathrm{RSA}_{N,e}$, mas os melhores algoritmos para inverter esta permutação com segredo são baseados em fatoração. De qualquer forma, acredita-se que a família RSA é de difícil inversão. Quanto à inversão com segredo, é fácil verificar que para todo $x \in \mathbb{Z}_N^*$,

$$F_{\mathrm{RSA}}^{-1}((N,d), F_{\mathrm{RSA}}((N,e),x)) = x^{ed} = x \pmod{N}.$$

Apresentamos agora o criptossistema de Rabin [Rabin, 1979].

**Criptossistema 3.** *Rabin*

**Geração de chaves.** Gere dois números primos grandes $p,q$ tais que $p \neq q$ e $p,q \equiv 3 \pmod 4$, e calcule $N = pq$. A chave privada é o par $SK = (p,q)$ e a chave pública é $PK = N$.

**Ciframento.** Calcule $e_{PK}(x) = x^2 \pmod{N}$.

**Deciframento.** Calcule $d_{SK}(y) = \sqrt{y} \pmod{pq}$.

Percebe-se pela descrição acima que o criptossistema de chave pública de Rabin é um caso particular do RSA com $e = 2$ e $d = 1/2$. Os índices passam a ser os valores de $N$ e os segredos passam a ser os pares $(p,q)$.

De forma geral, calcular raízes quadradas módulo um número composto $N$ (em particular, um produto de dois primos grandes) é um problema para o qual não há um algoritmo que o resolva em tempo polinomial. Por outro lado, quando se conhece a fatoração de $N$, basta calcular as raízes quadradas módulo os fatores primos de $N$ e aplicar o teorema chinês sobre o resto para combinar os resultados. No caso particular de $N = pq$, calcular $\sqrt{y} \pmod{pq}$ significa:

1. Calcular $w_1 = \sqrt{y} \pmod{p}$ (eficiente se $p \equiv 3 \pmod 4$);
2. Calcular $w_2 = \sqrt{y} \pmod{q}$ (eficiente se $q \equiv 3 \pmod 4$);
3. Usar o teorema chinês sobre o resto para resolver os quatro sistemas abaixo:

$$\begin{array}{lllll} (1) & x \equiv w_1 \pmod p & \text{e} & x \equiv w_2 \pmod q \\ (2) & x \equiv w_1 \pmod p & \text{e} & x \equiv -w_2 \pmod q \\ (3) & x \equiv -w_1 \pmod p & \text{e} & x \equiv w_2 \pmod q \\ (4) & x \equiv -w_1 \pmod p & \text{e} & x \equiv -w_2 \pmod q. \end{array}$$

Cada um desses sistemas possui uma solução única (pois $p,q$ são primos distintos) que também é solução de $\sqrt{y} \pmod{pq}$.

Isto mostra que as operações definidas pelo Rabin podem ser executadas eficientemente. Mas como podemos analisar a sua segurança? Certamente as idéias de Shannon não são aplicáveis aqui: a chave pública contém *toda* a informação necessária para a recuperação da chave privada. O que esperamos, no entanto, é que efetivamente extrair esta informação seja difícil.

Provamos então a segurança do Rabin reduzindo um problema difícil à quebra do criptossistema. Usamos aqui uma noção bastante ingênua de segurança, onde "quebrar" um criptossistema significa decifrar um texto cifrado sem conhecer a chave privada

correta. Na próxima sessão discutimos mais profundamente por que esta é uma noção bastante ingênua de segurança e apresentamos definições mais robustas. Começamos nossa demonstração então mostrando que fatoração e extração de raízes quadradas modulares são problemas computacionalmente equivalentes no sentido de que um pode ser reduzido a outro e vice-versa. Depois mostramos, trivialmente, que a extração de raízes pode ser reduzida à quebra do Rabin (como definida acima).

O método descrito acima para calcular $\sqrt{y} \pmod{pq}$ é a base da redução do problema de extração de raízes quadradas modulares para o problema da fatoração. Para a redução da fatoração à extração de raízes modulares, considere o seguinte algoritmo probabilístico que recebe como entrada $N$, um número composto:

1. Selecione de forma aleatória e uniforme um número $r \in \{1, \ldots, N-1\}$;

2. Calcule $g = \gcd(N, r)$. Se $g > 1$, então gere $g$ como saída e termine.

3. Seja $s = r^2 \pmod{N}$.

4. Execute o algoritmo de extração de raízes modulares com entrada $s$, obtendo como resultado $r'$ tal que $(r')^2 \equiv s \pmod{N}$.

5. Calcule $g = \gcd(N, r - r')$. Se $g > 1$, então gere $g$ como saída e termine.

Como o algoritmo imprime $g$, um divisor não-trivial de $N$, basta calcular a divisão $N/g$ para obter outro divisor. A fatoração completa de $N$ é obtida através da invocação recursiva do algoritmo passando como entrada tanto $g$ quanto $N/g$.

Demonstramos agora a correção do algoritmo. Podemos assumir que $r$ é coprimo com $N$. Assuma que $N = pq$ e portanto existem quatro raízes quadradas de $s$ módulo $N$; sejam $\{\pm r, \pm t\}$ estas raízes. Observe que $\Pr[r' = \pm t] = 1/2$. Como $r^2 \equiv t^2 \pmod{N}$, temos que $r^2 - t^2 \equiv 0 \pmod{N}$ e portanto $(r+t)(r-t) \equiv 0 \pmod{N}$, ou seja, $r-t$ (ou $r+t$) é um fator de $N$ com probabilidade $1/2$. Por conseguinte, o algoritmo encontra um fator de $N$ com probabilidade *pelo menos* $1/2$. Portanto, após $t$ execuções do algoritmo tem-se uma probabilidade $(1 - 2^{-k})$ de fatorar $N$.

Outra maneira de encarar este resultado é a seguinte: se existir um algoritmo $A$ capaz de calcular raízes módulo $N$ *para uma parcela não-desprezível* dos $x \in \mathbb{Z}_N$, então podemos construir o algoritmo $S_A$, como descrito acima, que consegue fatorar $N$ em tempo polinomial. A observação de que $A$ tem que ser capaz de calcular raízes (ou decifrar mensagens) numa parcela não desprezível dos $x \in \mathbb{Z}_N$ é necessária para garantir que $S_A$ rode em tempo polinomial e é fonte de uma importante objeção a esta redução: há aqui uma sutil suposição sobre o espaço de mensagens a serem cifradas, especificamente em relação ao seu tamanho. Se o espaço de textos claros tiver cardinalidade maior ou igual a $\log|\mathbb{Z}_N|$, um algoritmo $A'$ capaz de quebrar o esquema pode ser usado para criar um $S_{A'}$ que executa em tempo polinomial. No entanto, se o espaço de textos em claro tiver cardinalidade menor que $\log|\mathbb{Z}_N|$, um adversário $A''$ capaz de quebrar o esquema não gera um $S_{A''}$ que executa em tempo polinomial, ou seja, a redução acima não traz nenhuma garantia de segurança. Esta observação será discutida mais profundamente em §3.3.

### 3.2.3. Assinaturas Digitais

Além dos criptossistemas de chave pública, a criptografia assimétrica permite a construção de esquemas de assinaturas, os quais permitem verificar que uma determinada informação foi atestada por uma entidade. A seguir, definimos formalmente esquemas de assinatura.

**Definição 3** (Esquemas de assinatura)**.** *Um esquema de assinatura é uma tupla* $(G,S,V)$ *de algoritmos probabilísticos de tempo polinomial tais que:*

1. *Ao receber* $1^n$ *o algoritmo G (geração de chaves) retorna um par de strings;*
2. *Para todo todo par* $(s,v)$ *imagem de* $G(1^n)$*, para todo* $\alpha \in \{0,1\}^*$*, os algoritmos S (assinatura) e V (verificação) satisfazem*

$$\Pr[V(v,\alpha,S(s,\alpha)) = 1] = 1\,.$$

Muitas vezes no texto usaremos a notação $V_v(\alpha,\sigma)$ e $S_s(\alpha)$ para $V(v,\alpha,\sigma)$ e $S(s,\alpha)$. Note que esta definição de esquemas de assinatura não leva em consideração requisitos particulares de segurança para assinaturas digitais. De fato, assim como em nossa primeira definição de criptossistema, esquemas trivialmente inseguros são concretizações válidas desta definição[3]. De forma ingênua, pode-se implementar um esquema de assinatura através de permutações com segredo de forma análoga aos criptossistemas de chave pública: o algoritmo de verificação como um "ciframento" com a chave pública e o algoritmo de assinatura como um "deciframento" com a chave privada correspondente. Por exemplo, pode-se usar a família de permutações RSA como um possível esquema de assinatura onde o algoritmo $F_{\mathrm{RSA}}$ é o algoritmo de verificação e o algoritmo $F_{\mathrm{RSA}}^{-1}$ é o algoritmo de assinatura.

Na próxima seção, discutiremos mais profundamente requisitos de segurança mais fortes para criptossistemas e esquemas de assinatura no contexto da criptografia assimétrica, especificamente discutindo por que definições ingênuas como as usadas na demonstração de segurança do Rabin são inadequadas e fornecendo opções mais robustas que vieram a se tornar padrão.

## 3.3. Noções Fortes de Segurança

Os anos 80 trouxeram uma necessidade de reavaliação da recente revolução impulsionada pela criptografia assimétrica. As ferramentas utilizadas até então para a avaliação de segurança de criptossistemas não se adequavam bem ao novo cenário: por exemplo, um criptossistema assimétrico nunca poderia atingir sigilo perfeito pois sempre existe uma relação matemática entre chaves públicas e privadas. A segurança neste cenário não está mais baseada na *existência* de informação relativa a textos em claro nos respectivos textos cifrados, mas em quão difícil é calcular (ou utilizar) esta informação.

Um profundo estudo de formalização da criptografia, fortemente baseado na teoria da complexidade, marcou esta década, impulsionado por trabalhos como os de [Goldwasser e Micali, 1984] [Goldwasser e Micali, 1982], [Yao, 1982], [Blum e Goldwasser, 1985], e [Micali et al., 1988]. Neste primeiro momento algumas

[3]De fato, $S_s(\alpha) = \alpha$ e $V_v(\alpha,\sigma) = 1, \forall \alpha = \sigma$ obedecem à definição 3.

noções fortes de segurança foram definidas, criptossistemas que as seguiam foram propostos e, finalmente, a relação entre as diferentes noções de segurança foi esclarecida.

Posteriormente, a preocupação se voltou para o tipo de ataque contra o qual "segurança" era definida. Nos primeiros trabalhos considerava-se sempre segurança contra adversários passivos. Modelos que davam mais poder aos adversários foram projetados de maneira a provar a segurança dos criptossistemas em situações mais diversas, culminando com a busca por criptossistemas resistentes a ataques completamente adaptativos.

Mostraremos os principais resultados teóricos desta época, bases da segurança demonstrável que se tem atualmente, fazendo um panorama abrangente dos resultados que se conheciam até o início da década de 90.

### 3.3.1. Ciframento, ou Como Jogar Pôquer Mentalmente

No final dos anos 70 a comunidade de criptografia se deparava com dois problemas aparentemente simples mas que não pareciam ser facilmente resolvíveis no ambiente de criptografia assimétrica como definida em [Diffie e Hellman, 1976]:

1. Como jogar pôquer pelo telefone de maneira segura. [Shamir et al., 1979] [Lipton, 1981];
2. Como enviar de forma segura um bit de informação.

Estes são problemas que, intuitivamente, deveriam ser resolvidos por criptossistemas seguros. No entanto, não se conseguia nenhuma evidência matemática de que, a partir das idéias de Diffie & Hellman, estes problemas poderiam ser resolvidos de forma satisfatória.

Estes dois problemas destacam que a abordagem sugerida pelo artigo seminal de [Diffie e Hellman, 1976] para ciframento assimétrico, e implementada nos esquemas RSA [Rivest et al., 1978] e Rabin [Rabin, 1979], sofre de duas limitações importantes [Goldwasser e Micali, 1984]:

1. $f$ **ser uma função unidirecional não implica que ela seja de difícil inversão para um subconjunto do seu domínio.** Uma função unidirecional é uma função de difícil inversão para entradas escolhidas aleatoriamente do seu domínio. Isto não implica que esta mesma função será de difícil inversão quando se sabe que a mensagem cifrada é, por exemplo, um número de um domínio pequeno, ou a codificação de uma carta de baralho.
2. $f$ **ser uma função unidirecional não implica a dificuldade de calcular informações parciais sobre sua entrada.** Dado um $f(x)$ aleatório, se $f$ é unidirecional, sabemos que é difícil calcular o valor de $x$. Porém, esta definição por si só não diz nada sobre a (im)possibilidade de calcular informações parciais sobre $x$ como sua paridade ou um bit específico.

Um criptossistema seguro não deve ser vulnerável em qualquer das situações acima. Entretanto, qualquer implementação diretamente baseada nas idéias de [Diffie e Hellman, 1976] sofre destes problemas: por exemplo, como todo texto em claro $M$ tem um equivalente cifrado $E(M)$ único, predicados como *"$E(M)$ é par?"* são sempre facilmente calculáveis.

#### 3.3.1.1. Noções Fortes de Segurança

Com estes problemas em mente, três noções fortes de segurança foram propostas, cada uma à sua maneira, adaptando as idéias de Shannon à criptografia de chave pública. Nas definições abaixo $k$ é um parâmetro de segurança, os adversários $A$ são algoritmos de tempo polinomial e $\mathfrak{C}$ é o criptossistema sendo analisado.

- **Segurança Polinomial.**[4] Esta definição de segurança deriva diretamente da idéia de *indistinguibilidade*: dado um par de mensagens e uma delas cifrada, um adversário não deve ser capaz de distinguir, em tempo polinomial, a qual das mensagens em claro o texto cifrado corresponde. Dado um par de mensagens $\{m_0, m_1\}$ arbitrariamente escolhidas, sejam $i \stackrel{r}{\leftarrow} \{0,1\}$, $E \leftarrow \mathscr{C}(1^k)$ e $\alpha \leftarrow E(m_i)$. O criptossistema $\mathfrak{C}$ é *polinomialmente seguro* se, para todos os adversários $A$ e todo $c > 0$,
$$\Pr[A_k(E, m_0, m_1, \alpha) = m_i] < \frac{1}{2} + k^{-c}.$$
- **Segurança Semântica.**[5] Seja $f$ uma função definida no espaço de mensagens. Informalmente, $f(m)$ representa informação sobre $m$. A noção de segurança semântica traduz o fato de que deveria ser difícil calcular o valor de qualquer $f(m)$ dado o texto cifrado $E(m)$: é uma tradução do conceito de "sigilo perfeito" de Shannon para o ambiente onde todos os participantes estão limitados por um número polinomial de passos. Dados os três jogos a seguir:
  - **Jogo 1.** Escolha $m \stackrel{r}{\leftarrow} M$. Neste jogo, $A_1$ tem que calcular o valor de $f(m)$ sem conhecer $m$.
  - **Jogo 2.** Escolha $m \stackrel{r}{\leftarrow} M$. Calcule um $\alpha \leftarrow E(m)$ e forneça ao adversário. Neste jogo, $A_2$ tem que calcular o valor de $f(m)$ conhecendo $E(m)$.
  - **Jogo 3.** Deixe $A_3$ escolher uma função $f^*$ definida em $M$. Escolha $m \stackrel{r}{\leftarrow} M$, calcule um $\alpha \leftarrow E(m)$ e forneça ao adversário. $A_3$ tem que calcular o valor de $f^*(m)$.

  Seja $A_i^*$ o evento em que o adversário vence o jogo $i$. Um criptossistema $\mathfrak{C}$ é semanticamente seguro se
  $$\Pr[A_3 *] < \Pr[A_1^*] + k^{-c},$$
  ou seja, o conhecimento da mensagem cifrada e a possibilidade de escolher a função que se quer calcular não devem trazer vantagem para o adversário.
- **Segurança "Yao".**[6] A definição de segurança de Yao é baseada em teoria da informação, adicionando a limitação polinomial nas operações das partes envolvidas. Esta definição é um pouco menos intuitiva que as duas anteriores, e exige um conjunto maior de conceitos auxiliares. Por este motivo apresentaremos uma explicação um tanto superficial da definição, referindo o leitor a [Yao, 1982] e [Micali et al., 1988] para uma definição mais precisa.

[4] Também conhecida como *indistinguibilidade de textos cifrados*. Mantivemos o nome original, como definido em [Goldwasser e Micali, 1984].

[5] Como definido em [Goldwasser e Micali, 1984].

[6] Como definido em [Yao, 1982].

A(lice) tem $n^k$ mensagens que ela gostaria de transmitir a B(ob). Essas mensagens foram selecionadas de um conjunto $\mathscr{P}$ de possíveis mensagens com uma distribuição de probabilidade conhecida por A e B. Qual a quantidade mínima de bits que A precisa transmitir para B de maneira que este possa recuperar todas as $n^k$ mensagens? Seja $q_1$ esta quantidade de bits. Agora imagine que B conhece um texto cifrado $E(m_i)$ para cada uma das $n^k$. Seja $q_2$ o número de bits necessários a B nesta situação. Certamente $q_2 \leqslant q_1$. Um criptossistema é *Yao-Seguro* se $q_2 = q_1$, ou seja, se o número de bits que A precisa enviar a B é o mesmo independentemente de B conhecer ou não os textos cifrados.

Estas definições de segurança têm em comum a preocupação com a proteção de toda informação referente ao texto em claro. Não estava muito claro na época qual seria a definição correta de segurança, até o trabalho de [Micali et al., 1988], onde os autores demonstram o teorema a seguir.

