Instituto de Computação
Universidade Estadual de Campinas

Cifrassinatura sem Certificados

Erick Nogueira do Nascimento

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Erick Nogueira do Nascimento e aprovada pela Banca Examinadora.

Campinas, 21 de dezembro de 2011.

Ricardo Dahab
Instituto de Computação
Universidade Estadual de Campinas
(Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

ERRATA

Na folha **i e ii** - **Onde se lê:** Erick Nogueira do Nascimento **Leia-se:** Érick Nogueira do Nascimento

Prof. Dr. Paulo Lício de Geus
Coordenador de Pós Graduação
Instituto de Computação UNICAMP
Matr.103268

FICHA CATALOGRÁFICA ELABORADA POR
MARIA FABIANA BEZERRA MÜLLER - CRB8/6162
BIBLIOTECA DO INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E
COMPUTAÇÃO CIENTÍFICA - UNICAMP

N17c

Nascimento, Erick Nogueira do, 1986-
Cifrassinatura sem certificados / Erick Nogueira do Nascimento. - Campinas, SP : [s.n.], 2011.

Orientador: Ricardo Dahab.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

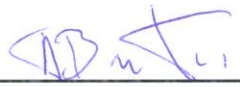
1. Criptografia. 2. Tecnologia da informação - Segurança . 3. Criptografia - Certificados e licenças. 4. Cifrassinatura. I. Dahab, Ricardo, 1957- II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

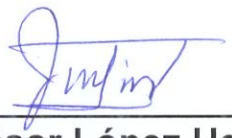
Título em inglês: Certificateless signcryption
Palavras-chave em inglês:
Crypotography
Information tecnology – Security
Computer network protocols
Signcryption
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Ricardo Dahab [Orientador]
Paulo Sérgio Licciardi Messeder Barreto
Julio César López Hernández
Data da defesa: 16-09-2011
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

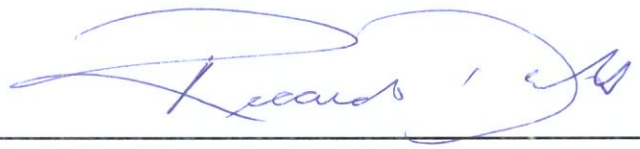
Dissertação Defendida e Aprovada em 16 de setembro de 2011, pela Banca examinadora composta pelos Professores Doutores:



Prof. Dr. Paulo Sérgio Licciardi Messeder Barreto
Escola Politécnica / USP



Prof. Dr. Julio César López Hernández
IC / UNICAMP



Prof. Dr. Ricardo Dahab
IC / UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

Cifrassinatura sem Certificados

Erick Nogueira do Nascimento[1]

Setembro de 2011

Banca Examinadora:

- Ricardo Dahab
  Instituto de Computação
  Universidade Estadual de Campinas (Orientador)
- Paulo Sérgio Licciardi Messeder Barreto
  Escola Politécnica
  Universidade de São Paulo
- Julio Cesar López Hernández
  Instituto de Computação
  Universidade Estadual de Campinas
- Marco Aurélio Amaral Henriques (Suplente)
  Instituto de Computação
  Universidade Estadual de Campinas
- Leonardo Barbosa Oliveira (Suplente)
  Instituto de Computação
  Universidade Estadual de Campinas

[1]Suporte financeiro de: Bolsa do CNPq (processo no. 133481/2009-3).

ERRATA

Na folha **iv** - **Onde se lê**: Erick Nogueira do Nascimento **Leia-se**: Érick Nogueira do Nascimento

Prof. Dr. Paulo Lício de Geus
Coordenador de Pós Graduação
Instituto de Computação UNICAMP
Matr.103268

Resumo

A criptografia de chave pública está cada vez mais presente nos sistemas computacionais, provendo a estes diversas propriedades de segurança, dentre as quais: confidencialidade, integridade, autenticidade e irretratabilidade. O modelo de criptografia de chave pública explicitamente certificado é o mais comumente empregado, e compreende uma infraestrutura de chave pública (PKI) composta por procedimentos, hardware, software e pessoal administrativo para a sua operação. Tal infraestrutura é complexa e onerosa, o que torna o seu uso proibitivo em diversas situações. Neste trabalho foram abordados paradigmas de criptografia de chave pública alternativos ao paradigma PKI, com foco no paradigma sem certificados. Dentro deste paradigma, e com ênfase em segurança demonstrável, foram estudados os esquemas de cifrassinatura, os quais provêem eficientemente e simultaneamente as propriedades da encriptação de chave pública com as propriedades da assinatura digital: confidencialidade, integridade, autenticidade e irretratabilidade.

Este trabalho tem como contribuições: (*i*) ataque contra a propriedade de indistinguibilidade do IBSC McCullagh-Barreto [MB04], (*ii*) proposta de correção do esquema CLSC Barbosa-Farshim [BF08], o qual havia sido quebrado por Selvi et al [SVR10b], (*iii*) exposição sistemática sobre segurança demonstrável, criptografia de chave pública sem certificados e cifrassinatura sem certificados.

Abstract

Public-key cryptography is ever more present on computational systems, providing them several security properties, including: confidentiality, integrity, authenticity and non-repudiation. The explicitly certified public-key cryptography model is the most commonly employed one, and it consists of a public-key infrastructure (PKI) which requires procedures, hardware, software and management personnel for its operations. Such infrastructure is complex and costly, making its use prohibitive in many cenarios. This work approached alternative paradigms for public-key cryptography, with focus on the certificateless paradigm. On this paradigm, and with emphasis on provable security, we studied signcryption schemes, which provide efficiently and simultaneously the properties of public-key encryption with those of digital signature: confidentiality, integrity, authenticity and non-repudiation.

This work has the following contributions: (*i*) attack against the indistinguishability property of IBSC McCullagh-Barreto [MB04] (*ii*) correction for the CLSC Barbosa-Farshim [BF08], which had been broken by Selvi et al [SVR10b], (*iii*) systematic exposition about provable security, certificateless public-key cryptography and certificateless signcryption.

Agradecimentos

A Deus, pois sem ele nada seria possível.

Aos meus pais e avós, pelo incentivo e suporte.

À minha esposa, Shirlei, pelo carinho e compreensão.

Ao meu orientador, Ricardo Dahab, pelo apoio, aconselhamento e oportunidades.

Ao CNPq, pelo apoio finaceiro.

Sumário

Lista de Figuras

Capítulo 1

Introdução

1.1 Motivação

O conceito de criptografia de chave pública surgiu no trabalho de Diffie-Hellman [DH76]. Um problema fundamental na criptografia de chave pública, que não havia sido resolvido pelos seus criadores é o problema da autenticidade das chaves públicas. Em um esquema criptográfico de chave pública, se não é garantida a autenticidade das chaves públicas dos usuários, então este esquema não é seguro. Um atacante sempre consegue forjar uma chave pública (e a correspondente chave privada) falsa para um usuário A e publicá-la no lugar da chave pública legítima deste usuário. Com isto, se o esquema em questão é de encriptação, todos os usuários que forem encriptar mensagens para A utilizarão a chave pública falsa, ao invés da legítima, permitindo que o atacante consiga decifrar estas mensagens (pois tem a chave privada correspondente a chave pública falsa). Analogamente, se o esquema é de assinatura, o atacante consegue produzir assinaturas falsas sobre mensagens arbitrárias (utilizando a chave privada correspondente a chave pública falsa), como se estas assinaturas tivessem sido geradas por A. Estas assinaturas falsas terão validação positiva quando passarem pelo algoritmo de verificação, e portanto o adversário é capaz de personificar o usuário A para qualquer outro usuário do sistema.

Uma solução proposta para o problema da autenticidade das chaves públicas foi a utilização de *certificados de chaves públicas*, constituindo o conceito de criptografia de chave pública *explicitamente certificada*, onde o termo *explicitamente* aparece devido ao uso de um artefato explícito e especificamente utilizado para garantir a autenticidade da chave pública, o *certificado*. O conceito de criptografia de chave pública *explicitamente certificada* utiliza a idéia de certificado de chave pública emitido por uma entidade confiável, chamada *autoridade certificadora* (CA), a qual *certifica* que uma dada entidade realmente possui a chave pública alegada. Por exemplo, se Bob deseja obter um certificado sobre sua chave pública P_B, a CA irá entrar em contato com Bob (provavelmente pessoalmente),

verificará a identidade de Bob (ID_B), verificará se Bob *sabe* a chave privada S_B correspondente a chave pública P_B (possivelmente por meio de um protocolo de conhecimento nulo). Se as verificações anteriores forem bem sucedidas, a CA tem a garantia de que P_B é de fato a chave pública de Bob, e então ela assina P_B utilizando a sua chave privada S_{CA}. Esta assinatura, juntamente com mais informações (dentre elas: ID_B, ID_{CA}, carimbo de tempo, prazo de validade, algoritmo de assinatura), constitui o certificado da chave pública de Bob (C_{CA,P_B}) emitido pela CA. Se a entidade Alice deseja verificar a autenticidade da chave pública de Bob (P_B), ela obtém C_{CA,P_B}, obtém P_{CA} de forma autenticada, e então verifica a assinatura de CA sobre P_B.

O uso de certificados necessita de um conjunto de procedimentos (geração, distribuição, validação e revogação) e os respectivos recursos de hardware, software e pessoal administrativo para sua operação. Este arcabouço de recursos e procedimentos necessários para o gerenciamento do ciclo de vida dos certificados constitui a PKI (Public Key Infrastructure) [AL99] [HFPS02]. A manutenção desta infraestrutura traz um elevado custo operacional para as empresas que necessitam da criptografia de chave pública, tornando muitas vezes proibitivo o seu uso em operações em pequena escala e também em dispositivos com baixo poder de processamento e/ou de armazenamento, como os sistemas embarcados (redes de sensores, telefones móveis, entre outros) [Gut02]. Devido a estes problemas, paradigmas alternativos a PKI foram propostos.

O conceito de certificação *implícita* surgiu como uma alternativa ao conceito de certificação explícita. Tal como no modelo explicitamente certificado, há garantia da autenticidade da chave pública, porém, esta garantia não é obtida através da validação da chave pública com o uso de um artefato explícito (o certificado), mas sim naturalmente (implicitamente), no momento do uso das chaves públicas. Ou seja, as operações criptográficas serão executadas corretamente (sem falhas de validação na decriptação ou na verificação de assinatura) se, e somente se, foram utilizadas as chaves públicas e privadas corretas (aquelas legitimamente associadas as entidades dos usuários do sistema). A grosso modo, isto significa que, se o usuário utilizar uma chave pública incorreta (p.ex. fornecida por um atacante), este uso não revelará quaisquer dados confidenciais ao atacante (se a operação é encriptação), e nenhum par (mensagem, assinatura) terá validação positiva se esta validação for realizada com a chave pública legítima (se a operação é assinatura), isto é, o adversário não consegue produzir falsificações.

Dentre os paradigmas de criptografia de chave pública baseados no conceito de certificação implícita, temos: criptografia baseada em identidades (IB-PKC), criptografia autocertificada (SC-PKC), criptografia sem certificados (CL-PKC), criptografia baseada em certificados (CB-PKC). O modelo CL-PKC foi o modelo escolhido para ser tratado neste trabalho, pois ele provê simultaneamente o que há de melhor, deixando de lado os problemas, dos paradigmas PKI e IB-PKC. Tal como PKI, mas diferentemente de

IB-PKC, a autoridade confiável (TA) de CL-PKC não tem custódia da chave privada dos usuários. Diferentemente de PKI, e analogamente a IB-PKC, não são necessários certificados de chave pública.

Em sistemas criptográficos, quando a confidencialidade (ou a autenticidade com irretratabilidade) é um requisito necessário, normalmente o outro requisito também é necessário, ou pelo menos é desejável. Logo, para se obter ambos os requisitos, podem ser utilizadas ambas as operações: encriptação, para garantir confidencialidade; e assinatura digital, para garantir autenticidade com irretratabilidade. Um esquema critográfico que garante confidencialidade, autenticidade e irretratabilidade é chamado de *esquema de cifrassinatura*.

Entretanto, a combinação em série ingênua de encriptação e assinatura pode ser problemática, pois, no contexto de sistemas criptográficos de chave pública, o custo computacional ou de comunicação (largura de banda ou atraso) de um esquema de encriptação (ou assinatura) é elevado, normalmente ordens de magnitude mais alto do que o custo de um esquema simétrico correspondente. Este fato torna custosa a utilização de um esquema de cifrassinatura *genérico* formado pela composição em série de um esquema de encriptação seguido de um esquema de assinatura (ou vice-versa). Para tanto, foram criados esquemas de cifrassinatura que não podem ser decompostos em um esquema de encriptação e um esquema de assinatura, e que obteem ganhos substanciais de eficiência em relação aos esquemas obtidos por composição genérica.

1.2 Objetivos

Esta dissertação tem como objetivo central o estudo de esquemas de cifrassinatura sem certificados (CLSC), com foco na segurança demonstrável de tais esquemas.

1.3 Contribuições

- Exposição sistemática sobre Segurança Demonstrável e criptografia de chave pública sem certificados (CL-PKC), incluindo extensa revisão literária sobre esquemas de cifrassinatura sem certificados (CLSC).

- Ataque sobre a confidencialidade do IBSC McCullagh-Barreto [MB04], apresentado no Capítulo 5.

- Correção do esquema Barbosa-Farshim [BF08], o qual havia sido quebrado por Selvi et al [SVR10b].

1.4 Organização

Este documento está organizado da seguinte forma:

O Capítulo 2 é dedicado à apresentação dos paradigmas alternativos para criptografia de chave pública.

A seguir, o Capítulo 3 trata da definição de conceitos de segurança demonstrável, o modelo do oráculo aleatório, discussões sobre a importância da segurança demonstrável e controvérsias.

Adiante, o Capítulo 4 apresenta o conceito de encriptação, assinatura, cifrassinatura e acordo de chaves no paradigma sem certificados, incluindo os modelos formais de segurança para cada tipo de esquema, além de considerações específicas de segurança no paradigma sem certificados, como, por exemplo, os ataques de KGC maliciosa.

Por fim, o Capítulo 5 apresenta os modelos de segurança para cifrassinatura de chave pública em geral, e no paradigma sem certificados em particular, e revisa os esquemas de cifrassinatura sem certificados na literatura, com foco na segurança. Ao final deste capítulo apresentamos um ataque sobre a indistinguibilidade do IBSC McCullagh-Barreto.

Capítulo 2

Modelos Alternativos de Criptografia de Chave Pública

Neste capítulo discutimos modelos de criptografia de chave pública alternativos ao modelo explicitamente certificado (PKC ou PKI). Serão abordados os modelos baseado em identidades (Seção 2.1), autocertificado (Seção 2.2) (incluindo certificado implícito, Subseção 2.2.1), sem certificados (Seção 2.3) e baseado em certificados (Seção 2.4).

2.1 Criptografia de Chave Pública Baseada em Identidades (IB-PKC)

A Criptografia Baseada em Identidades (IB-PKC ou IBC) surgiu em 1984 no trabalho de Shamir [Sha85]. Nesse trabalho foi proposto o uso das identidades dos usuários como as suas chaves públicas, tornando assim desnecessária toda e qualquer infra-estrutura para a distribuição confiável (autenticada) de chaves públicas, já que dada uma identidade, o usuário correspondente é univocamente determinado, desde que a identidade seja não-falsificável. Portanto, não há necessidade de um objeto de certificação (certificado) para autenticar a chave pública de um usuário.

No trabalho original de Shamir foram apresentados o modelo (esquema genérico) de encriptação baseada em identidades (IBE) e o modelo de assinatura baseada em identidade (IBS), além de um esquema IBS concreto. O autor deixou em aberto o problema de se obter um esquema IBE concreto, o qual foi fechado (satisfatoriamente) somente em 2001, com o IBE Boneh-Franklin [BF01], através do uso de emparelhamentos bilineares. De fato, esquemas IBE e IBS realmente práticos foram descobertos somente recentemente, principalmente devido ao uso de emparelhamentos bilineares.

Em esquemas sob o modelo IBC, a certificação ocorre implicitamente:

- em um esquema de assinatura baseado em identidade (IBS), se o assinante (Alice) gera uma assinatura de uma mensagem, e esta assinatura não pôde ser verificada pelo receptor (Beto), então ou Beto utilizou uma chave pública falsa (não pertencente a Alice) ou não utilizou a identidade correta de Alice (ou ambos), durante a verificação. Caso contrário, se Beto utilizou a identidade correta de Alice e a chave pública legítima dela, então Beto consegue verificar a assinatura de Alice.

- em um esquema de encriptação baseado em identidade (IBE), se o remetente encripta uma mensagem utilizando a chave pública (identidade) do destinatário, e o destinatário consegue decriptar esta mensagem usando a sua chave privada, então o remetente utilizou a chave pública (identidade) correta do destinatário.

Podemos então dizer, a grosso modo, que o *uso correto* (sem erros) de uma chave pública (identidade) garante que esta é a chave pública correta. Pelo contrário, o uso de uma chave pública falsa impede a correta inversão da operação criptográfica. Isto caracteriza o que chamamos de *certificação implícita*. Neste paradigma, a identidade de um usuário pode ser qualquer conjunto de informações que identifique unicamente este usuário.

A Figura 2.1 mostra o esquema genérico para IBS, contendo a autoridade confiável (TA), chamada de Centro de Geração de Chaves (KGC) no paradigma IBC. Como pode ser visto no IBS de Shamir [Sha85] (Figura 2.2), a KGC é responsável por gerar a chave privada dos usuários do sistema. Este é um grande problema, pois a TA é capaz de personificar qualquer usuário do sistema, e portanto o nível de confiança que deve ser depositado nela é muito grande. Devido a este problema de custódia de chaves por parte da TA, o uso de IBC não é recomendável em ambientes abertos como a Internet, ficando restrito a ambientes muito bem controlados.

O comprometimento da chave mestra de uma KGC em IB-PKC é, em geral, mais desastroso do que o comprometimento da chave de assinatura de uma CA em PKI. Em esquemas IBE, após o comprometimento da chave mestra da PKG todas as mensagens cifradas geradas *antes e depois* do incidente podem ser decifradas pelo adversário, ou seja, esquemas IBE não possuem *forward-secrecy* (sigilo a posteriori). Já em esquemas PKE, após o comprometimento da chave mestra da CA o adversário é capaz de gerar novos pares de chaves para usuários já existentes (ou não) no sistema e computar certificados para estas chaves, mas não é capaz de computar a chave privada correspondente a chave pública atual dos usuários já existentes no sistema. Portanto, o sigilo das mensagens cifradas *antes* do incidente é mantido após o incidente, ou seja os esquemas PKE apresentam forward-secrecy.

O segundo problema de esquemas IBC é a distribuição da chave privada pela TA ao usuário, a qual precisa ser feita via canal sigiloso e autêntico.

1. **Inicializa** (KGC)
   `Entrada`: 1^k, para um parâmetro de segurança k.
   `Saída`: chave privada mestra s e parâmetros públicos *params*.

2. **Extrai-Chave-Secreta** (KGC)
   `Entrada`: *params*, s, identificador ID_A.
   `Saída`: chave secreta x_A. Obs: x_A deve ser enviada a A através de um canal seguro (com sigilo e autenticidade).

3. **Assina** (A)
   `Entrada`: *params*, ID_A, x_A, mensagem m.
   `Saída`: assinatura σ.

4. **Verifica** (B)
   `Entrada`: *params*, ID_A, m, σ.
   `Saída`: m ou $\perp$.

Figura 2.1: Esquema IBS Genérico

O terceiro problema é invalidar o par de chaves de um usuário após um período de tempo (problema da *expiração*), ou seja, tornar impossível o uso deste par de chaves após o esgotamento deste período de tempo. Isto é necessário pois a chave pode ter sido comprometida sem o usuário saber, ou os algoritmos ou protocolos deixaram de ser seguros (p.ex. por avanços em criptanálise ou acréscimo de poder computacional). A solução adotada por PKI para este problema foi a inclusão da data de validade no certificado emitido pela CA.

Em IBC, porém, a chave pública de um usuário é a identidade dele, um valor fixo. Para resolver o problema de expirar o par de chaves é necessário adicionar algum tipo de informação que varie com o tempo. Uma das soluções é adicionar um campo de tempo à chave pública do usuário, resultando em uma chave pública da forma $ID' = (ID, Data)$ [1]. Deste modo, a TA pode gerar uma chave privada para os usuários para cada valor de *Data* (p.ex. diariamente). Se a chave privada for gerada diariamente, os usuários que desejam encriptar mensagens para um usuário A utilizam $ID' = (ID_A, DataCor)$ como chave pública de A onde *DataCor* é a data corrente. Para o usuário A decriptar as mensagens enviadas nesta data, ele deve possuir a chave privada desta data.

Na solução do parágrafo anterior, se a *TA* enviar a chave privada do usuário somente quando *chegar* a data, temos a propriedade de *encriptação para o futuro*, ou seja, um usuário A é capaz de cifrar uma mensagem m para um usuário B para a data d de modo

[1] *Data* pode ter, por exemplo, o seguinte formato: $DD/MM/YYYY$.

que B só consegue decifrar m na data d ou posteriormente.