**Teorema 3.** *As noções de segurança polinomial, segurança semântica e segurança "Yao" são equivalentes [Micali et al., 1988].*

Este resultado trouxe consigo um aumento na confiança de que estas noções de segurança eram, de fato, noções "corretas".

**Ciframento Determinístico.** Note que um esquema de ciframento determinístico, como o RSA e o Rabin, nunca pode atingir estas noções fortes de segurança. Como os três tipos de segurança são equivalentes, basta observar que um criptossistema determinístico nunca pode ser polinomialmente seguro: dados uma função de ciframento $E$, dois textos em claro ($m_0$ e $m_1$), e $\alpha = E(m_i)$ para $i \xleftarrow{r} \{0,1\}$, um adversário pode sempre aplicar $E$ a $m_0$ e $m_1$ e comparar o resultado a $\alpha$. Se $E$ é uma função determinística, o adversário sempre vai ser capaz de decidir corretamente a quem $\alpha$ corresponde. Veja que se $E$ fosse aleatorizada este procedimento não funcionaria pois existiria um número exponencialmente grande de possíveis textos cifrados para cada $m_i$: a probabilidade de o adversário aplicar $E$ a $m_i$ e obter $\alpha$ seria desprezível.

### 3.3.1.2. Ciframento Probabilístico

Goldwasser & Micali, responsáveis pelas duas primeiras definições de segurança apresentadas acima, propuseram um novo paradigma de ciframento: o *ciframento probabilístico*. Em [Goldwasser e Micali, 1984] eles argumentam a necessidade de quebrar a bijeção entre textos em claro e cifrados para que se consigam esquemas realmente seguros (i.e., segundo as definições em §3.3.1.1). A ferramenta para isto é, ainda segundo os autores, o uso de funções de ciframento que funcionem de maneira probabilística e que possam, para uma mensagem $m_i$ qualquer, gerar uma quantidade exponencialmente grande de diferentes textos cifrados $E(m_i)$, todos válidos. O deciframento continua sendo determinístico e $\Pr[D(E(m_i)) = m_i] = 1$.

Como um exemplo deste paradigma de ciframento probabilístico, os autores propuseram um criptossistema de chave pública e provaram que ele é polinomialmente seguro. Esta proposta utiliza, em lugar de funções unidirecionais com segredo, a idéia

de *predicados não-aproximáveis com segredo*. Basicamente um predicado $B : \{0,1\}^* \rightarrow \{0,1\}$ é *não-aproximável* e *com segredo* se qualquer pessoa pode escolher $x$ e $y$ tais que $B(x) = 1$ e $B(y) = 0$, mas só quem conhece o segredo é capaz de, dado um $z$, calcular $B(z)$.

Digamos que Bob escolha um predicado não-aproximável com segredo $B^* : \{0,1\}^k \rightarrow \{0,1\}$ tal que ele conheça o segredo $s^*$ correspondente. Bob pode então publicar $B^*$ e ele passará a servir como sua chave pública, enquanto $s^*$ será sua chave privada. Suponha agora que Alice quer enviar a Bob um bit $b \in \{0,1\}$ de maneira segura. Alice pode escolher um $x$ aleatório tal que $B^*(x) = b$ e enviar para Bob. Bob, conhecendo $s^*$, será capaz de calcular $B^*(x)$ e recuperar o valor de $b$; qualquer outra pessoa que intercepte a mensagem, no entanto, será incapaz de calcular $b$, pois isto implicaria o cálculo de $B^*(x)$.

Este esquema simples resolve o problema de como enviar um bit de forma segura. Agora podemos generalizar esta abordagem para uma mensagem de tamanho arbitrário simplesmente representando-a como uma seqüência de bits $M = m_0 m_1 m_2 \ldots m_{n-1}$, onde $m_i$ é o $i$-ésimo bit de $M$. Alice pode cifrar $M$ bit a bit, utilizando o procedimento descrito acima. O criptossistema `GM` é então definido como:

**Criptossistema 4.** *Goldwasser-Micali* (`GM`)

**Geração de chaves.** Selecione aleatoriamente um predicado não-aproximável e com segredo $B^*$ e um segredo $s^*$ associado a ele. A chave pública é (a descrição de) $B^*$, e a chave privada é $s^*$.

**Ciframento.** Seja $M = m_0 m_1 m_2 \ldots m_{n-1}$ a mensagem a ser cifrada. Para cada $m_i$ escolha aleatoriamente $x_i$ tal que $B^*(x_i) = m_i$. O texto cifrado é $C = x_0 || x_1 || \ldots || x_{n-1}$.

**Deciframento.** Seja $C = x_0 || x_1 || \ldots || x_{n-1}$ o texto a ser decifrado. Utilizando $s^*$ calcule $m'_i = B^*(x_i)$ para cada $i$. A mensagem original é $M = m_0 || m_1 || \ldots || m_{n-1}$.

Este criptossistema foi proposto em [Goldwasser e Micali, 1982] utilizando o problema do resíduo quadrático como predicado. Sua segurança polinomial foi provada neste mesmo artigo, tornando-o assim o primeiro criptossistema provado seguro sob uma noção forte de segurança. Apresentamos aqui este esquema principalmente como uma referência histórica, mas analisaremos mais profundamente uma variação deste cuja demonstração de segurança é mais simples e didática.

### 3.3.1.3. Ciframento Probabilístico Baseado em Permutações com Segredo

A idéia anterior de Goldwasser & Micali pode ser adaptada para utilizar *permutações com segredo*. Lembramos que, intuitivamente, uma permutação com segredo é uma função unidirecional com segredo que induz uma permutação no seu domínio (para mais detalhes, consulte §3.2.2).

Já discutimos que o fato de uma função (resp. permutação) ser de difícil inversão não implica que informações parciais sobre a entrada não possam ser calculadas a partir

do resultado. Para tornar explícito o que é difícil de ser calculado sobre um $x$ arbitrário dado apenas o conhecimento de $f(x)$ (resp. $p(x)$), definimos a noção de *hard-core* de uma função (resp. permutação).

**Definição 4** (Hard-Core de uma Função)**.** *Seja* $b : \{0,1\}^* \to \{0,1\}^*$ *uma função calculável em tempo polinomial tal que para todo* $|x| = |y|$ *tem-se* $|b(x)| = |b(y)|$*. Seja* $l(n) \stackrel{def}{=} |b(1^n)|$*. A função b é chamada de hard-core da função f se, para todo algoritmo polinomial A, todo polinômio positivo* $p(.)$ *e todo n suficientemente grande,*

$$\text{abs}(\Pr[A(f(X_n), b(X_n)) = 1] - \Pr[A(f(X_n), R_{l(n)}) = 1]) < \frac{1}{p(n)}, \tag{1}$$

*onde* $X_n$ *e* $R_{l(n)}$ *são duas variáveis aleatórias independentes uniformemente distribuídas em* $\{0,1\}^n$ *e* $\{0,1\}^{l(n)}$ *respectivamente.*

Definimos então criptossistema abaixo.

**Criptossistema 5.** *Um esquema simples de ciframento com chave pública* (`CS-2`)

**Geração de chaves.** Selecione aleatoriamente uma permutação $p^*$ e um segredo $s^*$ associado a ela; a chave pública é (a descrição de) $p^*$ e a chave privada é $s^*$.

**Ciframento.** Seja $\sigma$ o bit a ser cifrado e seja $b^*(.)$ um *hard-core* de $p^*$. Selecione aleatoriamente um elemento $r$ do domínio de $p^*$ e calcule o texto cifrado $C = (p^*(r), \sigma \oplus r)$.

**Decifrамento.** Seja $C = (\gamma_1, \gamma_2)$ o texto a ser decifrado. Utilizando $s^*$, calcule $r' = p^{-1}(\gamma_1)$; o bit original é $\sigma = \gamma_2 \oplus r'$.

Vamos apresentar uma demonstração simples da segurança deste criptossistema baseada nas propriedades da permutação com segredo e de seus *hard-cores*.

**Teorema 4.** *O criptossistema* `CS-2` *é polinomialmente seguro.*

*Demonstração.* Lembrando a definição de segurança polinomial, precisamos demonstrar que, dadas duas mensagens $(m_0, m_1)$ e uma delas cifrada $(\gamma)$, é difícil distinguir qual das duas foi cifrada. Lembre também que a noção de "difícil" aqui significa "algo que, em tempo polinomial, só pode ser feito com probabilidade desprezível no tamanho da entrada". Como estamos cifrando apenas um bit, basta provarmos que, dada a chave $\alpha$, é difícil distingüir o ciframento de 0 do ciframento de 1, i.e., é difícil distinguir as distribuições $E_\alpha(1) = (p_\alpha(r), 1 \oplus b(r))$ de $E_\alpha(0) = (p_\alpha(r), 0 \oplus b(r))$, para $r$ aleatório. Isto é, para todo algoritmo polinomial $A$ tem-se

$$\text{abs}(\Pr[A(p_\alpha(r), 1 \oplus b(r))] - \Pr[A(p_\alpha(r), 0 \oplus b(r))]) < \frac{1}{\text{poli}(n)}, \tag{2}$$

para $r \stackrel{r}{\leftarrow} \{0,1\}^n$, todo polinômio poli(.) e $n$ suficientemente grande. Intuitivamente, isto segue diretamente do fato de $b(.)$ ser um *hard-core* de $p_\alpha$. Relembrando a equação 1 (adaptada à permutação $p$), para qualquer algoritmo $A$,

$$\text{abs}(\Pr[A(p(X_n), b(X_n)) = 1] - \Pr[A(p(X_n), R_{l(n)}) = 1]) < \frac{1}{\text{poli}(n)}, \tag{3}$$

para todo polinômio poli(.) e todo $n$ suficientemente grande. Como no nosso cenário $l(n) = 1$ (o ciframento ocorre bit-a-bit), temos

$$\begin{aligned}\Pr[A(p(X_n), R_{l(n)}) = 1] &= \frac{1}{2}\Big(\Pr[A(p(X_n),1) = 1] + \Pr[A(p(X_n),0) = 1]\Big) \\ &= \frac{1}{2}\Big(\Pr[A(p(X_n), b(X_n)) = 1] + \Pr[A(p(X_n), b(X_n) \oplus 1) = 1]\Big).\end{aligned}$$

Substituindo então na eq. (3) tem-se

$$\begin{aligned}\mathrm{abs}(\Pr[A(p(X_n), b(X_n)) = 1] - \Pr[A(p(X_n), R_{l(n)}) = 1]) &= \\ \mathrm{abs}(\Pr[A(p(X_n), b(X_n)) = 1] - \Pr[A(p(X_n), b(n) \oplus 1) = 1])/2 &< \frac{1}{\mathrm{poli}(n)},\end{aligned}$$

que implica, trivialmente, a eq. (2), demonstrando a segurança polinomial de `CS-2`. □

Algumas observações cabem em relação a esta demonstração. A primeira é que ela é propositadamente simples por deslocar muito da sua complexidade para definições auxiliares: não especificamos como construir *hard-cores* de permutações (de fato, nem especificamos uma permutação a ser utilizada), e a segurança do criptossistema não depende diretamente da estrutura dos *hard-cores* ou da permutação. Em segundo lugar, note que esta demonstração de segurança, na verdade, não depende diretamente de o ciframento operar bit-a-bit, mas sim de que ele opere em blocos do contradomínio do *hard-core* $b(.)$ em questão. Usamos o exemplo em que este contradomínio é $\{0,1\}$ por facilidade de apresentação, mas nada impede que outros *hard-cores* sejam utilizados.

Tendo estas duas observações em mente, apresentamos a seguir uma versão do `CS-2` baseada no RSA, que chamaremos de `RSA-Aleatorizado` na discussão que segue, e que é demonstravelmente segura.

Lembramos que a permutação induzida pela Função-RSA é

$$F_{\mathrm{RSA}}((N,e),x) = x^e \pmod N.$$

Suponha agora que $F_{\mathrm{RSA}}$ é de fato uma permutação com segredo (isto é, que ela é de difícil inversão) e, adicionalmente, que o cálculo dos $m$ bits menos significativos de $x$ seja um *hard-core* de $F_{\mathrm{RSA}}$ para $|N| = k$. Podemos então construir um esquema análogo ao anterior que funciona em blocos de $m$ bits da seguinte maneira:

**Criptossistema 6.** *Instanciação do `CS-2` baseada no RSA* (`RSA-Aleatorizado`)

**Geração de chaves.** Selecionam-se aleatoriamente dois primos de $n$ bits, $p$ e $q$. Seja $N = pq$; note que $\phi(N) = (p-1)(q-1)$. Escolha aleatoriamente um par $(e,d)$ tal que $ed \equiv 1 \pmod{\phi(N)}$. A chave pública será o par $(N,e)$ e a chave privada será o par $(N,d)$.

**Ciframento.** Seja $M \in \{0,1\}^m$ a mensagem a ser cifrada. Escolha $r \xleftarrow{r} \{0,N-1\}$ e calcule o texto cifrado $C = (r^e \pmod M, \mathrm{BMS}_m(r) \oplus M)$, onde $\mathrm{BMS}_m(r)$ representa os $m$ bits menos significativos de $r$.

**Deciframento.** Seja $C = (\gamma_1, \gamma_2)$ o texto a ser decifrado. Através de $(N,d)$ calcula-se cada $m'_i = \gamma_2 \oplus (\gamma_1^d \pmod N)$.

**Teorema 5.** *Supondo que a extração dos m bits menos significativos da entrada seja um hard-core de $F_{RSA}$, o criptossistema* `RSA-Aleatorizado` *é polinomialmente seguro.*

A prova de segurança desta versão do RSA pode ser construída de forma análoga à segurança do `CS-2` e é deixada como exercício para o leitor.

Sabe-se que para $m \in O(\log n)$ a extração dos $m$ bits menos significativos é um *hard-core* de $F_{\mathrm{RSA}}$ [Chor e Goldreich, 1985]. Contudo, para valores assintoticamente maiores de $m$ a dificuldade da extração dos $m$ bits menos significativos ainda é um problema em aberto. Em particular, seria interessante usar algo próximo de $m = n/2$.

É importante destacar aqui que *o uso do RSA simples não é seguro* (uma vez que ele é determinístico). O uso do `RSA-Aleatorizado` para $m \in O(\log(n))$ é demonstravelmente seguro supondo que a Função-RSA induza uma permutação com segredo no seu domínio, todavia é ineficiente. O uso do `RSA-Aleatorizado` para $m = n/2$ é *possivelmente* seguro, mas este ainda é um problema em aberto. Logo, é preferível utilizar este último que possivelmente é seguro, dada uma conjetura razoável (que a extração de $n/2$ bits da entrada é um *hard-core* da Função-RSA), a utilizar o RSA simples que certamente é *inseguro*.

#### 3.3.1.4. Ciframento Probabilístico Eficiente

Em [Blum e Goldwasser, 1985] os autores apresentaram o primeiro criptossistema seguro realmente eficiente. As propostas existentes até então podem ser consideradas resultados de "plausibilidade", extremamente importantes em termos teóricos, mas que não traziam uma opção que fosse utilizável na prática.

Com este esquema os autores atingem eficiência comparável, e às vezes até superior, à do RSA. Ele, de alguma maneira, herda o conceito de utilizar um *hard-core* de uma permutação com segredo, como nos esquemas `CS-2` e `RSA-Aleatorizado`, mas usa um truque engenhoso para evitar a expansão de dados observada nestes esquemas: através da noção de *geradores pseudo-aleatórios*, algoritmos capazes de, a partir de uma *semente* aleatória, gerar uma seqüência de bits "pseudo-aleatória", indistinguível em tempo polinomial de uma seqüência verdadeiramente aleatória desde que a semente seja desconhecida. Sendo assim, para cifrar uma mensagem $M$ de tamanho $|M| = n$, usa-se uma semente de tamanho $k$ e a partir dela gera-se uma seqüência de $n$ bits pseudo-aleatórios usados para cifrar $M$.

Para facilitar a apresentação, manteremos o padrão de primeiro apresentar uma versão abstrata do esquema seguida de uma versão que apresenta a instanciação sugerida pelos autores. Na apresentação a seguir usamos a notação

$$p_\alpha^{i+1}(x) = p_\alpha(p_\alpha^i(x)) \quad \text{e} \quad p_\alpha^{-(i+1)}(x) = p_\alpha^{-1}(p_\alpha^{-i}(x)).$$

**Criptossistema 7.** *Blum-Goldwasser Abstrato* (BG−A)

**Geração de chaves.** Selecione uma permutação $p_\alpha$ com segredo associado $s^*$. A chave pública é (a descrição de) $p_\alpha$ e a chave privada é $s^*$.

**Ciframento.** Seja $M = m_0 m_1 \ldots m_{n-1}$ a mensagem a ser cifrada e seja $b^*(.) : \{0,1\}^k \to \{0,1\}$ um *hard-core* de $p_\alpha$. Selecione aleatoriamente um elemento $r$ do domínio de $p_\alpha$ e calcule o texto cifrado $C = (p^n(r), M \oplus G^n(r))$, onde

$$G_\alpha^n \stackrel{\text{def}}{=} b(r)||b(p_\alpha(r))||\ldots||b(p_\alpha^{n-1}(r)).$$

**Deciframento.** Seja $C = (\gamma_1, \gamma_2)$ o texto a ser decifrado. Utilizando $s^*$, calcule

$$M = \gamma_2 \oplus G_\alpha^{(|\gamma_2|)}(p_\alpha^{-|\gamma_2|}(\gamma_1)).$$

Como pode-se observar a seguir, o esquema é correto:

$$\begin{aligned} D_{s^*}(E_\alpha(M)) &= D_{s^*}(p_\alpha^{|M|}(r), M \oplus G_\alpha^{(|M|)}(r)) \\ &= (M \oplus G_\alpha^{|M|}(r)) \oplus G_\alpha^{|M|}(p_\alpha^{-|M|}(p_\alpha^{|M|}(r))) \\ &= M \oplus G_\alpha^{|M|}(r) \oplus G_\alpha^{|M|}(r) = M. \end{aligned}$$

Uma mensagem $M$ de $n$ bits é cifrada como uma string de $(n+k)$ bits neste esquema, gerando uma expansão de mensagem muito pequena e que tende a ser desprezível com o crescimento de $n$ (afinal, $k$ é constante). A segurança desta versão abstrata do esquema pode ser demonstrada com base nas propriedades das primitivas utilizadas.