O quarto problema de IBC é a revogação do par de chaves do usuário após o comprometimento da sua chave privada (problema da *revogação*), ou seja, inutilizar o par de chaves do usuário após o recebimento da notificação deste sobre o comprometimento de sua chave privada. Obviamente, a revogação, se for aplicada, ocorrerá antes da expiração do par de chaves do usuário. Uma solução para o problema da revogação é utilizar a solução do terceiro problema, variando *Data* em incrementos de tempo pequenos, tornando curta a janela de tempo de vulnerabilidade para o usuário que teve a sua chave privada comprometida.

2.2 Criptografia de Chave Pública Autocertificada (SC-PKC)

Para tentar resolver o problema da custódia de chaves por parte da TA, foi proposto por Girault [Gir91] um paradigma que se posiciona entre o paradigma de chave pública explicitamente certificado (PKI) e o paradigma IBC.

Para tanto, ele propõe a idéia de *chaves públicas autocertificadas* (SC-PKC), onde a chave pública é calculada em conjunto pela TA e pelo usuário, de maneira que o certificado esteja incluído na chave, e portanto não precise ser enviado separadamente. Diferentemente de IBC, em esquemas baseados em chaves públicas autocertificadas a chave secreta é de conhecimento exclusivo do usuário.

Apresentamos na Figura 2.3 o esquema de Girault para geração de chaves baseado no RSA. Neste esquema a TA pode agir de forma maliciosa gerando duas ou mais chaves públicas para um mesmo usuário. Porém, a existência de duas chaves públicas válidas caracteriza comportamento malicioso da TA, o qual é detectável. No artigo, Girault demonstra que o esquema atinge Nível-Girault 3 de segurança na escala definida pelo autor, significando que *qualquer* fraude por parte da TA é detectável. Este é o mesmo nível de segurança atingido pelo paradigma de certificação explícita.

Quando o usuário Bob deseja cifrar uma mensagem m para Alice, ele:

1. Obtém a chave pública P_A de Alice através de um canal possivelmente inseguro (não-autêntico).

2. Executa `Protocolo-Identificação` com Alice (para verificar se Alice é capaz de computar o logaritmo discreto de $P_U^e + \mathrm{ID}$ na base g, o que indica que ela sabe o valor s).

3. Encripta m com a chave pública P_A (utilizando algum esquema de encriptação de chave pública).

1. **Inicializa** (Esta função é executada pela KGC)
   (a) Gera um par de chaves RSA $((n, e), d)$, onde n, e e d são definidos como no RSA.
   (b) Escolhe funções de hash criptográficas H_1 e H_2.
   (c) A chave pública mestra (master public key) é $mpk := ((n, e), H_1, H_2)$, e a chave privada mestra (master private key) é $msk := d$.
2. **Extrai Chave Secreta** (Esta função é executada pela KGC)
   (a) Seja ID a identidade do usuário; a sua chave secreta é:
   $$x_{\mathrm{ID}} := H_1(\mathrm{ID})^d \ mod \ n$$
3. **Assina**
   (a) Seja m a mensagem a ser assinada.
   (b) O usuário ID com chave secreta x_{ID}, escolhe aleatoriamente $k \in \mathbb{Z}_n^*$ e calcula:
      i. $r := k^e \ mod \ n$.
      ii. $\sigma := x_{\mathrm{ID}} k^{H_2(r,m)} \ mod \ n$.
   (c) A assinatura é (σ, r).
4. **Verifica**
   (a) Seja (σ, r) a assinatura do usuário ID na mensagem m.
   (b) Aceita a assinatura se, e somente se:
   $$\sigma^e \equiv H_1(\mathrm{ID}) r^{H_2(r,m)} \ (mod \ n)$$

Figura 2.2: Esquema de assinatura baseado em identidade de Shamir [Sha85]

1. **Inicializa** (Esta função é executada pela TA)
   (a) Gere um par de chaves RSA $((n, e), d)$, onde n, e e d são definidos tal como no RSA.
   (b) A chave pública mestra é $mpk := (n, e)$.
   (c) A chave secreta mestra é $msk := d$.
2. **Gera-Chaves** (Executado entre o usuário U (com identidade ID) e a TA)
   - U:
     (a) Escolhe a chave secreta $s \in \mathbb{Z}_n^*$.
     (b) Calcula $v := g^{-s} \ mod \ n$.
     (c) Envia v para TA e executa uma prova de conhecimento nulo (Protocolo-Identificação) com a TA, para provar para ela que conhece o valor s, sem revelá-lo. Obs: no artigo original Girault propõe um protocolo para isso, mas ele não será apresentado.
   - TA:
     (a) Seja ID a identidade do usuário U.
     Calcula $P_U := (v - \mathrm{ID})^d \ mod \ n$ e envia para U.
   - U:
     (a) Neste ponto, U pode verificar se P_U é de fato a sua chave pública:

$$P_U^e + \mathrm{ID} \equiv g^{-s} \ (mod \ n).$$

Figura 2.3: Esquema de Geração de Chaves de Girault baseado no RSA

A principal vantagem deste paradigma é a combinação da inexistência de certificados digitais para as chaves públicas (certificação implícita) com a não custódia das chaves privadas por parte da TA.

À época da publicação de Girault, em que propôs o modelo de chave pública autocertificada [Gir91], não havia formalizações estabelecidas sobre a segurança de protocolos criptográficos de chave pública, e nem sobre as respectivas técnicas de demonstração. Por causa disto, os esquemas propostos sob este modelo, tanto os de Girault quanto os de outros pesquisadores, não continham demonstrações de segurança. Posteriormente, muitos destes esquemas foram quebrados.

No trabalho original [Gir91], Girault alegou que o esquema de geração de chaves atinge o nível 3 de confiança na TA, sem requerer a distribuição de certificados digitais. Entretanto, na análise de segurança do esquema realizada no trabalho de Saeednia [Sae03], foi mostrado que a TA é capaz de computar a chave privada dos usuários, portanto rebaixando o esquema ao nível 1 de Girault, o mesmo nível do modelo IBC. Uma correção ao modelo foi sugerida neste artigo, porém o modelo resultante perde a principal característica do modelo original, a noção de autocertificação.

Com o estabelecimento das noções formais de segurança para diversos tipos de protocolo, e a quebra de diversos protocolos que eram supostamente seguros, muitos pesquisadores tentaram obter esquemas demonstravelmente seguros baseados no modelo de chave pública autocertificada. Porém, os esquemas que obtiveram maior sucesso se encaixam em modelos que são variações do modelo original de Girault, dentre eles *assinatura autocertificada* [LK02, Sha07] e *certificado implícito* [BGV02].

2.2.1 Certificado Implícito

Um certificado tradicional (PKI) consiste de um conjunto de informações produzido por uma CA que liga uma chave pública a uma entidade U. Ele é composto minimamente por: I_U, a identidade de U; a chave pública W_U do usuário; e a assinatura $s_{\mathrm{CA,\,U}}$ da CA sobre o par (I_U, W_U). Podemos então enxergar o certificado como a tripla $(I_U, W_U, s_{CA,U})$, onde a assinatura $s_{CA,U}$ liga I_U a W_U. A ligação é explícita, pois há um componente explícito no certificado, a assinatura, para esta função. Se a assinatura é inforjável, então a ligação obtida é chamada *forte*.

Um certificado implícito [BGV02] é um outro mecanismo para obter a ligação entre uma chave pública e uma entidade. Neste paradigma, o certificado consiste de (I_U, B_U), onde B_U é um *valor de reconstrução* (público) da chave pública, o qual é obtido da execução de um protocolo entre U e CA, em que cada entidade utiliza um valor privado. A chave pública do usuário U pode ser reconstruída por $W_U := f(B_U, I_U, W_{CA})$, onde f é uma função pública. A chave privada do usuário U é calculada por $s_U = h(x_U, B_U, I_U, W_{CA})$,

onde h é uma função pública, e x_U é um valor privado de U (o mesmo valor privado utilizado durante a execução do protocolo que gerou B_U).

Como exemplo de esquema que utiliza certificados implícitos, apresentamos na Figuras 2.4 e 2.5 o esquema Elliptic Curve Qu-Vanstone (ECQV) para geração de certificados implícitos. O esquema de geração de certificados implícitos ECQV tem como participantes um usuário (U) e uma autoridade certificadora (CA). Ao final da execução do esquema o usuário A terá um par de chaves e obterá da CA um certificado implícito para este par de chaves. Mais detalhes sobre o paradigma de certificados implícitos e sobre o esquema ECQV, incluindo uma demonstração de sua segurança, podem ser obtidos em [BCV09, Str08, BGV02, Cer04].

2.3 Criptografia de Chave Pública Sem Certificados (CL-PKC)

O modelo de criptografia de chave pública sem certificados (CL-PKC) foi proposto por Al-Riyami e Paterson em 2003 [ARP03]. Este é um modelo intermediário entre o modelo de certificação tradicional (PKI) e o modelo baseado em identidades (IB-PKC). Por um lado, não são necessários certificados ligando a chave pública à entidade correspondente para garantir a autenticidade das chaves públicas; esta é uma característica do modelo IB-PKC. Por outro lado, no modelo CL-PKC não há custódia de chaves por parte da TA. Esta última característica está também presente em PKI.

Em sistemas CL-PKC há uma autoridade confiável, chamada de centro de geração de chaves (KGC), a qual é responsável por gerar *chaves privadas parciais* para as entidades do sistema. Quando uma entidade A solicita a chave privada parcial à KGC, a KGC verifica se a entidade é quem diz ser (verifica a identidade ID_A). Se for, a KGC computa a chave privada parcial de A (D_A) utilizando ID_A, a chave mestra s, e os parâmetros públicos do sistema, *params*. A chave privada parcial D_A é então enviada através de um canal confidencial e autêntico para a entidade A.

Ao receber D_A, a entidade A executa o algoritmo de geração da chave privada com os argumentos: chave privada parcial D_A e valor secreto x_A (valor escolhido por A e mantido sempre em sigilo), e então obtém a chave privada completa $S_A = (x_A, D_A)$, a qual será usada como chave de decriptação (no contexto de encriptação) ou chave de assinatura (no contexto de assinatura).

Pelo modo como a chave secreta foi construída, podemos ver que a KGC não tem conhecimento de S_A, já que esta foi obtida utilizando o valor secreto de A, além de D_A. Para computar a sua chave pública P_A, a entidade A utiliza somente o valor secreto x_A.

Podemos também ver que não há dependência temporal na construção do par de

1. **Inicializa** (Executado pela CA)
   (a) Escolhe uma curva elíptica E com gerador G de ordem n.
2. **Gera Par de Chaves e Certificado** (Executado entre o usuário e a TA)
   - U:
     (a) Escolhe $k \in_R \mathbb{Z}_n^*$. Calcula o ponto $R := kG$.
     (b) Envia o ponto R para a CA, através de um canal autêntico.
     (c) Neste momento, U obteve o par de chaves *inicial* (k, R), onde k é a chave privada e R é a chave pública.
   - CA:
     (a) Recebe o ponto R de U, através do canal autêntico.
     (b) Escolhe $q \in_R \mathbb{Z}_n^*$. Calcula o ponto $Q := qG$.
     (c) Calcula $D := Q + R$. Este é o valor usado para a *reconstrução* da chave pública de U.
     (d) Agrupa as informações que constarão no certificado, tais como: a identidade de U, a identidade da CA, o uso para o qual o par de chaves foi destinado, o número serial, e o prazo de validade do certificado. Denote este conjunto de informações por I_U.
     (e) O certificado implícito é então $IC := (I_U, D)$.
     (f) Assina IC: $s := h(\text{IC})q + x_{\text{CA}}$, onde x_{CA} é a chave privada da CA.
     (g) Envia para U o certificado $C := (\text{IC}, s)$, por um canal possivelmente inseguro.
   - U:
     (a) Recebe C. Extrai IC e s de C.
     (b) Valida o certificado C, verificando se o certificado implícito IC e a assinatura s foram gerados pela CA: $h(\text{IC})R + sG \stackrel{?}{=} h(\text{IC})D + P_{\text{CA}}$.
     (c) Se o certificado é válido, prossegue calculando as chaves:
     (d) Calcula a chave privada $x_u := s + k \times h(\text{IC})(mod\ n)$.
     (e) A chave pública de U pode ser calculada utilizando a informação de reconstrução D, o certificado implícito IC e a chave pública P_{CA} da CA: $P_U := h(\text{IC})D + P_{\text{CA}}$. Alternativamente, U pode calcular a sua chave pública fazendo $P_U := x_u G$.
     (f) O par de chaves *final* de U é (x_u, P_U).

Figura 2.4: Esquema ECQV para Geração de Certificados Implícitos

Validação do Certificado (Executado por um usuário B)

1. Suponha que a entidade B recebeu um certificado C e uma chave pública *inicial* R, ambos supostamente provenientes da entidade A.

2. Suponha também que B sabe a identidade e a chave pública da CA (obtidos por canal autêntico).

3. B então valida o certificado C, verificando se o certificado implícito $IC = (I_A, D)$ e a assinatura s foram gerados pela CA: $h(IC)R + sG \stackrel{?}{=} h(IC)D + P_{CA}$.

4. Se o certificado é válido, a chave pública *final* que será reconstruída é autêntica. E então B calcula a chave pública *final* de A: $P_A := h(IC)D + P_{CA}$.

Figura 2.5: Esquema ECQV - Validação do Certificado

chaves, no sentido de que a chave pública pode ser computada antes ou depois da chave privada, pois uma não depende do valor da outra. A possibilidade de receber a chave privada parcial da KGC, e então computar a chave privada *após* ter computado a chave pública, é uma das características do modelo CL-PKC que permite a implementação de *workflows criptográficos* [ARP03].

No modelo CL-PKC, a distribuição de chaves é mais simples do que no modelo PKI. Em esquemas CLS (Assinatura Sem Certificados), a chave pública pode ser enviada ao verificador juntamente com a assinatura e a mensagem assinada. Já em esquemas CLE (Encriptação Sem Certificados), a chave pública precisa ser disponibilizada em um diretório público.

No contexto de encriptação, o remetente (A) não precisa fazer verificação alguma sobre a chave pública do destinatário, pois em esquemas CLE seguros a certificação é implícita. Ou seja, se o adversário prover a A uma chave pública falsa (P_B^*) no lugar da chave pública legítima do destinatário (B), então o adversário (ou outra entidade, inclusive A) não será capaz de decifrar a mensagem que foi cifrada por A com o uso de P_B^*. Em outras palavras, B (e somente B) conseguirá decifrar a mensagem se, e somente se, A utilizou a chave pública legítima de B.

Analogamente, no contexto de assinatura, o verificador não precisa fazer qualquer verificação sobre a chave pública do assinador, pois a certificação ocorre implicitamente.

2.4 Criptografia de Chave Pública Baseada em Certificados (CB-PKC)

A criptografia de chave pública baseada em certificados (CB-PKC) foi introduzida por Gentry em 2003 [Gen03]. Neste trabalho, o autor apresentou como seria a encriptação sob tal paradigma (CBE). Esquemas de assinatura sob o modelo baseado em certificados (CBS), juntamente com um modelo de segurança foram primeiramente propostos no trabalho de Kang et al [KPH04]. Refinamentos do modelo de segurança para CBS e outros esquemas foram propostos em Li et al [LHM+07] e Liu et al [LAS07, LBSZ08].

A noção básica por trás da CBE é que, para decifrar, os usuários precisam de um certificado atualizado, bem como de uma chave secreta. Um criptossistema genérico sob o modelo CBE combina um esquema PKE com um esquema IBE, trazendo as melhores características de ambos: a certificação é implícita (ao contrário de PKE e semelhante a esquemas IBE) e não há custódia de chaves (diferentemente de IBE e semelhante a PKE).

Em esquemas CBE, dizemos que há *certificação implícita* quando um usuário deve ser capaz de encriptar (seguramente) uma mensagem para outro usuário meramente conhecendo a chave pública e parâmetros públicos da CA. E, além disto, o recipiente somente

consegue decifrar tal mensagem se o seu certificado é recente (do inglês *fresh*).

O problema da expiração e o problema da revogação do par de chaves podem ser resolvidos de maneira simples, com a CA parando de enviar certificados para os usuários revogados. Sem certificados para o período corrente não é possível obter a chave de decriptação (ou assinatura). Isto elimina a necessidade de uma estrutura complexa para tratar a revogação do par de chaves, como é o caso das soluções CRL e OCSP em PKI.

Como na PKI, os usuários escolhem o seu próprio par de chaves e requisitam um certificado da CA que valida a chave pública. O certificado recebido funciona como uma prova *explícita* de certificação (para convencer uma terceira parte, por exemplo) e também como uma *chave privada parcial* (a qual deve ser usada para computar a chave privada completa).

Dado que somente os usuários são capazes de computar as suas chaves privadas, o esquema não apresenta custódia de chaves por parte da CA. A única informação obtida da CA, o certificado, não precisa ser transmitido por um canal seguro (autêntico e sigiloso); isto facilita a sua distribuição. As únicas entidades que precisam realizar consultas a CA (para obtenção dos certificados do período corrente) são os usuários que possuem par de chaves no sistema. As entidades que precisam encriptar (verificar) mensagens para (de) usuários do sistema não precisam fazer quaisquer consultas a CA. Esta eliminação das consultas de terceiros (*third-party queries*) à CA traz uma grande redução no número de consultas que devem ser suportadas pelos servidores da CA.

No paradigma CB-PKC, apesar da certificação ocorrer implicitamente, ainda há certificados, e a obtenção frequente de certificados recentes pode ser um *gargalo* em algumas aplicações, devido ao elevado consumo de largura de banda.

Um esquema CBE é uma tupla de 6 algoritmos (Gen_{IBE}, Gen_{PKE}, $Upd1$, $Upd2$, Enc, Dec), descritos na Figura 2.6.

Como podemos ver no esquema genérico, CBE suporta encriptação para o futuro, o remetente (A) pode encriptar uma mensagem para o destinatário (B) utilizando um período de tempo i no futuro como argumento para o algoritmo Enc. Com isto, o usuário B somente conseguirá decriptar a mensagem no período de tempo i, momento em que este receberá da CA o certificado corresponde a este período.

O certificado $Cert'_i$ tem um papel duplo no modelo CBE. Ele pode ser visto como uma assinatura da CA sobre as informações do usuário (ID e chave pública PK_{PKE})), e como tal pode ser verificado publicamente utilizando a chave pública PK_{IBE} da CA, tendo portanto o papel de *prova explícita de certificação*. Mas também é utilizado como chave de decriptação, de forma a permitir a certificação implícita, o que elimina a necessidade de consulta de terceiros ao status do certificado.

No artigo original de Gentry [Gen03], são apresentados quatro esquemas CBE concretos. O autor apresenta uma sequência de esquemas, onde o esquema seguinte é uma

1. **Gen-IBE**
   `Entrada`: 1^{k_1}, para um parâmetro de segurança k_1, e (opcionalmente) o número total de períodos t.
   `Saída`: chave privada mestra da CA SK_{IBE} e parâmetros públicos *params*, dentre os quais a chave pública PK_{IBE} da CA.

2. **Gen-PKE**
   `Entrada`: 1^{k_2}, para um parâmetro de segurança k_2, e (opcionalmente) o número total de períodos t.
   `Saída`: a chave privada SK_{PKE} e a chave pública PK_{PKE} do usuário.

3. **Upd1** (Executado pela CA no início do período de tempo i)
   `Entrada`: SK_{IBE}, período de tempo i, identidade do usuário ID, PK_{PKE}, *params*.
   `Saída`: certificado $Cert'_i$, o qual é enviado ao Usuário.

4. **Upd2** (Executado pelo Usuário no início do período de tempo i, após `Upd1`)
   `Entrada`: período de tempo i, $Cert'_i$, *params* e (opcionalmente) $Cert_{i-1}$.
   `Saída`: certificado $Cert_i$.

5. **Enc** (Executado pelo Usuário remetente dentro do período de tempo i)
   `Entrada`: *params*, i, ID, PK_{PKE}, mensagem m).
   `Saída`: mensagem cifrada c.

6. **Dec**
   `Entrada`: params, $Cert_i$, SK_{PKE}, mensagem cifrada c.
   `Saída`: mensagem em claro m ou $\perp$ (indicando falha).

Figura 2.6: Esquema CBE Genérico

versão mais refinada do que o anterior, em termos de segurança e/ou de eficiência.

O primeiro esquema é chamado BasicCBE. O segundo esquema, chamado FullCBE, foi obtido a partir do primeiro através da transformação de Fujisaki-Okamoto [FO99], e é demonstrado seguro contra ataques adaptativos de texto cifrado escolhido, sob o modelo de segurança para esquemas CBE definido no artigo, com uma demonstração no modelo do oráculo aleatório. O terceiro esquema, CBE com Coberturas de Conjuntos (*subset covers*), mantém a segurança do anterior e é mais eficiente em termos de número de certificados que precisam ser gerados pela CA. O quarto esquema, CBE Incremental com Cobertura de Conjuntos, também mantém a segurança e é o mais eficiente deles. O esquema FullCBE é apresentado na Figuras 2.7 e 2.8.