*Demonstração.* Novamente, escolhemos a noção de segurança polinomial. É suficiente mostrar então que, para um par de mensagens arbitrárias $(M'_n, M''_n) \in \{0,1\}^{l(n)} \times \{0,1\}^{l(n)}$, as distribuições

$$\begin{aligned} D'_n &\stackrel{\text{def}}{=} (\alpha, p_\alpha^{|l(n)|}(X_n), M'_n \oplus G_\alpha^{l(n)}(X_n)) \text{ e} \\ D''_n &\stackrel{\text{def}}{=} (\alpha, p_\alpha^{|l(n)|}(X_n), M''_n \oplus G_\alpha^{l(n)}(X_n)), \end{aligned}$$

são polinomialmente indistinguíveis, onde $X_n$ é uma variável uniformemente distribuída em $\{0,1\}^n$.

Na construção da nossa demonstração usaremos, sem demonstrar, o seguinte lema:

**Lema 1.** *Sejam k e n inteiros. Para toda permutação com segredo $p_\alpha$ e todo $x \in D_i$, seja $G_\alpha^n(.)$ como definido anteriormente:*

$$G_\alpha^n(x) \stackrel{def}{=} b(x)||b(p_\alpha(x))||\ldots||b(p_\alpha^{n-1}(x)).$$

*Seja $X_n$ uma variável aleatória uniformemente distribuída no domínio de $p_\alpha$. Então, para todo $l(n) \in O(\text{poli}(n))$ as distribuições*

$$\{(G_\alpha^{l(n)}(X_n), p_\alpha^{l(n)}(X_n)))\}_{n\in\mathbb{N}} \qquad e \qquad \{(U_{l(n)}, p_\alpha^{l(n)}(X_n)))\}_{n\in\mathbb{N}}$$

*são indistinguíveis em tempo polinomial.*

Este lema nos mostra basicamente que $G_\alpha^n(.)$ é um gerador de bits pseudo-aleatórios: mesmo recebendo todas as informações públicas disponíveis, nenhum algoritmo polinomial consegue distinguir a saída de $G_\alpha^{l(n)}(.)$ de uma variável uniformemente distribuída $U_{l(n)}$.

Seguimos então definindo as distribuições

$$D_n \stackrel{\text{def}}{=} (p_\alpha^{l(n)}(X_n), M_n \oplus G_\alpha^{l(n)}(X_n)) \qquad \text{e} \qquad R_n \stackrel{\text{def}}{=} (p_\alpha^{l(n)}(X_n), M_n \oplus U_{l(n)}),$$

onde $U_{l(n)}$ é uma variável uniformemente distribuída em $\{0,1\}^{l(n)}$.

**Lema 2.** *Para toda seqüência de $M_i$'s, as distribuições $D_i$ e $R_i$ são indistinguíveis.*

*Demonstração.* A indistinguibilidade de $\{D_n\}_{n\in\mathbb{N}}$ e $\{R_n\}_{n\in\mathbb{N}}$ segue do lema 1. □

Para todos os possíveis pares $(M'_n, M''_n) \in \{0,1\}^{l(n)} \times \{0,1\}^{l(n)}$, definimos de forma análoga as distribuições

$$\begin{aligned}
D'_n &\stackrel{\text{def}}{=} (p_\alpha^{l(n)}(X_n), M'_n \oplus G_\alpha^{l(n)}(X_n)) \\
D''_n &\stackrel{\text{def}}{=} (p_\alpha^{l(n)}(X_n), M''_n \oplus G_\alpha^{l(n)}(X_n)) \\
R'_n &\stackrel{\text{def}}{=} (p_\alpha^{l(n)}(X_n), M'_n \oplus U_{l(n)}) \\
R''_n &\stackrel{\text{def}}{=} (p_\alpha^{l(n)}(X_n), M''_n \oplus U_{l(n)}).
\end{aligned}$$

Para provar que $D'_n$ e $D''_n$ são indistinguíveis, simplesmente aplicamos o lema 2 a $(D'_n, R'_n)$ e notamos que elas são indistinguíveis. Analogamente, $(D''_n, R''_n)$ são indistinguíveis. É fácil enxergar que $R'_n$ e $R''_n$ têm a mesma distribuição. Logo, $D'_n$ e $D''_n$ são (polinomialmente) indistinguíveis. □

**Instanciando.** Em [Blum e Goldwasser, 1985] os autores apresentam este esquema utilizando a operação de elevar ao quadrado módulo um inteiro de Blum como a permutação com segredo da definição de `BG-A`. Um inteiro de Blum é todo número $n$ igual ao produto de dois primos $p, q$ tais que $p \equiv q \equiv 3 \pmod 4$. Supondo que fatorar inteiros de Blum seja um problema difícil, a operação de elevar ao quadrado induz uma permutação com segredo sobre os resíduos quadráticos $\pmod n$. Esta instanciação do `BG-A`, referida aqui como `BG`, tem eficiência comparável à do RSA.

**Criptossistema 8.** *Blum-Goldwasser* (BG)

**Geração de chaves.** Selecione aleatoriamente dois primos $P$ e $Q$ de $n$ bits tais que $P \equiv Q \equiv 3 \pmod 4$. Seja $N = PQ$. Calcule então:

$$\begin{aligned} d_P &= ((P+1)/4)^{l(n)} \pmod{(P-1)} \quad (\in \{0,\ldots,P-2\}) \\ d_Q &= ((Q+1)/4)^{l(n)} \pmod{(Q-1)} \quad (\in \{0,\ldots,Q-2\}) \\ c_P &= Q(Q^{-1} \pmod P) \quad (\in \{0,\ldots,N-Q\}) \\ d_P &= P(P^{-1} \pmod Q) \quad (\in \{0,\ldots,N-P\}). \end{aligned}$$

A chave pública é $N$ e a chave privada $(P,Q,c_P,d_P,c_Q,d_Q)$.

**Ciframento.** Para cifrar uma mensagem $M \in \{0,1\}^{l(n)}$ usando a chave $N$, selecione $s_0 \xleftarrow{r} \{1,\ldots,N\}$. Para $i = 1,\ldots,l(n)+1$, calcule

$$s_i \leftarrow s_{i-1}^2 \pmod N \qquad \text{e} \qquad b_i = \text{BMS}(s_i),$$

onde $\text{BMS}(s)$ representa o bit menos significativo de $s$. O texto cifrado será $(s_{l(n)+1}, M \oplus (b_1||b_2||\ldots||b_{l(n)}))$.

**Deciframento.** Seja $C = (\gamma_1,\gamma_2)$ o texto a ser decifrado e seja $(P,Q,c_P,d_P,c_Q,d_Q)$ a chave privada. Comece por recuperar o valor de $s_1$ que é a $2^{l(n)}$-ésima raiz módulo $N$. Esta extração é tão eficiente quanto a extração de uma raiz quadrada:

$$\begin{aligned} s_P &= \gamma_1^{d_P} \pmod P \\ s_Q &= \gamma_1^{d_Q} \pmod P \\ s_1 &= c_P \cdot s_P + c_Q \cdot s_Q \pmod N. \end{aligned}$$

Lembrando que $\gamma_1 = s_1^{2^{l(n)}} \pmod N$, o resultado segue do TCR. Para todo $i = 1,\ldots,l(n)$, calcule $b_i = \text{BMS}(s_i)$ e $s_{i+1} \leftarrow s_i^2 \pmod N$. O texto em claro é $M = \gamma_2 \oplus b_1||b_2||\ldots||b_{l(n)}$.

É fácil observar que a eficiência deste esquema é comparável à do RSA. O ciframento requer $l(n)+1$ multiplicações modulares e o deciframento requer $l(n)+2$ multiplicações e duas exponenciações modulares. Supondo que exponenciações modulares têm um custo de $O(n)$ multiplicações, todo o processo de ciframento tem custo próximo de $2l(n)+3n$ multiplicações modulares. A versão aleatorizada do RSA apresentada aqui tem custo de $\lceil l(n)/n \rceil$ exponenciações modulares, aproximadamente $3l(n)$ multiplicações modulares (se $e = 3$, reduz-se a até $1{,}5l(n)$ multiplicações modulares).

A segurança do criptossistema BG é definida pelo teorema a seguir.

**Teorema 6.** *O criptossistema BG é seguro se a fatoração de inteiros de Blum for um problema difícil [Blum e Goldwasser, 1985].*

Enquanto não se sabe se a quebra do RSA é equivalente à fatoração, e se a conjetura que sustentaria a versão RSA-Aleatorizada-$n/2$ é verdadeira, a segurança

do BG deriva diretamente da dificuldade da fatoração, sem nenhuma suposição adicional, mantendo uma eficiência comparável à do RSA.

### 3.3.2. Ciframento e Adversários Ativos

Todas as demonstrações de segurança apresentadas até agora referem-se a ataques realizados de forma passiva: um adversário de posse de textos cifrados e informações públicas tenta descobrir alguma informação sobre as mensagens originais. Existem outros tipos de ataque, no entanto, onde adversários têm a capacidade de interagir com os usuários do sistema, requisitar deciframento de mensagens arbitrárias, e onde as demonstrações de segurança apresentadas aqui deixam de ser válidas. Estes cenários de ataque são importantes, e podem aparecer em muitas situações reais. Como lidar com eles? Basicamente, utilizamos três parâmetros para classificar os tipos de ataque:

1. *Que informação é conhecida pelo adversário?* Ele tem acesso apenas a textos cifrados ou a pares de texto em claro e cifrado?
2. *O adversário tem o poder de escolher a informação a que tem acesso?* Por exemplo, o adversário pode escolher o texto em claro a cuja versão cifrada terá acesso?
3. Em caso de poder escolher, *o adversário pode fazê-lo de forma adaptativa?* Ou seja, o adversário pode escolher os textos que deseja (de)cifrar durante o ataque ou precisa escolher todos *a priori*?

Dependendo então das respostas às perguntas acima obtemos os seguintes tipos de ataques: ataques de texto cifrado conhecido[7], ataques de texto em claro conhecido, ataques de texto em claro escolhido, ataques de texto cifrado escolhido, ataques adaptativos de texto em claro escolhido (CPA)[8] e ataques adaptativos de texto cifrado escolhido (CCA)[9].

Muitos destes cenários de ataque podem parecer artificiais, afinal por que algum usuário que valoriza o sigilo de suas mensagens se disporia a decifrar mensagens arbitrárias para um possível adversário (possibilitando assim um ataque adaptativo)? A verdade é que cenários em que isto pode ocorrer, por mais estranho que pareça, acontecem na prática. Portanto, é importante destacar a relevância da segurança contra estes ataques: eles podem surgir em muitas das situações onde esquemas de ciframento são utilizados.

**Demonstrações por jogos.** Na construção de provas de segurança contra ataques ativos é necessário, de alguma maneira, representar a possível interação entre usuário e adversário. A maneira mais difundida e provavelmente mais intuitiva de modelar esta interação é através de jogos. Digamos que queremos demonstrar que um certo criptossistema é CCA-seguro. Como sempre, utilizamos a noção de segurança polinomial. Propomos então o seguinte jogo entre duas partes: o adversário $A$, que tenta quebrar a segurança polinomial do criptossistema, e o simulador $S_A$ que tenta resolver algum problema difícil caso $A$ consiga quebrar o criptossistema. Para facilitar a discussão, definimos $A$ por dois

[7] As demonstrações de segurança que vimos até agora tratam deste tipo de segurança, a noção de segurança mais fraca dentre as definidas.

[8] Do inglês *Chosen-Plaintext Attack*. Geralmente este acrônimo se refere a ataques adaptativos.

[9] Do inglês *Chosen-Ciphertext Attack*. Geralmente este acrônimo se refere a ataques adaptativos.

algoritmos $(A_1, A_2)$ de modo que $A_1$ recebe os parâmetros do sistema e gera o par de mensagens-desafio, e posteriormente $A_2$ recebe este par de mensagens, tendo uma delas sido cifrada, e deve ser capaz de descobrir qual das duas foi cifrada. O jogo prossegue então da seguinte forma:

1. $S_A$ gera os parâmetros do sistema `params` e o par de chaves $(PK, SK)$, e executa $A(\texttt{params}, PK)$; se o ataque for de texto cifrado escolhido, $A$ deve ter acesso a um oráculo de deciframento (simulado por $S_A$);
2. $A_1$ executa um número polinomial de passos e devolve um par de mensagens $(m_0, m_1)$;
3. $S_A$ escolhe aleatoriamente um $i \in \{0,1\}$ e calcula $\alpha = E_{SK}(m_i)$;
4. $S_A$ executa $A_2(\texttt{params}, PK, m_0, m_1, \alpha)$; se o ataque for *adaptativo* de texto cifrado escolhido, $A_2$ também precisa ter acesso a um oráculo de deciframento (novamente, simulado por $S_A$).
5. $A_2$ executa um número polinomial de passos e retorna um chute $i'$.

$A$ vence o jogo se $i = i'$.

Levou muito tempo até que os pesquisadores conseguissem propor um esquema prático e demonstravelmente seguro contra adversários adaptativos. Apresentamos, a seguir, o primeiro tal criptossistema, originalmente proposto em [Cramer e Shoup, 1998]. A sua segurança depende de um problema, menos padrão do que a fatoração ou o logaritmo discreto, chamado de problema de Diffie-Hellman de Decisão. Definimos brevemente este problema antes de prosseguir com a apresentação do `Cramer-Shoup`.

Partindo do protocolo de [Diffie e Hellman, 1976] para estabelecimento de chaves podemos propor dois problemas aparentemente difíceis:

> **O Problema de Diffie-Hellman Computacional (`DHC`).** Esta versão é exatamente equivalente à quebra do protocolo de estabelecimente de chaves. Seja $\mathbb{G}$ o grupo onde o problema está definido e $g$ um gerador desse grupo. Dada a tupla $(g, g^a, g^b) \in \mathbb{G}^3$, o adversário deve calcular $g^{ab} \in \mathbb{G}$.
>
> **O Problema de Diffie-Hellman de Decisão (`DHD`).** Nesta versão do problema, o adversário precisa apenas *reconhecer* se o valor recebido é o valor correto: seja $\mathbb{G}$ o grupo onde o problema está definido e $g$ um gerador desse grupo. Dada a tupla $(g, g^a, g^b, g^c) \in \mathbb{G}^3$, o adversário deve responder se $g^{ab} \equiv g^c$.

Ambos os problemas, assim como o protocolo de estabelecimento de chaves, dependem da dificuldade do cálculo de logaritmos discretos. No entanto, de maneira semelhante à relação entre RSA e fatoração, não existe qualquer demonstração de que `DHD` ou `DHC` sejam equivalentes ao problema do logaritmo discreto. Inclusive, no caso geral, o `DHD` é estritamente mais fácil que o `DHC`, existindo inclusive uma área de criptografia que estuda as propriedades de grupos onde o `DHD` é fácil e o `DHC` é difícil, os chamados *Gap Diffie-Hellman Groups*, e propõe criptossistemas baseados nesta diferença (e.g. [Boneh e Franklin, 2003]). No entanto, quando se define o problema em $\mathbb{Z}_n^*$, para valores apropriados de $n$, não se sabe resolver `DHD` ou `DHC` sem calcular o logaritmo discreto.

O criptossistema de [Cramer e Shoup, 1998] é baseado no `ElGamal`, que pode ser demonstrado seguro contra adversários passivos caso o `DHD` seja difícil. Mais do que isso, ele é dotado de alguma complexidade extra que o torna seguro também contra adversários adaptativos. Faremos uma apresentação progressiva do `Cramer-Shoup` baseada no tratamento dado por [Katz, 2004]: começaremos propondo uma versão ligeiramente alternativa do `ElGamal` básico, que não muda significativamente suas propriedades de segurança (continua semanticamente seguro contra adversários passivos), mas introduz uma técnica de demonstração que será crucial na apresentação do `Cramer-Shoup` completo. Considere então a seguinte versão do criptossistema ElGamal:

**Criptossistema 9.** *ElGamal-A*

**Geração de Chaves.** Seja $\mathbb{G}$ um grupo de ordem $n$ onde o `DHD` é difícil e sejam $g_1, g_2$ geradores de $\mathbb{G}$ escolhidos aleatoriamente. Selecione $x, y \xleftarrow{r} \{1, \ldots, n-1\}$. Calcule $z = g_1^x g_2^y \pmod{n}$. A chave pública é $(g_1, g_2, z)$ e a chave privada é $(x, y)$.

**Ciframento.** Seja $M \in \{1, \ldots, n-1\}$ a mensagem a ser cifrada e seja $(g_1, g_2, z)$ a chave pública do destinatário. Escolha $r \xleftarrow{r} \{1, \ldots, n-1\}$. Calcule o texto cifrado $(g_1^r, g_2^r, z^r M)$.

**Deciframento.** Seja $(\gamma_1, \gamma_2, C)$ o texto cifrado e seja $(x, y)$ a chave privada do usuário. Calcule $M = C/\gamma_1^x \gamma_2^y$.

**Teorema 7.** *`ElGamal-A` é polinomialmente seguro contra adversários passivos se o `DHD` é difícil em* $\mathbb{G}$.

*Demonstração.* Suponha que um algoritmo $A = (A_1, A_2)$ consegue quebrar a segurança polinomial de `ElGamal-A`. Construímos então o algoritmo $S_A$ que consegue resolver o `DHD`. $S_A$ recebe como entrada uma tupla $(g_1, g_2, g_3, g_4)$ e deve ser capaz de descobrir se ela é uma tupla aleatória ou se é uma tupla de Diffie-Hellman. $S_A$ prossegue então da seguinte forma:

1. escolhe $x, y \xleftarrow{r} \{0, \ldots, n-1\}$;
2. calcula $h = g_1^x g_2^y$;
3. define $PK \stackrel{\text{def}}{=} (g_1, g_2, z)$ como chave pública;
4. executa $(m_0, m_1) \leftarrow A_1(PK)$, recebendo as duas mensagens-desafio;
5. escolhe $i \xleftarrow{r} \{0, 1\}$;
6. calcula $C = (g_3, g_4, g_3^x g_4^y . m_b)$;
7. executa $i' \leftarrow A_2(PK, C)$
   - se $i = i'$, $S_A$ devolve `ACEITA`;
   - caso contrário, $S_A$ devolve `FALHA`.

A estrutura da prova será basicamente a seguinte: primeiro, mostraremos que, se $S_A$ receber uma tupla de Diffie-Hellman como entrada ele simula perfeitamente um ataque real para $A$ e detecta a tupla com probabilidade relacionada à probabilidade de $A$ quebrar o criptossistema; por outro lado, se $S_A$ recebe uma tupla aleatória, $A$ não consegue qualquer informação sobre $i$ e, na melhor das hipóteses, tem 50% de chance de acerto.

**Lema 3.** *Se $S_A$ recebe uma tupla de Diffie-Hellman, a visão de A é a de um ataque real.*

*Prova do lema 3.* Se $(g_1, g_2, g_3, g_4)$ é uma tupla de Diffie-Hellman então existem $\alpha, \beta \in \mathbb{Z}_n$ tais que

$$(g_1, \quad g_2 \equiv g_1^{\alpha}, \quad g_3 \equiv g_1^{\beta}, \quad g_4 \equiv g_1^{\alpha\beta} \equiv g_2^{\beta}).$$

Logo, a chave pública e o texto cifrado têm a seguinte forma:

$$\begin{aligned} PK &= (g_1,\, g_2,\, z \equiv g_1^x g_2^y) \\ C &= (g_3 \equiv g_1^{\beta},\, g_4 \equiv g_2^{\beta},\, (g_1^{\beta})^x (g_2^{\beta})^y \cdot m_i) = (g_1^{\beta},\, g_2^{\beta},\, z^{\beta}.m_i), \end{aligned}$$

que é exatamente a distribuição esperada de uma execução "legítima" do protocolo. □

Observe que este lema implica que

$$\Pr[S_A \text{ "ACEITA"} \mid \text{é tupla DH}] = \Pr[i' = i \mid A \text{ quebra o criptossistema}].$$

**Lema 4.** *Se $S_A$ recebe uma tupla aleatória, A não tem qualquer informação sobre i, mesmo que A seja ilimitado computacionalmente.*

Um corolário imediato deste lema é que

$$\Pr[S_A \text{ "ACEITA"} \mid \text{é tupla aleatória}] < \frac{1}{2} + \frac{1}{p(n)}$$

para todo polinômio $p(.)$ e $n$ suficientemente grande. Seguimos então com a prova deste lema.

*Prova do lema 4.* Suponha que $S_A$ recebeu uma tupla aleatória como entrada. Existem $\alpha, \beta, \omega \in \mathbb{Z}_n$ tais que

$$(g_1,\, g_2 = g_1^{\alpha},\, g_3 = g_1^{\beta},\, g_4 = g_1^{\omega}).$$

Suponha que $(\omega \neq \alpha\beta)$ e $(\alpha \neq 0)$[10], o que é verdadeiro com imensa probabilidade. Isso implica que existem $r, r'$ tais que $g_3 = g_1^r$ e $g_4 = g_2^{r'}$. Analisemos agora o que $A$ sabe sobre $x$ e $y$. A partir da chave pública $PK = (g_1, g_2, h)$ sabe-se que $z = g_1^x g_2^y$ e que, portanto, $x$ e $y$ satisfazem

$$\log_{g_1} z = x + (\log_{g_1} g_2) \cdot y = x + \alpha y. \tag{4}$$

Para cada $x \in \mathbb{Z}_n$ existe um $y \in \mathbb{Z}_n$ único que satisfaz esta equação, logo existem exatamente $n$ pares $(x, y)$ válidos, igualmente prováveis.

Agora considere o termo $g_3^x g_4^y$. Analisemos a probabilidade de $g_3^x g_4^y = \mu$ para um $\mu \in \mathbb{G}$ arbitrário. Para que isto aconteça, temos que $\log_{g_1} \mu = \log_{g_1}(g_3^x g_4^y)$, ou seja:

$$\begin{aligned} \log_{g_1} \mu &= x.\log_{g_1} g_3 + y.\log_{g_1} g_4 \\ &= rx + r'\alpha y. \end{aligned} \tag{5}$$

[10] Observe que se $\alpha = 0$ então $g_2 \equiv 1$ e a tupla $(g_1, g_2, g_3, g_4)$ é de Diffie-Hellman se e somente se $g_4 \equiv g_3$.

Sejam $y_1 = \log_{g_1} h$ e $y_2 = \log_{g_1} \mu$. Então as equações (4) e (5) formam um sistema de equações lineares em $x$ e $y$ dado por $B\vec{x} = \vec{z}$, onde

$$B = \begin{pmatrix} 1 & \alpha \\ r & r'\alpha \end{pmatrix}, \quad \vec{x} = [xy]^T, \quad \vec{z} = [z_1 z_2]^T .$$

Lembrando que $r \neq r'$ e $\alpha \neq 0$, o sistema acima sempre tem uma solução única em $x, y$. Mas como $\mu$ é um elemento arbitrário do grupo, isso implica que todo $\mu' \in \mathbb{G}$ é possível e igualmente provável. Em outras palavras, dados $(g_1, g_2, g_3, g_4)$ e $z = g_1^x g_2^y$, para $x, y \in \mathbb{Z}_n$ escolhidos uniformemente, se $\log_{g_1} g_3 \neq \log_{g_2} g_4$, mesmo um algoritmo ilimitado computacionalmente não consegue prever o valor de $\mu = g_3^x g_4^y$ com probabilidade maior que $1/n$, já que todas as opções são igualmente prováveis.

Como, do ponto de vista de $A$, $g_3^x g_4^y$ é uniformemente distribuído em $\mathbb{G}$, ele não consegue obter nenhum informação sobre a mensagem cifrada e, conseqüentemente, sobre $i$. □

Combinando os dois lemas acima, sabemos que

$$X = \Pr[S_A \text{ "ACEITA"} \mid \text{é tupla DH}] = \Pr[i' = i \mid A \text{ quebra o criptossistema}], \text{ e}$$
$$Y = \Pr[S_A \text{ "ACEITA"} \mid \text{é tupla aleatória}] < \frac{1}{2} + \frac{1}{p(n)} .$$

Por conseguinte, a vantagem de $S_A$, $\varepsilon(k)$ é

$$\varepsilon(k) = X - Y = \lambda(k) ,$$

que é desprezível (em relação a $k$), provando o teorema. ■

Reiterando, o `ElGamal-A` tem as mesmas propriedades de segurança do `ElGamal` padrão, mas o utilizamos para apresentar a técnica de demonstração do lema 4, que será importante nos teoremas a seguir. Em seguida apresentamos mais um passo em direção ao `Cramer-Shoup`, uma versão simplificada do esquema que é segura somente contra ataques de texto escolhido não-adaptativos.

**Criptossistema 10.** *Cramer-Shoup-Lite*

**Geração de Chaves.** Seja $\mathbb{G}$ um grupo de ordem $n$ onde o `DHD` é difícil e sejam $g_1, g_2$ geradores de $\mathbb{G}$ escolhidos aleatoriamente. Selecione $x, y, a, b \xleftarrow{r} \{1, \ldots, n-1\}$. Calcule $z = g_1^x g_2^y \pmod n$ e calcule $c = g_1^a g_2^b \pmod n$. A chave pública é $(g_1, g_2, z, c)$ e a chave privada é $(x, y, a, b)$.

**Ciframento.** Seja $M \in \mathbb{G}$ a mensagem a ser cifrada e seja $(g_1, g_2, z, c)$ a chave pública do destinatário. Escolha $r \xleftarrow{r} \{1, \ldots, n-1\}$. Calcule o texto cifrado $(g_1^r, g_2^r, c^r, z^r M)$.

**Deciframento.** Seja $(\gamma_1, \gamma_2, \gamma_3, C)$ o texto cifrado e seja $(x, y, a, b)$ a chave privada do usuário. Se $(\gamma_3 = \gamma_1^a \gamma_2^b)$, $M = C/\gamma_1^x \gamma_2^y$; caso contrário, $M = \perp$.

**Teorema 8.** *O `Cramer-Shoup-Lite` é seguro contra ataques de texto escolhido não-adaptativos se o `DHD` for difícil em* $\mathbb{G}$.

*Demonstração.* A prova de segurança deste esquema é muito semelhante à prova de segurança do `ElGamal-A`: supomos a existência de um algoritmo $A = (A_1, A_2)$ capaz de quebrar o `Cramer-Shoup-Lite` e construímos um algoritmo $S_A$ que utiliza $A$ para resolver o `DHD`. O fator complicador aqui é que estamos lidando com um adversário ativo, então temos que garantir que as respostas às consultas de decifrамento não revelem nada sobre a simulação. $S_A$ prossegue da seguinte maneira:

1. escolhe $x, y, a, b \xleftarrow{r} \{0, \ldots, n-1\}$;
2. calcula $z = g_1^x g_2^y$ e $c = g_1^a g_2^b$;
3. faz a chave pública $PK = (g_1, g_2, z, c)$;
4. faz a chave privada $SK = (x, y, t, u)$;
5. executa $(m_0, m_1) \leftarrow A_1^{D_{SK}(.)}(PK)$, recebendo as duas mensagens-desafio;
6. escolhe $i \xleftarrow{r} \{0, 1\}$;
7. calcula $C = (g_3,\ g_4,\ g_3^a g_4^b,\ g_3^x g_4^y \cdot m_i)$;
8. executa $i' \leftarrow A_2(PK, C)$
9. se $i = i'$, $S_A$ retorna `ACEITA`;
10. caso contrário, $S_A$ retorna `FALHA`.

Denotamos por $A_1^{D_{SK}(.)}(.)$ a execução do algoritmo $A_1(.)$ com acesso ao oráculo $D_{SK}(.)$ que decifra mensagens utilizando a chave privada $SK$ de acordo com a especificação do criptossistema. A estrutura desta prova é bastante semelhante à da demonstração anterior, para o `ElGamal-A`. As principais mudanças ocorrem na demonstração do segundo lema abaixo.

**Lema 5.** *Se $S_A$ recebe uma tupla de Diffie-Hellman, a visão de A é a de um ataque real.*

A prova deste lema é completamente análoga à presente na demonstração de segurança do `ElGamal-A`. Obtemos, inclusive, o mesmo corolário:

$$\Pr[S_A \text{ "ACEITA"} \mid \text{é tupla DH}] = \Pr[i' = i \mid A \text{ quebra o criptossistema}].$$

**Lema 6.** *Se $S_A$ recebe uma tupla aleatória, A não tem qualquer informação sobre b, mesmo que A seja ilimitado computacionalmente, desde que A possa fazer apenas uma quantidade polinomial de consultas ao oráculo de decifрamento.*

Este lema, por sua vez, possui uma demonstração um pouco mais do que o seu análogo na demonstração de segurança anterior pois, aqui, temos que garantir que as consultas realizadas por $A$ não trazem nenhuma informação adicional. Perceba que se este lema for verdadeiro o teorema 8 está provado pelo mesmo argumento usado para finalizar a demonstração do teorema 7.

*Prova do lema 6.* Seja $(g_1, g_2, g_3, g_4)$ a tupla aleatória recebida por $S_A$. Podemos então escrevê-la como

$$(g_1,\ g_2 \equiv g_1^{\alpha},\ g_3 \equiv g_1^{\beta},\ g_4 \equiv g_1^{\omega}),$$

onde, com imensa probabilidade, $\alpha \neq 0$ e $\omega \neq \alpha\beta$ (assumimos esta situação no resto da prova). A partir da chave pública *PK*, *A* sabe que $z = g_1^x g_2^y$ e isso restringe $(x,y)$ de acordo com

$$\log_{g_1} z = x + (\log_{g_1} g_2).y = x + \alpha y, \tag{6}$$

novamente, como na demonstração anterior. Consideremos agora que informações *A* pode obter a partir de suas consultas de deciframento. Dividimos as consultas $(\gamma_1, \gamma_2, \gamma_3, C)$ realizadas por *A* em duas categorias: se $\log_{g_1} \gamma_1 \equiv \log_{g_2} \gamma_2$, então a consulta é considerada *legal*; caso contrário, ela é considerada *ilegal*. Prosseguimos então para mostrar dois fatos:

1. *A* só ganharia informação com a resposta para uma consulta *ilegal*;
2. consultas *ilegais* são rejeitadas com altíssima probabilidade.

Estes dois fatos provam que *A* não obtém qualquer informação adicional através de suas consultas de deciframento.

**Lema 7.** *A obtém informação extra sobre* $(x,y)$ *só se submeter uma consulta de deciframento* $(\gamma_1, \gamma_2, \gamma_3, C)$ *tal que:*

1. *o oráculo de deciframento não retorne* $\perp$*, e*
2. $\log_{g_1} \gamma_1 \neq \log_{g_2} \gamma_2$ *(a consulta for* ilegal*)*

Primeiro, suponha que o oráculo de deciframento retornou $\perp$ para um consulta. Isso significa que $\gamma_3$ não é da forma correta. Contudo, esta checagem não pode revelar qualquer informação extra sobre $(x,y)$, porque só envolve $a$ e $b$.

Suponha agora que *A* submete uma consulta tal que $\log_{g_1} \gamma_1 = \log_{g_2} \gamma_2 = \delta$, para algum $\delta$ arbitrário. Neste caso, de acordo com a resposta $M$ do simulador, *A* sabe que $M = C/\gamma_1^x \gamma_2^y$, o que gera a seguinte restrição nos valores de $x$ e $y$:

$$\begin{aligned} \log_{g_1} M &= \log_{g_1} C - (\log_{g_1} \gamma_1)x - (\alpha \log_{g_2} \gamma_2)y \\ &= \log_{g_1} C - \delta(x - \alpha y), \end{aligned}$$

que é linearmente dependente da equação (6). Logo, esta equação não introduz qualquer restrição aos valores de $x$ e $y$, o que nos permite concluir que consultas rejeitadas pelo oráculo e consultas legais não trazem qualquer informação adicional (sobre $x$ e $y$) para o adversário *A*. Falta-nos apenas provar que a probabilidade de uma consulta ilegal não ser rejeitada é desprezível, e a prova do teorema estará completa.

**Lema 8.** *A probabilidade de A submeter uma consulta de deciframento* $(\gamma_1, \gamma_2, \gamma_3, C)$ *tal que* $\log_{g_1} \gamma_1 \neq \log_{g_2} \gamma_2$ *e o oráculo de deciframento não a rejeitar é desprezível.*

Sejam $r_1 = \log_{g_1} \gamma_1$ e $r_2 = \log_{g_2} \gamma_2$. Para que o oráculo de deciframento não rejeite a consulta, *A* deve "prever" o valor $\gamma_3 = \gamma_1^a \gamma_2^b$. Mostramos a seguir que isso não pode ser feito com probabilidade não-desprezível. Considere o que *A* sabe sobre $(a,b)$. A partir da chave pública, sabe-se que $c = g_1^a g_2^b$ e, conseqüentemente,

$$\log_{g_1} c = a + \alpha b. \tag{7}$$

Seja $\gamma_3'$ um elemento arbitrário de $\mathbb{G}$. Temos que $\gamma_3' = \gamma_1^a \gamma_2^b$ se e somente se

$$\begin{aligned} \log_{g_1} \gamma_3' &= a \log_{g_1} \gamma_1 + b \log_{g_1} \gamma_2 \\ &= r_1 a + \alpha r_2 b. \end{aligned} \tag{8}$$

As equações (7) e (8) são linearmente independentes e têm solução única em termos de $a$ e $b$. Como $\gamma_3'$ é arbitrário, existe solução para qualquer $\gamma_3'$. Portanto $A$ tem probabilidade $1/n$ de acertar o $\gamma_3$ correto.