1. **Inicializa** (CA)
   `Entrada`: 1^k, para um parâmetro de segurança k.

   (a) Escolhe a chave secreta mestra $s \in \mathbb{Z}/q\mathbb{Z}$.

   (b) Escolhe os parâmetros públicos do sistema: $params := (q, \mathbb{G}_1, \mathbb{G}_2, e, n, P_{pub}, H_1, H_2, H_3, H_4)$, onde $\mathbb{G}_1$ e $\mathbb{G}_2$ são grupos de ordem prima q, $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ é um emparelhamento bilinear admissível, P é um gerador de $\mathbb{G}_1$, e as funções de hash são:
   $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$;
   $H_2 : \mathbb{G}_2 \to \{0,1\}^n$;
   $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$;
   $H_4 : \{0,1\}^n \to \{0,1\}^n$.
   O espaço de mensagens é $\mathcal{M} = \{0,1\}^n$. O espaço de texto cifrados é $\mathcal{C} = \mathbb{G}_1^* \times \{0,1\}^n \times \{0,1\}^n$.

   (c) Calcula a chave pública mestra $P_{pub} = sP$.

2. **Gera Par de Chaves Inicial** (A)

   (a) Escolhe $t_A \in_R \mathbb{Z}_q^*$. Faz $N_A := t_A P$.

   (b) O par de chaves *inicial* é (t_A, N_A).

   (c) Denote a *identificação* de A por $A_{info} = (ID_A, N_A)$.

3. **Certifica** (CA)
   `Entrada`: um período de tempo i, ID_A, chave pública inicial N_A.

   (a) Calcula a chave pública $P_A := H_1(P_{pub}, i, ID_A, N_A)$ e o certificado $Cert_A^i := sP_A$.

   (b) Envia $Cert_A^i$ para A.

Figura 2.7: Esquema de Encriptação Baseado em Certificados de Gentry (FullCBE)

4. **Gera Par de Chaves**
   `Entrada`: certificado $Cert_A^i$ para o período de tempo i.
   (a) Calcula $P'_A := H_1(A_{info})$ e a chave secreta $S_A := Cert_A^i + t_A P'_A$.

5. **Encripta**
   `Entrada`: mensagem m, identificação $A_{\text{info}} = (ID_A, N_A)$, período de tempo i.
   (a) Escolhe $\sigma \in_R \{0,1\}^n, r := H_3(m, \sigma)$;
   (b) $P_A := H_1(P_{pub}, i, ID_A, N_A)$, $P'_A := H_1(A_{info})$;
   (c) $U := rP$, $V := \sigma \oplus H_2((e(P_{pub}, P_A)e(N_A, P'_A))^r)$, $W := m \oplus H_4(\sigma)$;
   (d) Devolve $C := (U, V, W)$.

6. **Decripta**
   `Entrada`: $C = (U, V, W)$, chave secreta S_A.
   (a) $\sigma := V \oplus H_2(e(S_A, U))$;
   (b) $r := H_3(\sigma, m)$;
   (c) $m := W \oplus H_4(\sigma)$;
   (d) Verifica se $U \stackrel{?}{=} rP$. Se for, devolve m. Senão, rejeita.

Figura 2.8: Esquema de Encriptação Baseado em Certificados de Gentry (FullCBE) (Cont.)

Capítulo 3

Segurança Demonstrável

3.1 Introdução

Nesta seção apresentamos um breve histórico da área de segurança demonstrável.

3.1.1 Ataques Genéricos

Com o passar do tempo, muitos tipos diferentes de ataques foram identificados sobre as diversas primitivas e protocolos criptográficos. Antes de entrarmos em detalhe sobre os ataques específicos para cada tipo de primitiva ou protocolo, descrevemos dois tipos genéricos de ataques:

- Um *ataque passivo* (*passive attack*) é aquele em que o adversário somente consegue ler as mensagens que passam pelo canal de comunicação. Um adversário deste tipo, também chamado *adversário passivo* ou *observador* (*eavesdropper*), apenas ameaça a confidencialidade dos dados.

- Um *ataque ativo* (*active attack*) é aquele em que o adversário é capaz de, além de ler mensagens, adicionar, remover ou modificar mensagens no canal de comunicação. Ou seja, neste tipo de ataque supõe-se que o adversário tem *controle total* sobre o canal de comunicação. Um adversário deste tipo, também chamado *adversário ativo*, ameaça a integridade, autenticidade e confidencialidade dos dados.

3.1.2 Segurança Incondicional e Sigilo Perfeito

Um esquema criptográfico é dito ser *incondicionalmente seguro* se este esquema não revela informação suficiente para que o adversário possa *quebrá-lo*, supondo que o adversário tenha poder computacional ilimitado. A *segurança incondicional* de um esquema é uma

medida da Teoria da Informação que diz o quão próximo um esquema se encontra de ser incondicionalmente seguro. Este conceito se aplica a qualquer tipo de esquema criptográfico, por exemplo, de encriptação, assinatura ou acordo de chaves, ou ainda simétrico ou assimétrico. A definição exata de *quebra* depende do modelo de segurança adotado para o tipo de esquema em questão.

O conceito de segurança incondicional se aplica a qualquer tipo de esquema criptográfico, enquanto o conceito de *sigilo perfeito* é o conceito de segurança incondicional aplicado a esquemas de encriptação simétricos.

Sigilo Perfeito

Antes do trabalho seminal de Shannon [Sha49], a criptografia era considerada a *arte* de projetar esquemas que provessem segurança, mais especificamente sigilo. Como tal, os esquemas obtidos não possuíam quaisquer garantias de que fossem realmente seguros, e a confiança dos usuários em um dado esquema era consequência principalmente da resistência deste a tentativas de ataque realizadas ao longo dos anos.

O trabalho [Sha49] propôs a primeira formalização da noção de sigilo para esquemas de encriptação simétricos, definindo o significado de *sigilo perfeito*:

Definição 1 *(**Sigilo Perfeito**). Um esquema de encriptação simétrico* $\pi = (Gen, Enc, Dec)$ *sob um espaço de mensagens* $\mathcal{M}$ *tem* sigilo perfeito *se, e somente se, para toda distribuição de probabilidade sobre* $\mathcal{M}$*, para toda mensagem* $m \in \mathcal{M}$*, e para todo texto cifrado* $c \in \mathcal{C}$ *, temos*

$$P[M = m|C = c] = P[M = m].$$

(Obs: M e C são variáveis aleatórias sobre $\mathcal{M}$ *e* $\mathcal{C}$ *, respectivamente. Supomos* $P[C = c] > 0$*, pois caso contrário a probabilidade condicional não estaria definida).*

A definição acima pode ser interpretada como: um esquema de encriptação simétrico tem sigilo perfeito se as distribuições de probabilidade das mensagens e dos textos cifrados são independentes.

Intuitivamente, a definição diz que o conhecimento da distribuição dos textos cifrados ($P[C = c]$) não trará qualquer informação adicional ao adversário a respeito da distribuição das mensagens, além do que este já sabe a respeito da distribuição das mensagens (lado direito). Ou seja, um texto cifrado c não traz qualquer informação a respeito da mensagem que o originou. A partir de algumas substituições, podemos chegar a uma definição equivalente à anterior: $P[C = c|M = m] = P[C = c]$.

Considerações Históricas

Um exemplo de esquema de encriptação simétrico que atinge o nível de sigilo perfeito é o *one-time pad*, criado por Vernam em 1926 [Ver26], apesar de que no momento da publicação do artigo, ainda não havia o conceito de sigilo perfeito. A noção de sigilo perfeito para esquemas de encriptação simétricos só seria introduzida em 1949 por Shannon em [Sha49], no qual ele também apresentou teoremas caracterizando o conceito de sigilo perfeito, e mostrou que o one-time pad atinge o nível de sigilo perfeito. Dentre os teoremas apresentados neste último trabalho, encontra-se aquele que provê uma condição necessária para atingir o nível de sigilo perfeito: para um esquema de encriptação simétrico possuir sigilo perfeito é necessário que o tamanho da chave seja *pelo menos* igual ao tamanho da mensagem.

O *one-time pad* é um esquema onde a chave tem exatamente o mesmo tamanho da mensagem a ser cifrada, e o procedimento de cifração é simplesmente o *ou-exclusivo* da mensagem com a chave: $c := m \oplus k$. Este esquema possui sigilo perfeito, porém o custo para utilizá-lo para cifrar mensagens longas é muito elevado: uma dada chave (correspondente à uma instância do esquema) só pode ser utilizada para cifrar uma mensagem, pois caso contrário, se a chave inteira (ou uma parte desta) for reutilizada para cifrar uma outra mensagem, o esquema resultante é inseguro.

Para cifrar mais de uma mensagem no *one-time pad*, deve ser gerada uma chave cujo tamanho total seja igual a soma dos tamanhos das mensagens, o que inviabiliza o uso deste esquema na prática para cifrar muitas (ou grandes) mensagens, por causa dos custos envolvidos para a distribuição de uma chave secreta imensa entre as entidades comunicantes.

Indistinguibilidade Perfeita

Uma outra noção de sigilo, equivalente à noção de sigilo perfeito, é a noção de *indistinguibilidade perfeita* [KL08]:

Definição 2 *(**Indistinguibilidade Perfeita**) Um esquema de encriptação simétrico $\pi = (Gen, Enc, Dec)$ sobre um espaço de mensagens $\mathcal{M}$ tem sigilo perfeito se, e somente se, para toda distribuição de probabilidade sobre $\mathcal{M}$, para todo $m_0, m_1 \in \mathcal{M}$, e todo $c \in \mathcal{C}$, temos*

$$P[C = c | M = m_0] = P[C = c | M = m_1].$$

(Obs: M e C são variáveis aleatórias sobre $\mathcal{M}$ e $\mathcal{C}$, respectivamente. Supomos que $P[M = m_0] > 0$ e $P[M = m_1] > 0$).

Esta definição mostra que, para quaisquer mensagens m_0 e m_1, e texto cifrado c, a probabilidade do texto cifrado c corresponder a m_0 é igual à probabilidade do texto cifrado

c corresponder a m_1. Ou seja, em um esquema de encriptação com sigilo perfeito é impossível distinguir a encriptação de uma mensagem m_0 da encriptação de uma mensagem m_1.