Esta análise é válida para a *primeira* consulta realizada por $A$. Perceba que cada consulta rejeitada adicional revela *alguma* informação adicional sobre $(a,b)$, em particular que $\gamma_3' \neq \gamma_1^a \gamma_2^b$: na melhor das hipóteses isso elimina uma possibilidade de $(a,b)$. Logo, após $q_d'$ consultas rejeitas, ainda existem $n - q_d'$ soluções possíveis para $(a,b)$. Se durante a execução do ataque forem realizadas um total de $q_d$ consultas ao oráculo de deciframento, temos então que a probabilidade de alguma delas não ser rejeitada é $q_d/(n-q_d)$. Como $q_d$ é polinomial em $k$, e $n$ é exponencial em $k$, temos que essa probabilidade é desprezível. □

Juntando os lemas 5 e 6 concluímos que $A$ não consegue mais informação sobre $(x,y)$ do que a implícita pela equação (6). Neste caso, um argumento análogo ao da prova de segurança do teorema 7 mostra que $g_3^x g_4^y$ é uniformemente distribuído (do ponto de vista de $A$) e que $A$ não obtém qualquer informação sobre $i$. Isto completa a prova do lema 7 e do teorema. ■

Estamos prontos para analisar agora a versão completa do `Cramer-Shoup`, conforme proposto em [Cramer e Shoup, 1998]. A grande diferença entre o cenário da prova anterior e o da seguinte é que $A$ agora é adaptativo, ou seja, ele pode fazer consultas ao oráculo de deciframento mesmo depois de receber o texto cifrado-desafio. Temos que garantir então que também nestas consultas ele não possa conseguir qualquer informação adicional, da mesma maneira que com as consultas antes do desafio ser feito. Para atingir isto, adicionam-se mais duas variáveis desconhecidas (para $A$) de maneira que o número de incógnitas permaneça maior do que o número de equações conhecidas. Observe que o `Cramer-Shoup` utiliza uma função de *hash* resistente a colisões, conforme a definição 6, *mas não supõe o modelo do oráculo aleatório*.

**Criptossistema 11.** *Cramer-Shoup*

**Geração de Chaves.** Seja $\mathbb{G}$ um grupo de ordem $n$ onde o `DHD` é difícil, seja $H : \{0,1\}^* \to \mathbb{G}$ uma função de *hash* resistente a colisões e sejam $g_1, g_2$ geradores de $\mathbb{G}$ escolhidos aleatoriamente. Selecione $x,y,a,b,a',b' \xleftarrow{r} \{1,\ldots,n-1\}$. Calcule $z = g_1^x g_2^y \pmod n$, $c = g_1^a g_2^b \pmod n$ e calcule $d = g_1^{a'} g_2^{b'} \pmod n$. A chave pública é $(g_1, g_2, z, c, d, H)$ e a chave privada é $(x,y,a,b,a',b')$.

**Ciframento.** Seja $M \in \{1,\ldots,n-1\}$ a mensagem a ser cifrada e seja $(g_1, g_2, z, c, d, H)$ a chave pública do destinatário. Escolha $r \xleftarrow{r} \{1,\ldots,n-1\}$. Seja $h = H(g_1^r, g_2^r, z^r M)$. Calcule o texto cifrado $(g_1^r, g_2^r, (cd)^{hr}, z^r M)$.

**Deciframento.** Seja $(\gamma_1, \gamma_2, \gamma_3, C)$ o texto cifrado, seja $(x,y,a,b,a',b')$ a chave privada do usuário e seja $h = H(\gamma_1, \gamma_2, C)$. Se $(\gamma_3 = \gamma_1^{a+ha'} \gamma_2^{b+hb'})$, $M = C/\gamma_1^x \gamma_2^y$; caso contrário, $M = \perp$.

**Teorema 9.** *O* `Cramer-Shoup` *é polinomialmente seguro contra adversários adaptativos se o* `DHD` *é difícil em* $\mathbb{G}$.

*Demonstração.* Procedemos de forma bastante semelhante à prova anterior. De fato, $S_A$ é definido de maneira idêntica, exceto pelo fato de que $A_2$ agora também tem acesso ao oráculo de deciframento:

1. escolhe uma função de *hash* resistente a colisões $H(.)$;
2. escolhe $x, y, a, b, a', b' \stackrel{r}{\leftarrow} \{0, \ldots, n-1\}$;
3. calcula $z = g_1^x g_2^y$, $c = g_1^a g_2^b$, e $d = g_1^{a'} g_2^{b'}$;
4. faz a chave pública $PK = (g_1, g_2, z, c, d, H)$;
5. faz a chave privada $SK = (x, y, a, b, a', b')$;
6. executa $(m_0, m_1) \leftarrow A_1^{D_{SK}(.)}(PK)$, recebendo as duas mensagens-desafio;
7. escolhe $i \stackrel{r}{\leftarrow} \{0, 1\}$;
8. calcula $h = H(g_3,\ g_4,\ g_3^x g_4^y \cdot m_i)$
9. calcula $C = (g_3,\ g_4,\ g_3^{a+ha'} g_4^{b+hb'},\ g_3^x g_4^y \cdot m_i)$;
10. executa $i' \leftarrow A_2^{D_{SK}(.)}(PK, C)$
11. se $i = i'$, $S_A$ retorna `ACEITA`;
12. caso contrário, $S_A$ retorna `FALHA`.

**Lema 9.** *Se $S_A$ recebe uma tupla de Diffie-Hellman, a visão de A é a de um ataque real. Conseqüentemente,*

$$\Pr[S_A \text{ "}\texttt{ACEITA}\text{"} \mid \textit{é tupla DH}] = \Pr[i' = i \mid \textit{A quebra o criptossistema}].$$

A prova deste lema é exatamente como na demonstração anterior. O lema seguinte, no entanto, tem uma demonstração mais complicada dado que $A$ agora é adaptativo.

**Lema 10.** *Se $S_A$ recebe uma tupla aleatória, A não tem qualquer informação sobre i, mesmo que A seja ilimitado computacionalmente, desde que A possa fazer apenas uma quantidade polinomial de consultas ao oráculo de deciframento. Logo,*

$$\Pr[S_A \text{ "}\texttt{ACEITA}\text{"} \mid \textit{é tupla aleatória}] < \frac{1}{2} + \frac{1}{p(n)}.$$

*Prova do lema.* A estrutura geral desta prova é semelhante à da prova do lema 6 e assumimos o conhecimento desta última na apresentação da prova atual. Novamente, nosso objetivo é demonstrar que

1. $A$ não consegue fazer qualquer consulta "ilegal" que não é rejeitada pelo oráculo, e
2. que $A$ não consegue obter qualquer informação adicional a não ser através de consultas "ilegais".

Como o ponto 2 é exatamente igual ao `Cramer-Shoup-Lite`, focaremos aqui o ponto 1. Vejamos o que $A$ consegue de informação sobre $(a,b,a',b')$. A partir da chave pública $PK$, sabe-se que:

$$\log_{g_1} c = a + b.\alpha; \quad (9)$$

$$\log_{g_1} d = a' + b'.\alpha, \quad (10)$$

onde, como anteriormente, $\alpha = \log_{g_1} g_2$. Sejam $g_3 = g_1^r$ e $g_4 = g_2^{r'}$. Novamente, com imensa probabilidade, $r \neq r'$. Perceba que a análise de possíveis consultas realizadas durante a execução de $A_1(.)$ é análoga à análise presente na demonstração anterior. Vamos nos preocupar, portanto, com as consultas realizadas por $A_2(.)$. Quando $A$ recebe o texto cifrado-desafio, $(\gamma_1^* = g_3,\ \gamma_2^* = g_4,\ \gamma_3^* = g_3^{a+ha'} g_4^{b+hb'},\ C^* = g_3^x g_4^y \cdot m_i)$, sabe que:

$$\log_{g_1} \gamma_3^* = (a+ha')r + (b+hb')\alpha r'. \quad (11)$$

Por um argumento análogo ao presente na demonstração anterior, consultas "legais" (i.e., com $\log_{g_1} \gamma_1 = \log_{g_2} \gamma_2$) não trazem informação adicional para $A$. Queremos mostrar então que qualquer consulta $(\gamma_1, \gamma_2, \gamma_3, C)$ ilegal será rejeitada com imensa probabilidade pelo oráculo. Analisemos três casos para a consulta $(\gamma_1, \gamma_2, \gamma_3, C)$, lembrando que $A$ não pode submeter a consulta $(\gamma_1, \gamma_2, \gamma_3, C) = (\gamma_1^*, \gamma_2^*, \gamma_3^*, C^*)$:

**Caso 1.** Se $(\gamma_1, \gamma_2, C) = (\gamma_1^*, \gamma_2^*, C^*)$, mas $\gamma_3 \neq \gamma_3^*$, a consulta certamente é rejeitada (porque o valor do *hash* não será correto).

**Caso 2.** Se $(\gamma_1, \gamma_2, C) \neq (\gamma_1^*, \gamma_2^*, C^*)$, mas $H(\gamma_1, \gamma_2, \gamma_3) = H(\gamma_1^*, \gamma_2^*, \gamma_3^*)$ então $A$ encontrou uma colisão na função de *hash* (e como a função é resistente a colisões, isso só pode acontecer com probabilidade desprezível).

**Caso 3.** Se $H(\gamma_1, \gamma_2, C) \neq H(\gamma_1^*, \gamma_2^*, C^*)$ então uma análise mais cuidadosa, feita a seguir, é necessária.

Sejam $h^* = H(\gamma_1^*, \gamma_2^*, C^*)$, e $h = H(\gamma_1, \gamma_2, C)$. Sejam $\log_{g_1} \gamma_1 = r_1$ e $\log_{g_2} \gamma_2 = r_2$. Já que estamos tratando de consultas ilegais, sabemos que $r_1 \neq r_2$. Vamos nos focar na primeira consulta ilegal feita por $A$. Esta não será rejeitada se

$$\log_{g_1} \gamma_3 = (a+ha')r_1 + (b+hb')\alpha r_2.$$

Esta equação é linearmente independente das equações (9), (10) e (11) em relação às incógnitas $(a,b,a',b')$. De maneira semelhante à prova anterior, existem $n$ possíveis, e igualmente prováveis, $\gamma_3 \in \mathbb{G}$, então a primeira consulta ilegal será aceita com probabilidade apenas $1/n$. Continuando então de maneira análoga, chega-se à conclusão de que consultas ilegais não são rejeitadas pelo oráculo com probabilidade desprezível. □

O teorema segue diretamente dos lemas acima. ■

Isto conclui então a prova de segurança do `Cramer-Shoup`. Veja em particular que, apesar de fazer uso de uma função de *hash*, a prova de segurança supõe apenas a resistência a colisões, e não uma função de *hash* ideal (como no modelo

do oráculo aleatório, discutido em §3.4). O `Cramer-Shoup` foi assim o primeiro criptossistema prático e seguro contra adversários adaptativos proposto na literatura. Apesar de ser baseado na dificuldade do problema de Diffie-Hellman de Decisão, uma suposição razoavelmente forte, ele representou um resultado de grande importância para a criptografia moderna.

### 3.3.3. Assinaturas

Depois de desenvolver com certo de grau de detalhe noções de segurança para ciframento de chave pública, voltaremos nossa atenção para o problema de gerar assinaturas digitais e avaliar a sua segurança. O que significa "ser seguro" para um esquema de assinaturas? A noção intuitiva de segurança é a de "não-falsificação": estudaremos então o quão difícil é falsificar uma assinatura numa mensagem arbitrária. A primeira divisão importante é o quanto de informação um possível adversário tem sobre o usuário (digamos, Alice) cujas assinaturas deseja falsificar:

> **Ataque apenas com a chave.** O adversário conhece apenas os parâmetros do sistema e a chave pública de Alice.
>
> **Ataque de mensagem conhecida.** O adversário conhece também uma lista de mensagens anteriormente assinadas por Alice.
>
> **Ataque de mensagem escolhida.** O adversário, além das informações acima, tem acesso a um oráculo de assinatura, a quem ele pode submeter mensagens arbitrárias e que devolve uma assinatura de Alice válida na mensagem.

A outra grande classificação que se pode fazer é: o que, exatamente, é considerado "sucesso" para o adversário?

> **Quebra total.** O adversário consegue descobrir a chave privada de Alice.
>
> **Falsificação *universal*.** O adversário é capaz de gerar falsificações para mensagens arbitrariamente escolhidas sem ter observado uma assinatura de Alice para esta mensagem.
>
> **Falsificação *existencial*.** O adversário é capaz de gerar um par assinatura-mensagem $(\sigma, M)$, mas não é capaz de determinar nada *a priori* sobre a mensagem.

Observe que, enquanto concentramos nosso tratamento inicial da segurança de criptossistemas em adversários passivos, nosso tratamento de esquemas de assinaturas será focado em segurança contra adversários adaptativos. Definimos então, formalmente, o que daqui para frente consideraremos um esquema de assinaturas (existencialmente) seguro (contra ataques de mensagem escolhida).

**Definição 5.** *Seja $A$ um algoritmo polinomial arbitrário com acesso a um oráculo $O$. Denotamos por $Q_A^O(x)$ as consultas feitas por $A$ ao oráculo $O$ quando recebe $x$ como entrada. Seja $A^O(x)$ a saída de $A$ quando recebe $x$ como entrada. Um esquema de assinatura $(G,S,V)$ é seguro se para todo algoritmo polinomial $A$, todo polinômio $p(.)$, e todo $k$ suficientemente grande*

$$\Pr\left[(V_v(\sigma,M)=1)\wedge(M\notin Q_M^{S_s}(v)) \mid (\sigma,M)\leftarrow A^{S_s}(v)\right] < \frac{1}{p(k)},$$

*onde* $(s,v) \leftarrow G(1^k)$.

Um maneira mais intuitiva de formular a definição de segurança acima segue:

1. Seja $(G,S,V)$ um esquema de assinatura como na definição 3.
2. Dado um par de chaves $(s,v)$ gerado de acordo com o parâmetro de segurança em questão ($k$).
3. Nenhum algoritmo $A$ que receba a chave pública $v$ como parâmetro e tenha acesso a um oráculo que calcule assinaturas corretamente ($S_s$) pode ser capaz de gerar, com probabilidade não-desprezível, um par assinatura-mensagem $(\sigma,M)$ tal que:
   - o par seja aceito pelo algoritmo de verificação ($V_v(\sigma,M) = 1$);
   - $M$ nunca tenha sido consultada ao oráculo ($M \notin Q_M^{S_s}(v)$).

Perceba que, quando se trata de esquemas de assinaturas, a relevância de adversários ativos é muito mais clara do que no caso de esquemas de ciframento: naturalmente um possível adversário terá acesso a várias mensagens assinadas por um dado usuário pois estas costumam estar publicamente disponíveis (enquanto textos decifrados por usuários não são, em geral, facilmente disponibilizadas). Portanto realmente não faz muito sentido pensar em segurança contra adversários passivos no contexto de esquemas de assinatura.

#### 3.3.3.1. *"Hash and Sign"*

Não discutiremos nenhum esquema de assinatura concreto até abordarmos o modelo do oráculo aleatório. No entanto, discutiremos um resultado muito interessante que apóia a prática comum de assinar o *hash* de uma mensagem. Em primeiro lugar, vamos definir precisamente o tipo de função de *hash* ao qual nos referimos na discussão a seguir.

**Definição 6.** *(Funções de hash resistentes a colisões) Seja* $l : \mathbb{N} \to \mathbb{N}$. *Uma coleção de funções* $\{h_s : \{0,1\}^* \to \{0,1\}^{l(|s|)}\}_{s\in\{0,1\}^*}\}$ *é chamada de* `hash resistente a colisões` *se:*

1. ***Cálculo eficiente.*** *Existe um algoritmo polinomial que, dados s e x, calcula* $h_s(x)$.
2. ***Difícil de gerar colisões.*** *Diz-se que o par* $(x,x')$ *forma uma colisão na função h se* $h(x) = h(x')$ *mas* $x \neq x'$. *Para que h seja resistente a colisões, tem-se que para todo algoritmo polinomial A, todo polinômio* $p(.)$ *e todo n grande o suficiente,*

$$\Pr[A_{h_s}(1^n) \text{ é uma colisão de } h_s] < \frac{1}{p(n)},$$

*onde a probabilidade é tomada em relação à escolha de* $h_s$ *e em relação às escolhas aleatórias de* $A(.)$.

Definimos então esquemas de assinaturas *de tamanho restrito*. Intuitivamente, um esquema de assinaturas $l$-restrito só permite a assinatura de documentos de tamanho $l(n)$, onde $n$ é o parâmetro de segurança do sistema.

**Definição 7.** *(Esquema de assinaturas para documentos de tamanho fixo) Seja* $l : \mathbb{N} \to \mathbb{N}$. *Um esquema de assinatura l-restrito é uma tupla* $(G,S,V)$ *de algoritmos probabilísticos de tempo polinomial tais que, analogamente à definição 3:*

1. *ao receber* $1^n$ *o algoritmo G retorna um par de strings;*
2. *para todo n e todo par* $(s,v)$ *imagem de* $G(1^n)$*, para todo* $\alpha \in \{0,1\}^{l(n)}$*, os algoritmos S e V satisfazem*

$$\Pr[V_v(\alpha, S_s(\alpha)) = 1] = 1\,.$$

*Tal esquema é considerado* seguro *se preenche os requisitos (análogos aos) presentes na definição 5 para adversários restritos a fazer consultas de tamanho* $l(n)$ *e geram uma falsificação* $\varsigma = (M,\sigma)$ *onde* $|M| = l(n)$*.*

Faz todo o sentido, na busca de um esquema de assinatura seguro, o projeto inicial de um esquema que opere em blocos de tamanho fixo, de maneira semelhante ao que fizemos para criptossistemas: posteriormente pode-se tentar estender este esquema para assinar mensagens arbitrárias. Trataremos nesta seção de como fazer esta extensão, ou seja, suponha que você conseguiu um esquema de assinatura seguro (contra ataques de mensagem escolhida) mas apenas para mensagens de tamanho $l(n)$ (onde, digamos, $l(n) = n$). Como utilizá-lo para assinar mensagens de tamanho arbitrário?