Indistinguibilidade Adversarial Perfeita

Uma outra definição equivalente de sigilo perfeito é obtida com a introdução de um *experimento* envolvendo um adversário $\mathcal{A}$ e um *desafiante* $\mathcal{C}$. Um desafiante é uma entidade imaginária responsável por gerar os parâmetros do sistema e testar o adversário, mostrando que $\mathcal{A}$ não é capaz de distinguir entre a encriptação de uma mensagem e a encriptação de outra mensagem. Nesta definição, supomos que o adversário é passivo (observador).

O experimento é definido para qualquer esquema de encriptação simétrico $\pi = (Gen, Enc, Dec)$ sobre o espaço de mensagem $\mathcal{M}$ e para qualquer adversário $\mathcal{A}$. Denotamos por $PrivK^{eav}_{\mathcal{A},\pi}$ uma execução do experimento, para $\mathcal{A}$ e π fixos. A saída de uma execução do experimento é denotada por $PrivK^{eav}_{\mathcal{A},\pi} \in \{0,1\}$, sendo utilizado o valor 1 se, e somente se, o adversário tem sucesso em distinguir os textos cifrados. Segue o experimento:

Experimento de Indistinguibilidade para Adversários Observadores

- $\mathcal{A}$ recebe 1^n e devolve mensagens $m_0, m_1 \in \mathcal{M}$.
- $\mathcal{C}$ executa $k := Gen(1^n)$, obtendo a chave do sistema. Escolhe o bit $b \in_R \{0,1\}$, e envia $c := Enc_k(m_b)$ para $\mathcal{A}$ (c é chamado de *texto cifrado de desafio*).
- $\mathcal{A}$ devolve um bit b'.
- $\mathcal{C}$ verifica se $b' \stackrel{?}{=} b$. Se forem iguais, $\mathcal{C}$ devolve 1 como saída do experimento, caso contrário, devolve 0.

A definição de sigilo perfeito com base neste experimento é a de que o máximo que um adversário pode fazer para determinar b é equivalente (tem a mesma probabilidade de) a escolher b' aleatoriamente. Formalmente:

Definição 3 ***(Indistinguibilidade Adversarial Perfeita Contra Adversários Observadores)*** *Um esquema de encriptação simétrico $\pi = (Gen, Enc, Dec)$ sobre o espaço de mensagem $\mathcal{M}$ tem* sigilo perfeito *se, e somente se, para todo adversário observador $\mathcal{A}$, temos*

$$P[PrivK^{eav}_{\mathcal{A},\pi} = 1] = \frac{1}{2}.$$

3.1.3 Segurança Computacional

Para provar que um esquema é incondicionalmente seguro, é necessário obter um limitante inferior para o tempo necessário para quebrá-lo, para qualquer adversário. Para isto, é suficiente uma demonstração de que todo algoritmo probabilístico de tempo polinomial (PPT) não é capaz de quebrá-lo.

Até o momento não existem demonstrações de segurança incondicional para esquemas criptográficos modernos. Uma demonstração deste tipo requer avanços significativos em teoria da complexidade, os quais os pesquisadores desta área acreditam estarem longe de ocorrer. Em particular, a existência de uma demonstração deste tipo implicaria em $P \neq NP$ [KL08].

Devido a estas dificuldades e o fato de esquemas com segurança incondicional serem inviáveis para uso prático, foram propostas noções de segurança que se fundamentam na quantidade de poder computacional disponível ao adversário; daí o nome *segurança computacional* para esta classe de noções de segurança. Diferentemente dos esquemas com segurança incondicional, os quais não podem ser quebrados nem mesmo por um adversário com poder computacional ilimitado, os esquemas com segurança computacional *sempre* podem ser quebrados se o adversário tiver recursos computacionais suficientes.

O ponto chave em segurança computacional é conseguir determinar com precisão qual a quantidade de recursos necessária para um adversário quebrar o esquema criptográfico, em função do parâmetro de segurança (n). Depois que esta função (quantidade de recursos em função de n) foi determinada, subtrai-se um *fator de segurança* (também em função de n) para cobrir eventuais erros introduzidos na análise de segurança. A função resultante é então utilizada para instanciar o esquema: escolhe-se um valor para o parâmetro de segurança n, e a partir daí deriva-se o tamanho dos parâmetros do sistema, dentre eles o tamanho das chaves.

A importância da precisão ao determinar a função reside no fato de que a eficiência do esquema (inversamente proporcional ao tamanho adotado para os parâmetros do sistema) é proporcional ao valor desta função, e portanto, quanto mais *justa* for a função, mais eficiente será o esquema para qualquer valor do parâmetro de segurança.

Uma observação importante [MvOV96] sobre segurança computacional é a de que esquemas de encriptação de chave pública não podem ser incondicionalmente seguros (sigilo perfeito) pois, dados um texto cifrado c e a chave pública P que foi utilizada para encriptar a mensagem que originou c, então o adversário pode então realizar um ataque de força bruta cifrando todas as mensagens do espaço de mensagens $\mathcal{M}$, até encontrar a mensagem cujo texto cifrado correspondente seja igual a c.

Seguem abaixo notações e definições de conceitos que serão utilizados posteriormente.

Definições e Notações

Seja x um valor e y uma variável. Então, $y := f(x)$ denota a atribuição do valor $f(x)$ à variável y. Analogamente, se $\mathcal{A}$ é um algoritmo *determinístico*, então $y := \mathcal{A}(x)$ denota a atribuição à variável y do valor devolvido pela execução de $\mathcal{A}$ na entrada x . Se durante a sua execução $\mathcal{A}$ tem acesso a um oráculo $\mathcal{O}$, representamos esta execução por $\mathcal{A}^{\mathcal{O}}$.

Seja $\mathcal{A}$ um algoritmo probabilístico. Denotamos por $y \stackrel{R}{:=} \mathcal{A}(x)$ a atribuição à variável y do valor devolvido pela execução de $\mathcal{A}$ na entrada x, com o uso de uma fita R gerada aleatoriamente para esta execução específica. Para denotar a execução de $\mathcal{A}$ utilizando uma fita específica R, escrevemos $y := \mathcal{A}(x; R)$. Particularmente, se S é um conjunto finito, então denotamos por $y \in_R S$ a atribuição a y de um valor amostrado aleatoriamente e uniformemente de S.

Ao analisar a probabilidade de ocorrência de um evento S dependente da ocorrência de outros eventos, denotaremos este evento por um predicado $\phi(x_1, ..., x_n)$ e denotaremos tal probabilidade por $Pr[\phi(x_1, ..., x_n) : x_1 := X_1, ..., x_n := X_n]$, onde $X_1, ..., X_n$ são as distribuições de probabilidade dos outros eventos relevantes.

Um *parâmetro de segurança* (n) é um parâmetro de um algoritmo criptográfico que está diretamente relacionada ao esforço necessário para quebrá-lo. Ao instanciar o algoritmo, deve-se atribuir um valor a este parâmetro; quanto maior o valor, maior será o nível de segurança desta instância do algoritmo. Na especificação de algoritmos criptográficos, quando este valor aparece (pois normalmente é omitido), ele é o primeiro na lista de parâmetros do algoritmo, e é denotado por 1^n (notação unária), indicando uma entrada de tamanho n bits.

Definição 4 *(**Algoritmo Probabilístico de Tempo Polinomial (PPT)**). Seja n um parâmetro de segurança. Um algoritmo probabilístico $\mathcal{A}$ é dito de tempo polinomial em n se existe um polinômio p, tal que, para toda entrada x, $\mathcal{A}(x)$ termina dentro de $p(n)$ passos, independentemente do valor da fita aleatória utilizada por $\mathcal{A}$.*

Ao argumentar sobre a *vantagem* de um adversário sobre um dado esquema, ou sobre a probabilidade de uma dado evento de interesse, muitas vezes pretendemos mostrar que estas probabilidades são *pequenas*. Em segurança demonstrável, uma quantidade é dita *pequena* se ela é uma *função desprezível*, segundo a definição abaixo.

Definição 5 *(**Função Desprezível**). Seja n um parâmetro de segurança. Uma função f é* desprezível *em n se, para todo polinômio p, existe um inteiro $N(p)$ tal que $f(n) \leq \frac{1}{|p(n)|}$ para todo $n \geq N(p)$.*
Denotaremos por $negl(n)$ uma função desprezível genérica.

Apresentamos a seguir as noções de segurança computacional mais comuns na literatura, e a relação entre elas.

Segurança Semântica

Seja $\mathcal{M}$ um espaço de mensagens e f uma função definida sobre $\mathcal{M}$. Informalmente, supomos que para uma mensagem $m \in \mathcal{M}$, $f(m)$ representa alguma informação (parcial ou completa) sobre m.

Intuitivamente, a noção de *segurança semântica* [GM84] significa que deve ser difícil para um adversário obter qualquer informação parcial a respeito da mensagem m dado apenas o texto cifrado correspondente $c = Enc_k(m)$. Ou seja, deve ser difícil calcular o valor de $f(m)$ (qualquer que seja a função f), quando este tem somente acesso à $c = Enc_k(m)$.

Formalmente, dados os jogos:

- **Jogo 1.** $\mathcal{C}$ escolhe $m \in_R \mathcal{M}$. $\mathcal{A}_1$ deve calcular $f(m)$ sem conhecer m, conhecendo somente a descrição de f.
- **Jogo 2.** $\mathcal{C}$ escolhe $m \in_R \mathcal{M}$, calcula $c := Enc_k(m)$ e envia c a $\mathcal{A}_2$. $\mathcal{A}_2$ deve calcular $f(m)$, conhecendo somente a descrição de f e o valor c.
- **Jogo 3.** $\mathcal{A}_3$ escolhe uma função f^* com domínio $\mathcal{M}$. $\mathcal{C}$ faz $c := Enc_k(m)$ e envia c a $\mathcal{A}_3$. $\mathcal{A}_3$ deve calcular o valor de $f^*(m)$, conhecendo somente a descrição de f^* e o valor c.

Definição 6 *(**Segurança Semântica**) Seja A_i^* o evento em que o adversário vence o jogo i, tal como definido acima. Um esquema de encriptação simétrico (Gen, Enc, Dec) é* semanticamente seguro na presença de um adversário observador *se:*

$$Pr[A_3^*] < Pr[A_1^*] + n^{-c}$$

para uma constante c suficientemente grande, onde n é o parâmetro de segurança. Ou seja, o conhecimento do texto cifrado e a possibilidade de escolher a função (informação a ser extraída de m) não devem trazer vantagem significativa para o adversário.

Segurança Polinomial

A noção de *segurança polinomial* [GM84] pode ser definida através de um experimento (jogo) entre um adversário observador e um desafiante. O experimento tem como parâmetros um esquema de encriptação simétrico $\pi = (Gen, Enc, Dec)$, um adversário observador $\mathcal{A}$ e um parâmetro de segurança n. Denotamos por $PrivK^{eav}_{\mathcal{A},\pi}(n)$ uma execução do experimento, para $\mathcal{A}$, π e n fixos. A saída de uma execução do experimento é denotada por $PrivK^{eav}_{\mathcal{A},\pi}(n) \in \{0,1\}$, onde o valor 1 é utilizado se, e somente se, o adversário tem sucesso em descobrir o valor do bit b.

Experimento de Indistinguibilidade Computacional de Textos Cifrados

- O adversário $\mathcal{A}$ recebe 1^n e devolve um par de mensagens m_0 e m_1 de igual comprimento.
- $\mathcal{C}$ gera uma chave $k := Gen(1^n)$, escolhe um bit aleatório $b \in_R \{0,1\}$ e envia o texto cifrado de desafio $c := Enc_k(m_b)$ para $\mathcal{A}$.
- $\mathcal{A}$ devolve um bit b'.
- $\mathcal{C}$ verifica se $b' \stackrel{?}{=} b$. Se forem iguais, $\mathcal{C}$ devolve 1 como saída do experimento, caso contrário, devolve 0.

Definição 7 ***(Indistinguibilidade Computacional de Textos Cifrados)*** *Seja n um parâmetro de segurança. Um esquema de encriptação simétrico $\pi = (Gen, Enc, Dec)$ tem* textos cifrados indistinguíveis *sob a presença de um adversário observador se, para todo adversário PPT $\mathcal{A}$, existe uma função desprezível* negl *tal que:*

$$P[PrivK^{eav}_{\mathcal{A},\pi}(n) = 1] \leq \frac{1}{2} + negl(n),$$

onde a probabilidade é tomada sobre a fita aleatória de $\mathcal{A}$, e as fitas aleatórias usadas nos algoritmos que compõem o esquema π.

Segurança "Yao"

A noção de *segurança "Yao"* [Yao82] é baseada em conceitos da teoria da informação, e não será apresentada neste trabalho. Referimos o leitor ao trabalho [CDD07] para uma descrição simplificada desta noção.

Equivalência entre as Noções

Além de propor as noções de segurança semântica e indistinguibilidade (segurança polinomial), o trabalho [GM84] também provou que indistinguibilidade implica segurança semântica.

Em [MRS88] foi mostrado que a implicação inversa também é verdadeira, ou seja, segurança semântica implica indistinguibilidade, e também que noção de segurança Yao é equivalente às outras duas noções, concluindo que as três noções, polinomial, semântica e Yao, são equivalentes. Dentre estas três noções, a noção de segurança polinomial (indistinguibilidade) é a mais utilizada em demonstrações, pois é a mais fácil de se trabalhar do que as outras duas, e de acordo com a equivalência acima, um esquema de encriptação seguro sob esta noção também é seguro sob a noção de segurança semântica, a qual é a noção

mais intuitiva e tradicional de segurança para encriptação. Portanto, é "seguro" realizar demonstrações segundo a noção de indistinguibilidade.

A equivalência entre segurança semântica e indistinguibilidade provada por Micali et al [MRS88] vale somente contra adversários passivos (adversários que não realizam decriptações). Por muito tempo não se sabia se estas noções de segurança eram equivalentes contra adversários ativos (os quais podem realizar decriptações). De fato, segundo Koblitz e Menezes [KM06], este assunto não foi nem mesmo tratado na literatura até 2002. Somente em 2003 esta equivalência foi provada [WSI03]. Referimos o leitor a [Gol04] para provas desta equivalência contra ambos os tipos de adversário.

3.2 Modelos de Segurança

3.2.1 Modelos de Segurança para Encriptação de Chave Pública

Dois tipos de ataque historicamente considerados contra esquemas de encriptação, de chave privada ou de chave pública, são:

- **Ataque Apenas Com Texto Cifrado (KCA)**: supõe que o adversário tem acesso somente aos textos cifrados correspondentes a cada mensagem de uma lista de mensagens.
- **Ataque de Texto Claro Conhecido (KPA)**: supõe que o adversário tem acesso a uma lista $(m_1, c_1), ..., (m_n, c_n)$, onde m_i é uma mensagem e $c_i = Enc_{pk}(m_i)$, mas não tem controle sobre o valor das mensagens desta lista.

Estes modelos de ataques, embora importantes historicamente, não são considerados suficientemente seguros para esquemas de encriptação modernos, e portanto foram definidos modelos que suportam adversários mais poderosos. Definimos abaixo alguns destes modelos de segurança (noções de segurança) tradicionalmente aceitos para esquemas de encriptação de chave pública.

Definição 8 ***(Indistinguibilidade Contra Ataques Adaptativos de Texto Claro (Cifrado) Escolhido (IND-CPA, IND-CCA ou IND-CCA2))***
Seja $\pi = (Gen, Enc, Dec)$ um esquema de encriptação, n um parâmetro de segurança e $\mathcal{A}$ um adversário de π de tempo polinomial em n.
Seja $IND^{O_1,O_2}(\pi, \mathcal{A}, n)$ o seguinte jogo de segurança, onde O_1 e O_2 são os oráculos que o adversário $\mathcal{A}$ tem acesso na Fase 1 e na Fase 2, respectivamente:

1. $(pk, sk) := Gen(1^n)$.
2. *Fase 1.* $(m_0, m_1, state) := \mathcal{A}^{O_1}(pk)$.

3. Desafio. $b \in_R \{0,1\}$*;* $c := Enc_{pk}(m_b)$.

4. Fase 2. $b' := \mathcal{A}^{\,O_2}(c, state)$.

O adversário pode realizar consultas aos oráculos O_1 *e* O_2 *de modo adaptativo, ou seja, uma consulta pode depender das respostas das consultas anteriores).*

Dizemos que π *é IND-CPA/CCA/CCA2 seguro se, para todos os adversários* $\mathcal{A}$ *de tempo polinomial,* $|Pr[b' = b] - 1/2|$ *é desprezível. Em cada um dos níveis de segurança os oráculos* O_1 *e* O_2 *podem ser:*

- *CPA:* $O_1 = \{Enc_{pk}\}$, $O_2 = \{Enc_{pk}\}$.
- *CCA:* $O_1 = \{Enc_{pk}, Dec_{sk}\}$, $O_2 = \{Enc_{pk}\}$.
- *CCA2:* $O_1 = \{Enc_{pk}, Dec_{sk}\}$, $O_2 = \{Enc_{pk}, Dec_{sk}\}$

No nível CCA2, supomos também que $\mathcal{A}$ *não faz a consulta* $Dec_{sk}(c)$ *ao oráculo* O_2*, onde* c *é o desafio.*

Atualmente, um esquema de encriptação de chave pública é considerado *seguro* se ele é semanticamente seguro (ou, equivalentemente, tem indistinguibilidade de textos cifrados) sob ataques adaptativos de texto cifrado escolhido (IND-CCA2 adaptativos).

A escolha de um esquema de encriptação de chave pública baseado no nível de segurança deste esquema (IND-CPA, IND-CCA ou IND-CCA2 adaptativos) vai depender dos requisitos da aplicação onde este estará inserido. Em geral, esquemas com nível de segurança menor serão mais eficientes do que aqueles com nível de segurança maior. Portanto, em aplicações que demandam alto desempenho do esquema criptográfico, e em que se tem certeza de que o adversário não consegue, em hipótese alguma, obter decriptações de textos cifrados de sua escolha, esquemas IND-CPA são suficientes. Porém, em aplicações em que não se pode concluir isto com certeza, deve ser empregado um esquema IND-CCA ou IND-CCA2.

Seguem abaixo alguns outros conceitos importantes sobre encriptação (simétrica ou assimétrica).

Definição 9 *(**Não-Maleabilidade** (Non-Malleability)). Um esquema de encriptação é dito* não-maleável *se, dado um texto cifrado* c *(correspondente a uma mensagem* m *desconhecida), é computacionalmente inviável obter um texto cifrado válido* c' *correspondente a uma mensagem* m'*, onde* m' *está "relacionada" com* m *por uma função conhecida.*

Teorema 1 *(**Equivalência entre Seg. Semântica e Não-Maleabilidade sob Ataques CCA2**). Um esquema de encriptação é não-maleável sob ataques CCA2 se, e somente se, este esquema é semanticamente seguro contra ataques CCA2 (IND-CCA2 seguro).*

Para mais detalhes sobre não-maleabilidade e resultados sobre não-maleabilidade de alguns esquemas de encriptação, referimos o leitor a Dolev et al [DDN91] que introduziu o conceito, e também aos livros [MvOV96, Sma10].

Definição 10 ***(Plaintext Awareness)****. Um esquema de encriptação é dito "plaintext-aware" se é computacionalmente inviável para um adversário produzir um texto cifrado c sem ter conhecimento da mensagem m correspondente.*

A noção de *plaintext awareness* é uma noção forte de segurança para encriptação, pois esquemas que possuem esta propriedade *não permitem* a realização de ataques de texto cifrado escolhido (CCA); o adversário pode até tentar estes ataques, mas eles não vão ter efeito algum contra o esquema.

3.2.2 Modelos de Segurança para Assinatura de Chave Pública

Destacamos três modelos de ataque para esquemas de assinatura de chave pública:

- **Ataque Apenas com Chave Pública** (Public Key-only Attack). Também conhecido como *No-Message Attack* (NMA): o adversário tem acesso apenas à chave pública dos usuários do sistema.
- **Ataque de Mensagem Conhecida**: o adversário tem acesso apenas a uma lista (m_1,s_1),...,(m_n,s_n), onde m_i é uma mensagem e $s_i = Sign_{sk}(m_i)$, onde sk é a chave privada do usuário alvo do ataque.
- **Ataque de Mensagem Escolhida (CMA)**: o adversário pode construir a lista de mensagens a serem assinadas pelo usuário alvo, ou seja, ele tem controle sobre o valor de $m_i, i = 1, ..., n$, na lista do ataque anterior.

Podemos também considerar três metas adversariais, ou tipos de comprometimento:

- **Quebra Total**: o adversário é capaz de determinar a chave privada sk do usuário alvo.
- **Falsificação Seletiva**: o adversário recebe uma mensagem m e é capaz de produzir uma assinatura válida s para m, ou seja, consegue obter s tal que $Ver_{pk}(m, s) = True$, onde pk é a chave pública do usuário alvo.
- **Falsificação Existencial** (EUF): o adversário é capaz de encontrar um par (m, s) tal que $Ver_{pk}(m, s) = True$ e m não foi assinada anteriormente pelo usuário alvo.