A prática comum é utilizar um função de *hash*: escolhemos uma função de *hash* $H(.)$ que mapeia mensagens de tamanho arbitrário para strings de $l(n)$ bits e, dada $M$ para ser assinada, assinamos $H(M)$. Este é o conhecido paradigma do `Hash-And-Sign`. Provamos a seguir que, se a função $H(.)$ for resistente a colisões, o paradigma do `Hash-And-Sign` provê um esquema de assinaturas seguro.

**Esquema de Assinatura 1.** *Assinaturas sem restrição de tamanho* (`Hash-And-Sign`)

Sejam $l$ e $(G,S,V)$ como definição 7 e $H_n : \{0,1\}^* \to \{0,1\}^{l(n)}$ uma função de *hash* resistente a colisões como na definição 6. Construímos então o esquema de assinaturas $(G',S',V')$ a seguir:

**Geração de Chaves** $(G')$**.** Recebendo $1^n$ como entrada, $G'$ faz $(s,v) \leftarrow G(1^n)$. Escolha $H_n : \{0,1\}^* \to \{0,1\}^n$ e devolva o par $((H_n,s),(H_n,v))$.

**Assinatura** $(S')$**.** Seja $(H_n,s)$ a chave de assinatura e seja $M\{0,1\}^*$ a mensagem a ser assinada. $S'$ faz $\sigma \leftarrow S_s(H_n(M))$ e devolve $\sigma$.

**Verificação** $(V')$**.** Seja $(H_n,v)$ a chave de verificação e seja $(M,\sigma)$ o par mensagem-assinatura. $V'$ devolve o mesmo que $V_v(H_n(M),\sigma)$.

**Teorema 10.** *Suponha que* $(G,S,V)$ *é um esquema de assinaturas l-restrito seguro. Suponha que* $H_n$ *é uma função de hash livre de colisões. Então* $(G',S',V')$*, como definido acima, é um esquema de assinatura seguro sem restrição no tamanho das mensagens assinadas.*

*Demonstração.* Suponha que exista um adversário $A$ capaz de atacar o esquema $(G',S',V')$. Construímos então um algoritmo $S_A$ que utiliza $A$ e que ou forja assinaturas do esquema $l$-restrito $(G,S,V)$ ou gera colisões na função de *hash* $H_n(.)$. $S_A$ tem tempo de execução e probabilidade de sucesso relacionados aos de $A$. Descrevemos $S_A$ a seguir:

1. $S_A$ recebe como entrada uma chave pública $PK = v$ do esquema $l$-restrito;

2. escolhe uma função de *hash* aleatória $H_n$;
3. executa $A^{S_{s,H_n}}(H_n, v)$; (o oráculo $S_{s,H_n}$ é descrito a seguir)
4. $A$ devolve um par mensagem assinatura $(M^*, \sigma^*)$ no esquema "complexo" $(G', S', V')$;
5. $S_A$ devolve o par $(H_n(M^*), \sigma^*)$ como uma falsificação do esquema $l$-restrito.

As distribuições das chaves simuladas por $S_A$ são exatamente como num ataque real. A simulação do oráculo $S_{s,H_n}$ é feita da seguinte forma (lembre que $S_A$, que é um adversário do esquema $l$-restrito, tem acesso a um oráculo $S_s$ que gera assinaturas para este esquema):

> $S_{s,H_n}(M)$. $S_A$ simplesmente utiliza o seu oráculo para simular este, devolvendo $S_s(H(M))$.

Suponha agora que com probabilidade $\varepsilon'(n)$ $A$ consegue gerar uma falsificação do esquema complexo. Seja $M_i$ a $i$-ésima consulta realizada por $A$ ao oráculo $S_{s,H_n}$ e $\sigma_i$ a respectiva resposta. Lembre que $(M^*, \sigma^*)$ é saída de $A$. Analisemos dois casos:

> **Caso 1.** $H_n(M^*) \neq H_n(M_i)$ para todo $i$. Neste caso o par $(H(M^*), \sigma^*)$ é uma falsificação válida para o esquema $l$-restrito porque este valor de *hash* $(H(M^*))$ não havia sido consultado ainda ao oráculo $S_s$.
>
> **Caso 2.** $H_n(M^*) = H_n(M_i)$ para algum $i$. Neste caso, conseguimos uma colisão da função de *hash*.

Seja $S_A^*$ o evento de o caso 1 acontecer e seja $C_A^*$ o evento de o caso 2 acontecer. Temos então que

$$\Pr[A^*] = \Pr[S_A^*] + \Pr[C_A^*].$$

Como a função de *hash* é resistente a colisões e o esquema $l$-restrito é seguro, temos que

$$\Pr[S_A^*] + \Pr[C_A^*] < \frac{1}{p(n)},$$

para $n$ suficientemente grande. Isso mostra que $\Pr[A^*]$ é desprezível em $n$, contradizendo a hipótese de que $A$ era capaz de atacar o esquema (com probabilidade não-desprezível). □

Perceba que, na realidade, não fornecemos um esquema de assinatura com a construção acima. Provamos apenas que, se existir um esquema de assinatura seguro para mensagens de tamanho $l(n)$ e uma função de *hash* resistente a colisões com imagem em $\{0,1\}^{l(n)}$, podemos estender este esquema para assinar mensagens de tamanho arbitrário de maneira extremamente eficiente, pagando apenas o preço do *hash*. Este fato justifica, de alguma maneira, a técnica largamente utilizada de assinar o *hash* de mensagens: o problema é que, na prática, não são utilizados esquemas demonstravelmente seguros, invalidando assim a construção geral do `Hash-And-Sign`.

### 3.3.4. Próximos Passos

Os anos 80 trouxeram importantíssimas contribuições teóricas para a criptografia moderna. A formalização de noções fortes de segurança e o projeto de esquemas que as atingiam proporcionaram à criptografia uma base sólida sobre a qual continuar o seu desenvolvimento. O problema é que quando se trata da ameaça de adversários ativos estes resultados geralmente se restringiam a resultados de possibilidade: os esquemas propostos estavam longe de serem práticos. As técnicas que realmente se utilizavam na prática, por outro lado, constituíam-se principalmente de heurísticas, esquemas que conquistaram a confiança geral da comunidade pela incapacidade dos pesquisadores em quebrá-los.

Claramente o grande desafio para os pesquisadores no início dos anos 90 seria a aproximação entre os esquemas efetivamente utilizados na prática e as noções fortes de segurança desenvolvidas nos anos anteriores, especificamente quando se trata de adversários ativos (já que, como vimos aqui, existiam esquemas eficientes e demonstravelmente seguros contra adversários passivos). Estudaremos a seguir um paradigma de projeto de algoritmos criptográficos que tem esta aproximação entre a prática e a teoria como principal objetivo e foi largamente adotado ao longo da década de 90: o paradigma do oráculo aleatório.

## 3.4. O Paradigma do Oráculo Aleatório

O paradigma do oráculo aleatório surgiu basicamente como uma tentativa de formalizar um conjunto de práticas heurísticas utilizadas pela comunidade criptográfica para lidar com adversários adaptativos. O uso de funções de *hash* neste cenário tornou-se cada vez mais corriqueiro, mas apenas a suposição de que estas seriam resistentes a coliões não parecia ser suficiente para provar muitas das propriedades que se esperavam delas e que, ao menos aparentemente, estas efetivamente possuíam na prática. A intuição geral era a de que uma boa função de *hash* se comportaria efetivamente como uma função pseudo-aleatória, não permitindo o cálculo de qualquer correlação entre suas entradas e saídas. Seguindo essa linha heurística de pensamento, propõe-se que funções de *hash* sejam encaradas como caixas-pretas que implementam funções efetivamente aleatórias, apesar de estas certamente não o serem.

No paradigma do oráculo aleatório o projeto de protocolos segue portanto três passos:

1. suponha que todos os potenciais participantes do protocolo têm acesso a um oráculo público e aleatório;
2. prove que o protocolo é correto (e seguro) neste *Modelo do Oráculo Aleatório*;
3. instancie o protocolo na prática simulando o oráculo aleatório com um objeto tal como uma função de *hash*.

Certamente não se pode afirmar que a demonstração obtida no passo (2) prova que o protocolo é efetivamente seguro na prática. Espera-se, no entanto, que quaisquer vulnerabilidades que possam existir sejam problemas na instanciação (i.e., escolha de uma função de *hash* inapropriada) e não uma falha "estrutural" do protocolo. Divide-se então o problema de projetar um protocolo complexo em (1) projetar uma versão "ideal" do protocolo e (2) projetar uma boa implementação para o oráculo aleatório. Este paradigma

traduz, de alguma maneira, uma opinião geral da comunidade de que uma boa função de *hash* não deve possibilitar a detecção de correlações entre suas saídas, agindo, até certo ponto, como uma função pseudo-aleatória. Nas próximas subseções discutiremos alguns dos protocolos que foram desenvolvidos neste modelo "ideal" e, finalmente, discutiremos as principais objeções e controvérsias que cercam o paradigma do oráculo aleatório.

### 3.4.1. Ciframento no Modelo do Oráculo Aleatório

Na seção anterior, vimos alguns exemplos de criptossistemas polinomialmente seguros, mas sempre contra adversários passivos. Agora mostraremos que, no modelo do oráculo aleatório, existe um criptossistema extremamente eficiente e que é polinomialmente seguro contra ataques de texto cifrado escolhido. Ele pode ser encarado como uma extensão do criptossistema 7 onde, ao invés de sucessivas aplicações da permutação $p$, utilizamos uma função de *hash* ($G$) como gerador pseudo-aleatório (uma vez que estamos no modelo do oráculo aleatório) e utilizamos uma outra função de *hash* ($H$) para "validar" textos cifrados legítimos: esta validação protege o esquema de adversários adaptativos. Este criptossistema, assim como sua prova de segurança, foi originalmente apresentado em [Bellare e Rogaway, 1993].

**Criptossistema 12.** *Bellare-Rogaway contra ataques ativos* (BR)

**Geração de chaves.** Seja $p : \mathbb{G} \to \mathbb{G}$ uma permutação com segredo $s^*$ e sejam $G : \{0,1\}^* \to \{0,1\}^\infty$ e $H : \{0,1\}^* \to \{0,1\}^k$ duas funções aleatórias. A chave pública $PK$ será $(p, G, H)$ e a chave privada $SK$ será $s^*$.

**Ciframento.** Seja $M \in \{0,1\}^*$ a mensagem a ser cifrada e seja $PK = (p, G, H)$ a chave pública do destinatário. Escolha $r \xleftarrow{r} \mathbb{G}$. O texto cifrado será então $C = p(r)||x \oplus G(r)||H(rx)$.

**Deciframento.** Seja $C = \gamma_1||\gamma_2||\gamma_3$ o texto a ser decifrado. Utilizando $s^*$, calcula-se $r' = p^{-1}(\gamma_1)$. Seja então $X = \gamma_2 \oplus G(r')$. Se $H(r'||X) = \gamma_3$, então $M = X$; caso contrário, $M = \perp$. Devolva $M$.

**Teorema 11.** *O criptossistema* BR *é seguro contra ataques de texto cifrado escolhido.*

*Demonstração.* Seja $A = (A_1, A_2)$ um adversário capaz de quebrar o BR. Construímos então um algoritmo $S_A$ que utiliza $A$ para inverter a permutação $p$. $S_A$ funcionará da seguinte maneira:

1. recebe como entrada $(\mathbb{G}, p, Y)$ e deve encontrar $x \in \mathbb{G}$ tal que $Y = p(x)$;
2. executa $(m_0, m_1) \leftarrow A_1(PK)$;
3. escolhe $i \xleftarrow{r} \{0,1\}$;
4. calcula $C = Y||\gamma_2||\gamma_3$ para $\gamma_2 \xleftarrow{r} \{0,1\}^{|m_0|}$ e $\gamma_3 \xleftarrow{r} \{0,1\}^k$;
5. executa $i' \leftarrow A_2(PK, C)$;

Ao longo de todo o ataque, $S_A$ precisa simular os seguintes oráculos:

**G($r_i$).** Se $p(r_i) = Y$, $S_A$ pára e devolve $r_i$. Caso contrário, devolve uma string aleatória do tamanho apropriado.

**H($r_i||M_i$).** Se $p(r_i) = Y$, $S_A$ pára e devolve $r_i$. Caso contrário, devolve uma string aleatória do tamanho apropriado.

**Decifra($C_i = (\gamma_1, \gamma_2, \gamma_3)$).** $S_A$ busca, entre as consultas já feita a $G(r)$ e $H(r||x)$, por $(r', x')$ tais que:

- $H(r'||x') = \gamma_3$;
- $p(r') = \gamma_1$;
- $G(r') \oplus x' = \gamma_2$.

Caso um par $(r', x')$ apropriado seja encontrado, o oráculo devolve $x'$; caso contrário, devolve `inválido`.

Perceba então que $S_A$ tem sucesso quando $A$ faz uma consulta que contenha $r_i$, tal que $p(r_i) = Y$, a um dos oráculos. Precisamos então mostrar duas coisas:

1. $S_A$ simula corretamente uma execução legítima do criptossistema;
2. a probabilidade de $A$ conseguir adivinhar $i$ mas não fazer a consulta que inverte $Y$ é essencialmente $1/2$.

Para estruturar um pouco melhor a prova, vamos dividi-la em lemas.

**Lema 11.** *$S_A$ simula corretamente uma execução legítima do criptossistema.*

*Prova do Lema 11.* Os oráculos $G(.)$ e $H(.)$ são simulados perfeitamente. O oráculo de deciframento também é correto, exceto quando devolve "`inválido`" para uma consulta válida, i.e., onde $(\gamma_1, \gamma_2, \gamma_3)$ são tais que

$$\gamma_3 = H(p^{-1}(\gamma_1)||\gamma_2 \oplus G(p^{-1}(\gamma_1))) . \tag{12}$$

Seja $D_{\text{err}}$ este evento em que o oráculo devolve "`inválido`" erroneamente: exceto quando $D_{\text{err}}$ ocorre, a simulação é correta. Temos então que

$$\Pr[\, A^* \mid \overline{D_{\text{err}}} \,] = \Pr[\mathbf{SUCC}_A] = \lambda(k) + \frac{1}{2} ,$$

onde $\Pr[A^*]$ é a probabilidade de sucesso de $A$ no ambiente simulado por $S_A$, e $\Pr[\mathbf{SUCC}_A]$ é a probabilidade de sucesso de $A$ num ataque real. Perceba, no entanto, que por causa da aleatoriedade de $H(.)$ e $G(.)$, a probabilidade de $D_{\text{err}}$ ocorrer é igual à probabilidade de $A$ adivinhar corretamente a saída destas funções aleatórias sem as consultar (de maneira a satisfazer a eq. (12)). Ou seja,

$$\Pr[D_{\text{err}}] \leqslant q_d 2^{-k} ,$$

onde $q_d$ é o total de consultas feitas ao oráculo de deciframento. Como $q_d$ é polinomial em $k$, essa probabilidade é desprezível. Logo, com imensa probabilidade, a simulação executada por $S_A$ é correta. □

**Lema 12.** *A probabilidade de A conseguir adivinhar i mas não fazer a consulta que inverte Y é essencialmente* $1/2$.

*Prova do Lema 12.* Devido à aleatoriedade de $G(.)$ e $H(.)$, $\gamma_2 = G(r) \oplus m_i$ é essencialmente uma cifra perfeita: a única maneira através da qual $A$ pode obter *qualquer* informação sobre qual $m_i$ foi cifrada é consultando $r$ tal que $p(r) = \gamma_1$. Seja $S_A^*$ o evento

em que $S_A$ inverte com sucesso a permutação $p$ (porque $A$ realizou a consulta "esperada" a $G(.)$ ou $H(.)$). Temos então que

$$\Pr[\, A^* \mid \overline{D_{\text{err}}} \wedge \overline{S_A^*} \,] = \frac{1}{2}.$$

□

Segue então que

$$\begin{aligned}
\Pr[A^*] &= \Pr[\, A^* \mid D_{\text{err}} \,]\Pr[D_{\text{err}}] + \Pr[\, A^* \mid \overline{D_{\text{err}}}]\Pr[\overline{D_{\text{err}}}] \\
&= \Pr[\, A^* \mid D_{\text{err}} \,]\Pr[D_{\text{err}}] + \Pr[\, A^* \mid \overline{D_{\text{err}}} \wedge \overline{S_A^*} \,].\Pr[\overline{D_{\text{err}}} \wedge \overline{S_A^*}] + \\
&\quad \Pr[\, A^* \mid \overline{D_{\text{err}}} \wedge S_A^* \,]\Pr[\overline{D_{\text{err}}} \wedge S_A^*] \\
&\leqslant \Pr[D_{\text{err}}] + \frac{1}{2} + \Pr[S_A^*] \\
&\leqslant q_s 2^{-k} + \frac{1}{2} + \Pr[S_A^*].
\end{aligned}$$

Lembrando que $\Pr[A^*] = 1/2 + \lambda(k)$ para algum $\lambda(k)$ não-desprezível e que a simulação falha com probabilidade no máximo $q_s 2^{-k}$, temos que

$$\Pr[S_A^*] \geqslant \frac{1}{2} + \lambda(k) - \frac{1}{2} - 2.q_s 2^{-k} = \lambda(k) - q_s 2^{-k+1}.$$

que é não-desprezível (em $k$), provando assim o teorema. ■

### 3.4.2. Assinando no Modelo do Oráculo Aleatório

Suponha a existência de uma permutação com segredo. Provamos que, no Modelo do Oráculo Aleatório, o simples esquema de Hash-And-Sign usando qualquer permutação com segredo é seguro. Para facilitar nossa apresentação, utilizaremos a função RSA (definição 2) como exemplo. Provamos então que, no Modelo do Oráculo Aleatório, o esquema de Hash-And-Sign com RSA é seguro.

**Esquema de Assinatura 2.** *Hash-And-Sign com RSA* (`HaS-RSA`)

**Geração de Chaves.** Selecionam-se aleatoriamente dois primos de $n$ bits, $p$ e $q$; seja $N = pq$. Note que $\phi(N) = (p-1)(q-1)$. Escolha aleatoriamente um par $(e,d)$ tal que $ed \equiv 1 \pmod{\phi(N)}$. A chave pública será o par $(N,e)$ e a chave privada será o par $(N,d)$.

**Assinatura.** Seja $(d,N)$ a chave privada do usuário, seja $M$ a mensagem a ser assinada e seja $H(.)$ a função de *hash* apropriada. A assinatura $\sigma$ é calculada como

$$\sigma = H(M)^d \pmod{N}.$$

**Verificação.** Seja $(e,N)$ a chave pública do usuário, seja $(\sigma,M)$ o par assinatura-mensagem a ser verificado e seja $H(.)$ a função de *hash* apropriada. Aceite a assinatura $\sigma$ se e somente se

$$\sigma^e = H(M) \pmod{N}.$$

A estrutura da prova de segurança é bastante simples: vamos supor a existência de um algoritmo $A$ que é capaz de (com alta probabilidade) quebrar o `HaS-RSA`. Construiremos então um outro algoritmo, $S_A$, que internamente utiliza $A$ e é capaz de inverter a função RSA sem conhecer o segredo correspondente. Em outras palavras, $A$ representa um algoritmo probabilístico de tempo polinomial que:

1. Recebe como entrada a chave pública $(e,N)$ de um usuário arbitrário;
2. Tem acesso a um oráculo `H(.)` que calcula $H(x)$ para qualquer $x$;
3. Tem acesso a um oráculo `Assina` que calcula uma assinatura válida para qualquer $x$ (i.e., calcula $\sigma$ tal que $\sigma^e \equiv H(x) \pmod N$);
4. Devolve um par $(\sigma^*, M^*)$ tal que $(\sigma^*)^e \equiv H(M^*) \pmod N$.

O algoritmo $S_A$, por sua vez, funcionará da seguinte maneira:

1. Recebe como entrada uma instância do Problema-RSA, i.e., uma tupla $(Y,e,N)$ tal que $(\exists X)\,(Y \equiv X^e \pmod N)$.
2. A partir de $(Y,e,N)$ gera uma chave pública $(e',N')$;
3. Executa $A(e',N')$ e simula os dois oráculos a que $A$ tem acesso: `H(.)` e `Assina`;
4. Seja $(\sigma^*,M^*)$ o par devolvido por $A$. $S_A$ usa esta resposta para calcular $X$ tal que

$$X^e \equiv Y \pmod N.$$

**Teorema 12.** *Se a Função-RSA induzir uma permutação com segredo no seu domínio, o esquema de assinatura `HaS-RSA` é existencialmente seguro contra ataques de mensagem escolhida no modelo do oráculo aleatório.*

*Demonstração.* Seja $A$ um algoritmo polinomial capaz de gerar uma falsificação existencial do `HaS-RSA` com um ataque de mensagem escolhida no modelo do oráculo aleatório. Contruímos um algoritmo $S_A$ capaz de inverter a Função-RSA. $S_A$ recebe $(Y,e,N)$ como entrada. Seja $k = |N|$ o parâmetro de segurança e seja a chave pública $PK = (e,N)$. $S_A$ executa $A(e,N)$ como parâmetro. Seja $\lambda(k)$ a probabilidade de sucesso de $A$. Sabe-se que $\lambda(k)$ é não-desprezível, ou seja, para todo polinômio $p(.)$ e $k$ suficientemente grande,

$$\lambda(k) \geqslant \frac{1}{p(k)}.$$

Seja $q_H$ (resp. $q_S$) a quantidade de consultas feitas por $A$ ao oráculo `H(.)` (resp. `Assina`) durante a execução. Assumimos, sem perda de generalidade, que não são feitas consultas repetidas e que qualquer consulta a `Assina` é precedida de uma consulta a `H(.)` para a mesma mensagem. $S_A$ escolhe $t \xleftarrow{r} \{1,\ldots,q_H\}$. $S_A$ simula então os oráculos `H(.)` e `Assina` da seguinte maneira:

- **H($M_i$).** Se esta é a $t$-ésima consulta realizada por $A$, retorne $H(M_t) = Y$. Caso contrário, escolha $r_i \xleftarrow{r} \{0,1\}^k$, calcule $y_i = r_i^e \mod N$, e devolva $H(M_t) = y_i$.
- **Assina($M_j$).** Se $M_j = M_t$, $S_A$ pára e retorna `FALHA`. Caso contrário, $S_A$ encontra $i$ tal que $M_i = M_j$ e retorna $r_i$.[11]

[11] Perceba que como supomos que consultas a `Assina` são sempre precedidas de consultas a `H(.)` para a mesma mensagem, sempre poderemos encontrar $i, M_i = M_j$.

Se $A$ falhar, $S_A$ também falha. Com probabilidade $\lambda(k)$, $A$ devolve um par $(\sigma^*, M^*)$ válido. Se $M^* \neq M_t$, $S_A$ pára e devolve `FALHA`. Caso contrário, a saída de $S_A$ é $\sigma^* \equiv Y^e \pmod{N}$, invertendo a Função-RSA com sucesso.

Precisamos analisar agora a probabilidade $\varepsilon(k)$ de sucesso de $S_A$. Seja $A^*$ o evento em que $A$ tem sucesso ($\Pr[A^*] = \lambda(k)$). Qual a probabilidade de $M^*$ não ter sido consultada a `H(.)`? Como a resposta de `H(.)` é completamente aleatória, a chance $A$ adivinhá-la é basicamente $1/2^k$. Logo,

$$\begin{aligned}\Pr[A^*] &= \Pr[(A^*) \wedge (M^* \in \{M_1, \ldots, M_{q_H}\})] + \Pr[(A^*) \wedge (M^* \notin \{M_1, \ldots, M_{q_H}\})] \\ &= \sum_{i=1}^{q_H} \Pr[(A^*) \wedge (M^* = M_i)] + 2^{-k} = \lambda(k)\,.\end{aligned}$$

Como $t$ é escolhido aleatoriamente, tem-se que

$$\begin{aligned}\varepsilon(k) = \Pr[(A^*) \wedge (M^* = M_t)] &\geqslant \sum_{i=1}^{q_H} \Pr[(A^*) \wedge (M^* = M_i)]/q_H \\ &\geqslant (\lambda(k) - 2^{-k})/q_H, \qquad (13)\end{aligned}$$

e, como $q_H$ é polinomial (em $k$), se $\lambda(k)$ é não-desprezível, $\varepsilon(k)$ também o é. □

Perceba que esta redução em momento algum utiliza características específicas do RSA. Na verdade, uma redução análoga a esta pode ser usada para provar que *qualquer* permutação com segredo gera um esquema de assinatura baseado no `Hash-and-Sign` que é seguro no modelo do oráculo aleatório, o que nos leva ao teorema a seguir.

**Teorema 13.** *Qualquer permutação com segredo dá origem a um esquema de assinaturas existencialmente seguro contra ataques de mensagem escolhida no modelo do oráculo aleatório.*

A prova deste teorema é análoga à prova do teorema 12 e fica como exercício para o leitor.

#### 3.4.2.1. Eficiência da Redução

O leitor mais atento vai perceber que a redução apresentada na sub-seção anterior, apesar de manter o caráter polinomial do algoritmo, tem uma grande perda de eficiência. Relembrando a equação (13), $\varepsilon(k) \geqslant (\lambda(k) - 2^{-k})/q_H$, onde $q_H$ pode ser um número bastante grande, limitado apenas pelo número de passos de $A$. Quando tratamos de reduções em teoria da complexidade de algoritmos, geralmente a eficiência da redução não é um tema importante: o crucial é manter o caráter polinomial. Em criptografia, no entanto, a aplicação destas reduções de segurança requer um pouco mais de cuidado.

Para facilitar um pouco a discussão que segue introduziremos uma notação mais apropriada: dizemos que uma primitiva é $(t, \varepsilon)$-segura se todo algoritmo $A$ limitado a $t$ passos tem probabilidade no máximo $\varepsilon$ de "quebrar" a primitiva[12]. Nessa notação

[12] Perceba que as noções exatas de "quebrar" e "seguro" (e.g. que tipo de operações podem ser realizadas por $A$, oráculos aos quais tem acesso) dependem da primitiva em análise.

podemos então dizer que, se a Função-RSA é $(t,\varepsilon)$-segura, a equação (13) nos garante que o HaS−RSA é $(t,\lambda)$-seguro para

$$t \approx t' \quad \text{e} \quad \lambda \geqslant \varepsilon \cdot q_H .$$

Provamos que HaS−RSA é seguro se a Função-RSA for de difícil inversão fornecendo um algoritmo $S_A$ capaz de usar qualquer algoritmo $A$ que quebre HaS−RSA para inverter a Função-RSA. Supondo que inverter a Função-RSA é difícil, HaS−RSA é difícil. Mas, no mundo real, o que significa a afirmação "inverter a Função-RSA é difícil"? Significa que, para um conjunto de parâmetros factíveis, os melhores algoritmos conhecidos para inverter a Função-RSA têm chance desprezível de sucesso quando executados por um tempo factível. Usemos então a notação acima e digamos que um adversário que executa $2^{60}$ passos tem chance $2^{-100}$ de inverter a Função-RSA para um parâmetro de segurança $k$ específico, ou seja, esta é $(2^{60},2^{-100})$-segura. Digamos ainda que consideramos um esquema seguro se ele é $(2^{60},\varepsilon)$-seguro para $\varepsilon \leqslant 2^{-70}$. Se tentarmos então usar a redução acima para deduzir o nível de segurança do HaS−Sign quando instanciado com este mesmo parâmetro de segurança temos um problema. A primeira questão é: qual o valor de $q_H$, o limite superior no número de consultas a $H(.)$ executadas por $A$? O único limite indiscutível seria $2^{60}$, o número de passos do adversário, mas façamos o exercício de supor que uma em cada mil operações de $A$ seja uma consulta a $H(.)$. Isso nos dá, por exemplo, $q_H \approx 2^{50}$. A equação (13) nos garante então que esta instanciação do HaS−RSA é $(2^{60},2^{-50})$-segura, o que é insatisfatório pela nossa definição de segurança.

Os valores exatos nesta discussão não são relevantes; o importante é o questionamento do significado de uma redução ineficiente. Certamente a polinomialidade da redução é essencial. Mas apenas isso é suficiente para garantir a segurança do esquema resultante? Existe uma forte polêmica na comunidade criptográfica em relação a o quão relevante é a eficiência de uma redução e a como demonstrações como a apresentada anteriormente devem ser encaradas. Se tentarmos utilizar a redução para deduzir bons parâmetros para instanciar o esquema na prática, teremos que aumentar bastante o parâmetro de segurança (tornando as operações do esquema menos eficientes) para compensar a ineficiência da redução. Tendo isso em mente, surgiram novas propostas de esquemas que mantivessem as propriedades de segurança do HaS−RSA mas tivessem reduções de segurança eficientes. Analisamos a seguir uma variação de uma das mais importantes destas propostas, o PSS−RSA.

### 3.4.2.2. Assinaturas Aleatorizadas com o RSA

Seguindo o espírito da discussão anterior sobre a eficiência de reduções de segurança, em [Bellare e Rogaway, 1996] os autores propuseram um novo esquema de assinatura baseado no RSA, chamado de PSS−RSA, que tem uma redução de segurança eficiente. Analisaremos aqui uma versão simplificada deste esquema, seguindo [Coron, 2002], que ilustra muito bem as idéias presentes no PSS−RSA: manteremos o nome original do esquema, PFDH−RSA *(Probabilistic Full Domain Hash RSA)*. Esta é uma versão aleatorizada do HaS−RSA onde a mensagem é concatenada a um fator aleatório $r$ antes

de ser assinada. Este fator $r$ tem então que ser enviado junto à assinatura para que ela possa ser posteriormente verificada. Seja $k_0 < k$ um parâmetro do sistema. Nesta versão simplificada assumimos que $k_0 \geqslant \log_2 q_s$, onde $q_s$ é o número de assinaturas que permitiremos a um potencial adversário observar durante um ataque. Os resultados obtidos seriam muito próximos para $k_0 < \log_2 q_s$, mudando apenas a análise final quanto à probabilidade de sucesso, presente no artigo original [Coron, 2002].

**Esquema de Assinatura 3.** *Probabilistic Full Domain Hash RSA* (`PFDH-RSA`)

**Geração de Chaves.** Selecionam-se aleatoriamente dois primos de $n$ bits, $p$ e $q$; seja $N = pq$. Note que $\phi(N) = (p-1)(q-1)$. Escolha aleatoriamente um par $(e,d)$ tal que $ed \equiv 1 \pmod{\phi(N)}$. A chave pública será o par $(N,e)$ e a chave privada será o par $(N,d)$.

**Assinatura.** Seja $(N,d)$ a chave privada do usuário, seja $M$ a mensagem a ser assinada e seja $H(.)$ a função de *hash* apropriada. Escolha $r \xleftarrow{r} \{0,1\}^{k_0}$. A assinatura $\sigma$ é calculada como

$$\sigma = H(M||r)^d \pmod{N}.$$

**Verificação.** Seja $(N,e)$ a chave pública do usuário, seja $(\sigma,M,r)$ a tupla a ser verificada e seja $H(.)$ a função de *hash* apropriada. Aceite a assinatura $\sigma$ se e somente se:

$$\sigma^e = H(M||r) \pmod{N}.$$

Perceba que a geração de chaves aqui é exatamente como em `HaS-RSA`, e o que muda na geração/verificação de assinaturas é o uso do fator de aleatorização $r$.

**Teorema 14.** *Se a Função-RSA é $(t',\varepsilon')$-segura, o esquema de assinaturas `PFDH-RSA`[$k_0$] é $(t,q_h,q_s,\varepsilon)$-existencialmente-seguro no modelo do oráculo aleatório, onde*

$$k_0 \geqslant \log_2 q_s \tag{14}$$
$$t = t' - (q_h + q_s - q_h q_s) O(poli(k)) \tag{15}$$
$$\varepsilon = 4.\varepsilon' \tag{16}$$

*Demonstração.* Seja $A$ um algoritmo capaz de $(t,q_h,q_s,\varepsilon)$-quebrar o `PFDH-RSA`. Construímos então um algoritmo $S_A$ que utiliza $A$ para $(t',\varepsilon')$-quebrar a Função-RSA. $S_A$ recebe $(Y,e,N)$ como entrada e deve calcular $X$ tal que $X^e \equiv Y \pmod{N}$. Seja $k = |N|$ um parâmetro de segurança e seja a chave pública $PK = (N,e)$. $S_A$ manterá um contador $i$ ao longo da simulação, inicialmente valendo zero. $S_A$ executa $A$ com $PK$ como parâmetro. Assumimos, sem perda de generalidade, que $A$ não repete consultas ao oráculo $H(.)$.

Quando uma certa mensagem $M$ aparece pela primeira vez em uma consulta (seja de *hash* ou de assinatura), o contador $i$ é incrementado e $M_i = M$. $S_A$ gera então uma lista $L_i$ contendo $q_s$ números aleatórios em $\{0,1\}^{k_0}$. $S_A$ simula então os oráculos `H(.)` e `Assina` da seguinte maneira:

- **H($M_i, r_j$).**
  Escolhe um $x \xleftarrow{r} \{1, N-1\}$ e o associa a $r_j$.

Se $r_j \in L_i$ então devolva $H(M_i, r_j) = x^e \pmod N$;
caso contrário, devolva $H(M_i, r_j) = Yx^e \pmod N$.

- **Assina($M_i$).**
  $S_A$ escolhe o próximo $r' \in L_i$ aleatório. Se $H(M_i, r')$ não foi consultado ainda,
  - escolhe um $x \xleftarrow{r} \{1, N-1\}$ e o associa a $r'$;
  - faz $H(M_i, r_j) = x^e \pmod N$.

  Seja $x$ o valor associado a $r'$. Devolva a assinatura $(\sigma = x, r')$.

Finalmente, $A$ retorna a falsificação $\varsigma = (\sigma^*, M^*, r^*)$. A consulta $H(M^*, r^*)$ foi realizada com alta probabilidade[13]. Seja então $x$ o valor associado ao par $(M^*, r^*)$. Seja $L^*$ a lista relativa à mensagem $M^*$. Então, se $r^*$ não pertencer a $L^*$, temos que $H(M^*, r^*) \equiv Yx^e$ e a falsificação nos fornece

$$\sigma^* \equiv H(M^*, r^*)^d \equiv Y^d x^{ed} \equiv Y^d x \pmod N .$$

É fácil então obter $X = \sigma^* x^{-1}$, invertendo assim a instância da Função-RSA em questão.

Precisamos agora calcular a probabilidade de sucesso $\varepsilon(k)$ de $S_A$. Como por definição $A$ não pode ter consultado o oráculo `Assina(.)` na mensagem $M^*$, ele não pode ter obtido qualquer informação sobre $L^*$. Logo, a probabilidade de $r^* \notin L^*$ é $(1 - 2^{-k_0})^{q_s}$. Se $k_0 > \log_2 q_s$ e $q_s \geqslant 2$, obtém-se

$$\left(1 - 2^{-k_0}\right)^{q_s} \geqslant \left(1 - \frac{1}{q_s}\right)^{q_s} \geqslant \frac{1}{4} .$$

Como a probabilidade de sucesso de $A$ é $\varepsilon$, temos que $\varepsilon' \geqslant \varepsilon/4$. Além disso, o tempo de execução é dominado pela geração das listas $L_i$ ($q_h q_s$, $q_h$ listas de tamanho $q_s$), mais as rotinas de resposta para $(q_s + q_h)$ consultas de oráculo. □

Os resultados de [Coron, 2002] para o `PSS-RSA` e para o `PFDH-RSA` implicam que $k_0 \approx \log_2 q_s$, digamos 40 ou 50 bits. Apesar da eficiência da demonstração, permitindo o uso de um parâmetro de segurança $k$ menor, gostaríamos de não "desperdiçar" tanto espaço com a aleatorização da mensagem. É exatamente isso que nos traz ao próximo esquema de assinaturas que abordamos.

#### 3.4.2.3. Katz & Wang

Em [Katz e Wang, 2003] os autores apresentam um resultado surpreendente: eles provam que a aleatorização utilizada no `PSS-RSA` e no `PFDH-RSA` pode ser reduzida a apenas um bit. Utilizando apenas este bit adicional, eles conseguem uma redução eficiente entre a Função-RSA e o esquema de assinatura. Apresentamos primeiro a definição do esquema de assinaturas, seguida pela demonstração de sua segurança.

[13] Devido à aleatoriedade de $H(.)$, a chance de esta consulta não ser feita é $\leqslant 1/2^k$.

**Esquema de Assinatura 4.** *Katz-Wang RSA* (`KW-RSA`)

**Geração de Chaves.** Selecionam-se aleatoriamente dois primos de $n$ bits, $p$ e $q$; seja $N = pq$. Note que $\phi(N) = (p-1)(q-1)$. Escolha aleatoriamente um par $(e,d)$ tal que $ed \equiv 1 \pmod{\phi(N)}$. A chave pública será o par $(N,e)$ e a chave privada será o par $(N,d)$.

**Assinatura.** Seja $(N,d)$ a chave privada do usuário e seja $M$ a mensagem a ser assinada. Se $M$ já foi assinada antes, devolva a assinatura anteriormente calculada. Seja $H(.)$ a função de *hash* apropriada. Escolha um bit aleatório $b \xleftarrow{r} \{0,1\}$. A assinatura $\sigma$ é calculada como

$$\sigma = H(M,b)^d \pmod{N}.$$

**Verificação.** Seja $(N,e)$ a chave pública do usuário, seja $(\sigma,M)$ o par assinatura-mensagem a ser verificado e seja $H(.)$ a função de *hash* apropriada. Aceite a assinatura $\sigma$ se e somente se

$$\sigma^e \equiv H(M,0) \pmod{N} \quad \text{ou} \quad \sigma^e \equiv H(M,1) \pmod{N}.$$

Assumimos aqui que, apesar de haver duas assinaturas válidas para cada mensagem, usuários assinam apenas uma vez cada mensagem (i.e., se a assinatura da mesma mensagem for solicitada repetidas vezes, a mesma assinatura é devolvida). Este pode parecer um requisito complicado, requerendo que os usuários tenham "memória" de todas as mensagens assinadas, mas este não é o caso[14].

**Teorema 15.** *Se a Função-RSA é $(t',\varepsilon')$-segura, o esquema de assinaturas `KW-RSA` é $(t,q_h,q_s,\varepsilon)$-existencialmente-seguro no modelo do oráculo aleatório, onde*

$$t = t' - (q_h + q_s)O(\text{poli}(k)) \tag{17}$$
$$\varepsilon = 2.\varepsilon' \tag{18}$$

*Demonstração.* Seja $A$ um algoritmo capaz de $(t,q_h,q_s,\varepsilon)$-quebrar o `KW-RSA`. Construímos então um algoritmo $S_A$ que utiliza $A$ para $(t',\varepsilon')$-quebrar a Função-RSA. $S_A$ recebe $(Y,e,N)$ como entrada e deve calcular $X$ tal que $X^e \equiv Y \pmod{N}$. Seja $k = |N|$ o parâmetro de segurança e seja a chave pública $PK = (N,e)$. $S_A$ executa $A$ com $PK$ como parâmetro. Assumimos, sem perda de generalidade, que $A$ não repete consultas ao oráculo $H(.)$. $S_A$ simula os oráculos `H(.)` e `Assina` da seguinte maneira:

- **H($M_i,b$).**
  Escolhe um bit aleatório $c \in \{0,1\}$ e $t_1,t_2 \xleftarrow{r} \{1,N-1\}$.
  Se $b = c$
  - $H(M_i,b) = Yt_{i1}^e \pmod{N}$
  - $H(M_i,\hat{b}) = t_{i2}^e \pmod{N}$;

  caso contrário,
  - $H(M_i,b) = t_{i2}^e \pmod{N}$

[14] Perceba que a escolha do bit $b$ pode ser feita de forma determinística para uma dada mensagem, por exemplo, escolhendo $b = G(s||M)$, para uma função de hash $G$ e a chave privada $s$.

– $H(M_i, \hat{b}) = Yt_{i1}^e \mod N$.

- **Assina($M_i$).**
  Se $M_i$ ainda não foi consultada ao oráculo $H(.)$,

  – consulta $H(M_i, b)$, para $b \xleftarrow{r} \{0,1\}$.

  Devolva a assinatura $\sigma = t_{i2}$.

Finalmente, $A$ retorna a falsificação $\varsigma = (\sigma^*, M^*)$. A consulta $H(M^*, b)$, para algum $b \in \{0,1\}$, foi realizada com alta probabilidade. A assinatura $\sigma^*$ será então a $e$-ésima raiz de $t_{i2}^e$ ou de $Yt_{i1}^e$. Então, se $\sigma^* = t_{i2}^e$, $S_A$ "FALHA". Caso contrário, $X = \sigma^* t i 1^{-1}$ e $S_A$ consegue inverter a Função-RSA. Como as distribuições de $t_{i2}^e$ ou de $Yt_{i1}^e$ são indistinguíveis para $A$, a probabilidade de isso acontecer é $\frac{1}{2}$, ou seja $\varepsilon = 2\varepsilon'$. □

Este é um resultado realmente curioso e que pede uma discussão mais cuidadosa. Como destaca [Koblitz e Menezes, 2004], é, até certo ponto, um desafio à intuição que um bit possa fazer tanta diferença: esta redução é mais eficiente do que a apresentada para `HaS-RSA` por um fator de $q_H$, digamos $2^{40}$. Por outro lado, não podemos ignorar os fatos: a redução está aí e é correta. Há pelo menos uma década pesquisadores tentam obter resultados semelhantes para a versão simples do `HaS-RSA` e nada foi conseguido. Aparentemente `KW-RSA` é mais seguro que `HaS-RSA` — pelo menos no modelo do oráculo aleatório.

### 3.4.3. Controvérsias ao Redor do Modelo do Oráculo Aleatório

Um dos tópicos de maior controvérsia atualmente na comunidade criptográfica é a validade do modelo do oráculo aleatório: de um lado, ele obviamente não representa um modelo fiel da realidade mas, por outro lado, nenhum exemplo *prático* de problema de segurança foi encontrado nesses quinze anos em que ele vem sendo largamente utilizado. Tentaremos aqui sucintamente esclarecer um pouco o que está por trás desta discussão sem ter qualquer pretensão, no entanto, de poder chegar a uma resposta definitiva em um texto de caráter introdutório como este.

Certamente todo o estudo formal da criptografia está baseado em modelos: o próprio modelo padrão não é uma descrição fiel da realidade, nem poderia ser. Existe um consenso geral, no entanto, de que as suposições feitas na sua concepção são bastante razoáveis e que vulnerabilidades que não se encaixem na sua "visão de mundo" (e.g. *side-channel attacks*[15]) devem ser analisadas através de outras ferramentas. No cerne do modelo do oráculo aleatório está, no entanto, uma suposição completamente irreal: a existência de funções *realmente* aleatórias é impossível de ser realizada na prática. Espera-se, no entanto, que este requisito seja "satisfatoriamente" satisfeito por funções de *hash* adequadamente escolhidas: que um protocolo seguro no modelo do oráculo aleatório não contivesse falhas "estruturais", e qualquer ataque a ele seria necessariamente um ataque às funções de *hash* utilizadas na sua implementação real.

[15] Ataques deste tipo buscam obter informações de ou então injetar falhas nos dispositivos físicos onde o processamento criptográfico é executado ou onde informações criptográficas são armazenadas. O leitor interessado poderá encontrar mais informações em [Koeune e Standaert, 2005] e [Bar-El et al., 2006]

Em [Canetti et al., 1998] os autores fornecem um esquema de assinaturas que é demonstrado seguro no modelo do oráculo aleatório mas que é *não-instanciável* no seguinte sentido: sua instanciação através de qualquer família de funções é insegura. O esquema é altamente artificial e não é em nada parecido com qualquer uso "normal" do modelo do oráculo aleatório, mas o resultado é inegável: existe uma distância explorável entre o modelo do oráculo aleatório e o modelo padrão. A este resultado seguiu-se outro de [Bellare et al., 2004], razoavelmente mais realista mas que ainda viola certas "regras práticas" de protocolos criptográficos[16], o que leva muitos pesquisadores a desacreditarem ambos os resultados como meramente teóricos.

A situação do modelo do oráculo aleatório é, portanto, incerta. Construções que não baseiam sua segurança neste modelo são ainda bastante ineficientes, ou dependem de suposições ainda menos estudadas. Por outro lado, os resultados de [Canetti et al., 1998] e [Bellare et al., 2004], apesar de encarados por muitos pesquisadores como distantes da realidade, são, no mínimo, preocupantes. O futuro próximo da pesquisa em segurança demonstrável deve encarar o problema de encontrar vulnerabilidades mais palpáveis no modelo do oráculo aleatório, ou melhor limitar o cenário onde elas podem existir às já conhecidas (e irreais).

## 3.5. Considerações finais

Procuramos, através deste texto cobrir os tópicos introdutórios mais importantes da segurança demonstrável, seguimos principalmente uma abordagem histórica dos resultados mais importantes obtidos na área, começando com os artigos seminais do início dos anos 80, que estabeleceram as bases formais sob as quais a criptografia viria a ser estudada nos anos vindouros, passando por avanços relevantes que surgiram nos anos seguintes e chegando ao paradigma do oráculo aleatório. Povoamos o texto com demonstrações de segurança dos mais variados esquemas, contra os mais variados tipos de ataques. Esperamos assim ajudar o leitor a se sentir mais confortáveis com a linguagem e as principais técnicas utilizadas nestas demonstrações.

Certamente não foi possível cobrir a totalidade dos assuntos relevantes à pesquisa nesta área e tópicos de extrema importância tiveram que ser deixados de lado: não tratamos de criptossistemas baseados em emparelhamentos bilineares ou esquemas baseados em identidade (e variações), não abordamos o modelo de grupo genérico ou as técnicas de replay de oráculo (e o lema da bifurcação), lidamos exclusivamente com esquemas de ciframento e de assinaturas, ignorando assim classes de protocolos mais complexos. Todos estes tópicos são de extrema importância e cabem em textos mais avançados sobre o assunto.

Sentimos, no entanto, que o conteúdo coberto no texto capacita pesquisadores a entender boa parte das provas de segurança que são produzidas atualmente, e isto é de extrema importância para o desenvolvimento da criptografia enquanto ciência. É extremamente negativo, por exemplo, que resultados de extrema expressividade, como

[16]Especificamente, a construção de [Bellare et al., 2004] é de um criptossistema híbrido, usando um criptossistema de chave pública para cifrar a chave de um esquema de chave privada que é usado para efetivamente cifrar os dados. Neste exemplo eles utilizam informação do esquema de chave pública no esquema de chave privada, o que é evitado em qualquer implementação realista deste tipo de criptossistema.

o `RSA-OAEP` [Bellare e Rogaway, 1994], possam ter falhas em suas demonstrações de segurança descobertas sete anos após a sua publicação e utilização em diversos padrões internacionais [Shoup, 2001]. Sentimos que isto acontece, em parte, pela não-familiaridade de boa parte da comunidade de pesquisa com as técnicas utilizadas na construção destas demonstrações. Esperamos, no entanto, que quanto mais pesquisadores sintam-se confortáveis com estas técnicas, e sintam-se aptos a entender as demonstrações de resultados que lhe parecem interessantes, maior será a probabilidade de erros nas mesmas serem mais rapidamente encontrados.

## Referências Bibliográficas

[Bar-El et al., 2006] Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., e Whelan, C. (2006). The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE*, 94(2).

[Bellare et al., 2004] Bellare, M., Boldyreva, A., e Palacio, A. (2004). An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. Em [Cachin e Camenisch, 2004], p. 171–188.

[Bellare e Rogaway, 1993] Bellare, M. e Rogaway, P. (1993). Random oracles are practical: a paradigm for designing efficient protocols. Em *CCS '93: Proceedings of the 1st ACM conference on Computer and communications security*, p. 62–73, New York, NY, USA. ACM Press.

[Bellare e Rogaway, 1994] Bellare, M. e Rogaway, P. (1994). Optimal asymmetric encryption. Em *EUROCRYPT*, p. 92–111.

[Bellare e Rogaway, 1996] Bellare, M. e Rogaway, P. (1996). The exact security of digital signatures - how to sign with rsa and rabin. Em *EUROCRYPT*, p. 399–416.

[Blum e Goldwasser, 1985] Blum, M. e Goldwasser, S. (1985). An *efficient* probabilistic public key encryption scheme which hides all partial information. Em *Proceedings of CRYPTO 84 on Advances in cryptology*, p. 289–302, New York, NY, USA. Springer-Verlag New York, Inc.

[Boneh e Franklin, 2003] Boneh, D. e Franklin, M. (2003). Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615.

[Cachin e Camenisch, 2004] Cachin, C. e Camenisch, J., editores (2004). *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 de *Lecture Notes in Computer Science*. Springer.

[Canetti et al., 1998] Canetti, R., Goldreich, O., e Halevi, S. (1998). The random oracle methodology, revisited (preliminary version). Em *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, p. 209–218, New York, NY, USA. ACM Press.

[Chor e Goldreich, 1985] Chor, B. e Goldreich, O. (1985). Rsa/rabin least significant bits are 1-2- + 1/poly(log n) secure. Em *Proceedings of CRYPTO 84 on Advances in cryptology*, p. 303–313, New York, NY, USA. Springer-Verlag New York, Inc.

[Coron, 2002] Coron, J.-S. (2002). Optimal security proofs for pss and other signature schemes. Em *EUROCRYPT '02: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, p. 272–287, London, UK. Springer-Verlag.

[Cramer e Shoup, 1998] Cramer, R. e Shoup, V. (1998). A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. Em *CRYPTO '98: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, p. 13–25, London, UK. Springer-Verlag.

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

[Goldreich, 2003] Goldreich, O. (2003). *Foundations of Cryptography: Volume I Basic Tools*. Cambridge University Press, Cambridge.

[Goldreich, 2004] Goldreich, O. (2004). *Foundations of Cryptography: Volume II Basic Applications*. Cambridge University Press, Cambridge.

[Goldwasser e Micali, 1982] Goldwasser, S. e Micali, S. (1982). Probabilistic encryption & how to play mental poker keeping secret all partial information. Em *STOC '82: Proceedings of the fourteenth annual ACM symposium on Theory of computing*, p. 365–377, New York, NY, USA. ACM Press.

[Goldwasser e Micali, 1984] Goldwasser, S. e Micali, S. (1984). Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299.

[Katz, 2004] Katz, J. (2004). Lecture notes — advanced topics in cryptography. `http://www.cs.umd.edu/~jkatz/gradcrypto2/scribes.html`.

[Katz e Wang, 2003] Katz, J. e Wang, N. (2003). Efficiency improvements for signature schemes with tight security reductions. Em *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, p. 155–164, New York, NY, USA. ACM Press.

[Koblitz e Menezes, 2004] Koblitz, N. e Menezes, A. (2004). Another look at "provable security". Cryptology ePrint Archive, Report 2004/152.

[Koeune e Standaert, 2005] Koeune, F. e Standaert, F.-X. (2005). *Foundations of Security Analysis and Design III: FOSAD 2004/2005 Tutorial Lectures*, volume 3655 de *Lecture Notes in Computer Science*, capítulo A Tutorial on Physical Security and Side-Channel Attacks, p. 78–108. Springer, Berlin Heidelberg.

[Lipton, 1981] Lipton, R. J. (1981). How to cheat at mental poker. Proceedings of the AMS Short Course on Cryptology.

[Micali et al., 1988] Micali, S., Rackoff, C., e Sloan, B. (1988). The notion of security for probabilistic cryptosystems. *SIAM J. Comput.*, 17(2):412–426.

[Rabin, 1979] Rabin, M. O. (1979). Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT-LCS-TM-212, Massachusetts Institute of Technology.

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

[Shamir et al., 1979] Shamir, A., Rivest, R. L., e Adleman, L. M. (1979). Mental poker. Technical Report MIT-LCS-TM-125, Massachusetts Institute of Technology.

[Shoup, 2001] Shoup, V. (2001). Oaep reconsidered. Em *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, p. 239–259, London, UK. Springer-Verlag.

[Stinson, 2006] Stinson, D. R. (2006). *Cryptography: Theory and Practice*. Chapman & Hall/CRC, Boca Raton, London, New York, 3 edição.

[Yao, 1982] Yao, A. C. (1982). Theory and applications of trapdoor functions. Em *Proc. 23rd IEEE Symposium on Foundations of Computer Science (FOCS)*, p. 80–91.