- **Falsificação Existencial Forte** (sEUF): o adversário é capaz de encontrar um par (m, s) tal que $Ver_{pk}(m, s) = True$ e, em todas as vezes em que o adversário consultou m ao oráculo de assinatura, em nenhuma destas vezes ele recebeu s como resposta.

O modelo de segurança tradicionalmente aceito para esquemas de assinatura de chave pública é composto pelo ataque mais forte dentre os apresentados acima, o ataque de mensagem escolhida (CMA), e o tipo de comprometimento mais fraco (mas que ainda possa causar algum impacto), a falsificação existencial (fraca ou forte). Segue abaixo este modelo:

Definição 11 ***(Inforjabilidade Existencial (Forte) Contra Ataques de Mensagem Escolhida - (s)EUF-CMA)***
Sejam $\pi = (Gen, Sign, Ver)$ um esquema de assinatura, n um parâmetro de segurança e $\mathcal{A}$ um adversário de π de tempo polinomial em k. Seja (s)EUF-CMA$(\pi, \mathcal{A}, n)$ o seguinte jogo de segurança:

1. *$(pk, sk) := Gen(1^n)$.*
2. *Fase de Falsificação: $(m, \sigma) := \mathcal{A}^{O_{Sign_{sk}}}(n, pk)$. Seja Q o conjunto de pares (mensagem, resposta) que $\mathcal{A}$ consultou ao oráculo $O_{Sign_{sk}}$, e Q_1 o conjunto de mensagens em Q.*
3. *A saída do experimento é 1 se, e somente se, $Ver_{pk}(m, \sigma) = 1$ e:*
   Se a falsificação é existencial, exige-se $m \notin Q_1$.
   Se a falsificação é existencial forte, exige-se $(m, s) \notin Q$.

Obs: $O_{Sign_{sk}}$ é o oráculo de assinatura (do usuário alvo), o qual é disponibilizado ao adversário.

O esquema π tem inforjabilidade existencial (forte) sob ataques adaptativos de mensagem escolhida *se, para todos os adversários $\mathcal{A}$ de tempo polinomial, $|Pr[$(s)EUF-CMA$(\pi, \mathcal{A}, k) = 1] - 1/2|$ é desprezível.*

Uma importante propriedade de muitos esquemas de assinatura de chave pública, que também está presente em muitos esquemas de cifrassinatura, é a seguinte:

Definição 12 ***(Irretratabilidade** (non-repudiation)). Um esquema de assinatura de chave pública $\mathcal{S}$ é dito possuir* irretratabilidade *se, para quaisquer entidades A, B, C: se a entidade A produziu uma assinatura s sobre uma mensagem m com a sua chave privada sk_A utilizando o esquema de assinatura $\mathcal{S}$, então a segunda entidade (B) consegue provar para a terceira entidade (C), que A, e somente A, foi capaz de produzir a assinatura s sobre a mensagem m. Em outras palavras, A nunca consegue provar o contrário, ou seja, não consegue* negar *que assinou a mensagem.*

Lema 1 *(**Caracterização de esquemas de assinatura de chave pública com irretratabilidade**). Um esquema de assinatura de chave pública provê* irretratabilidade *se, e somente se, assinaturas produzidas neste esquema são inforjáveis e a verificação de assinaturas pode ser feita publicamente (usualmente, usando somente a chave pública do assinador).*

Uma observação importante é a de que esquemas de autenticação de chave privada (MACs) não podem prover irretratabilidade, pois a chave privada de um usuário é compartilhada com um (ou mais) usuários.

Esta característica pode ou não ser necessária em aplicações (alguns protocolos podem até exigir o contrário), mas normalmente é, pois permite a responsabilização do usuário que assinou.

3.3 Níveis de Segurança de Girault

Esquemas de chave pública, sejam eles sob o modelo tradicional PKC, IB-PKC, CL-PKC ou CB-PKC, necessitam de uma entidade não usuária do sistema, uma terceira parte confiável (TTP) ou autoridade confiável, na qual os usuários do sistema devem confiar (com um certo nível de confiança).

A segurança de muitos esquemas de chave pública dependem, em parte, de como esta autoridade se comporta. Para melhor analisar o nível de confiança que os usuários de um dado esquema, sob um dado paradigma, devem depositar na autoridade confiável, Girault [Gir91] propôs os seguintes níveis de segurança (chamados de níveis de segurança de Girault):

- **Nível 1**: a autoridade conhece, ou é capaz de calcular facilmente, a chave privada dos usuários. Portanto, ela pode personificar qualquer usuário do sistema. Supõe também que as ações da autoridade não são detectáveis.
- **Nível 2**: a autoridade desconhece, ou não é capaz de calcular facilmente, a chave privada dos usuários. Supõe que ela é capaz de gerar chaves públicas falsas, porém válidas, para os usuários. Portanto, ela pode personificar qualquer usuário do sistema. Supõe também que as ações da autoridade não são detectáveis.
- **Nível 3**: a autoridade desconhece, ou não é capaz de calcular facilmente, a chave privada dos usuários. Supõe que as ações da autoridade são detectáveis.

Como podemos ver nesta definição, quanto maior o nível, menor a confiança que deve ser depositada na autoridade confiável.

No modelo de certificação tradicional (PKI), a autoridade confiável alcança nível 3. Os modelos alternativos (IB-PKC, CL-PKC e CB-PKC) em geral atingem níveis mais baixos e, consequentemente, a autoridade é mais poderosa e os usuários precisam ter maior confiança nela.

3.4 O Modelo do Oráculo Aleatório

O *modelo do oráculo aleatório* (ROM) é um modelo idealizado para a demonstração de esquemas criptográficos, que supõe a existência de *oráculos aleatórios* (funções aleatórias) no lugar de funções de hash (funções determinísticas).

Este modelo foi informalmente introduzido por Fiat e Shamir [FS87], e posteriomente formalizado por Bellare e Rogaway [BR93] para criptografia em geral e, em particular, para encriptação e assinatura de chave pública.

Uma das motivações para a criação deste modelo é o fato de que, quando o artigo [BR93] foi proposto, muitos esquemas criptográficos eficientes usados na prática não continham demonstrações de segurança, pois os pesquisadores não conseguiam obter demonstrações (principalmente contra adversários adaptativos) para estes esquemas, apesar de não serem conhecidas vulnerabilidades sobre eles.

Por outro lado, os esquemas criptográficos que até então possuiam demonstrações formais de segurança eram ineficientes em comparação com os esquemas anteriores, e portanto eram raramente utilizados na prática.

Dado este contexto, o objetivo da metodologia do oráculo aleatório foi prover um modelo de segurança mais fraco do que o modelo padrão, porém seguro o bastante para que demonstrações de esquemas sob este modelo proporcionassem confiança sobre a segurança dos esquemas no mundo real. Um modelo no qual esquemas eficientes já existentes pudessem ser demonstrados seguros, e também no qual uma metodologia pudesse ser desenvolvida para auxiliar o projeto de novos esquemas seguros. Uma demonstração de segurança no modelo ROM para um esquema criptográfico pode ser vista como um *meio-termo* entre uma demonstração no modelo padrão e nenhuma demonstração.

Definição de Oráculo Aleatório

Um *oráculo aleatório* é uma função aleatória $H : \{0,1\}^n \rightarrow \{0,1\}^m$, n e m inteiros positivos e $m < n$, que pode ser avaliada somente através de consultas. Ou seja, o usuário do oráculo envia uma consulta $x \in \{0,1\}^n$ ao oráculo e recebe de volta $H(x)$. O funcionamento interno da função H é desconhecido pelo usuário. Do ponto de vista deste, a função é uma caixa preta. Tudo o que este sabe sobre a função vem da sequência $(x_1,y_1 = H(x_1))$, ..., $(x_l,y_l = H(x_l))$ de pares (consulta, resposta) obtidos das consultas

feitas ao oráculo até o momento.

Esquemas criptográficos nos quais são utilizadas funções que seguem a definição acima de oráculo aleatório são ditos *esquemas criptográficos sob o modelo do oráculo aleatório.*

Metodologia

O modelo do oráculo aleatório pode ser utilizado no projeto de esquemas criptográficos a partir da seguinte metodologia:

1. Propõe-se um esquema criptográfico π.
2. Substitui-se cada função de hash utilizada em π por um oráculo aleatório. Chame de π^O o esquema resultante.
3. Demonstra-se que π^O é seguro no modelo ROM.
4. Substitui-se cada um dos oráculos aleatórios em π^O por uma função hash criptográfica real (p. ex. SHA-256). Chame de π' o esquema resultante.
5. Obtém-se uma demonstração de segurança de π' sob o modelo padrão.

Seguindo esta metodologia, se o esquema π^O foi demonstrado seguro no modelo ROM (Etapa 3), mas ainda não foi obtida demonstração de segurança de π' no modelo padrão (Etapa 5), ainda assim é garantido que o esquema não contém falhas estruturais (exceto falhas que se baseiam em fraquezas na estrutura da função de hash). Ou seja, todo o ataque que venha a ser encontrado sob o esquema π' deve fazer uso de propriedades específicas da função (ou funções) de hash criptográfica real utilizada no esquema.

Se um esquema é demonstrado seguro no ROM, uma consequência disto é que, se existe um ataque contra o esquema quando instanciado com uma função de hash concreta H, podemos substituir H por uma outra função de hash H' que não tenha as fraquezas de H que levaram ao ataque, e então o esquema resultante será resistente a este ataque específico.

Portanto, esquemas demonstrados seguros somente no modelo ROM são considerados "sound", pois, apesar de ainda não haverem demonstrações no modelo padrão para tais esquemas, a demonstração obtida é um forte indício de que o esquema é de fato seguro. Apesar de diversas críticas contra o modelo ROM, este modelo é muito utilizado para a obtenção de demonstrações de segurança para novos esquemas criptográficos, e tem ampla aceitação pela comunidade criptográfica.

Para mais detalhes sobre o modelo do oráculo aleatório e suas limitações, referimos o leitor ao trabalhos [BR93], [KL08] e [CDD07].

3.5 Segurança Demonstrável

O conceito de *segurança demonstrável* (*provable security*) surgiu na série de trabalhos de Bellare e Rogaway [BDJR97, BGR95, BKR94], nos quais também foi introduzido e amadurecido o chamado *modelo padrão*.

Segundo Bellare [Bel98], o projeto de *primitivas atômicas* seguras, por exemplo as *funções one-way* como a função RSA e as cifras de bloco como o DES, pode ser visto mais como uma arte do que uma ciência. Mas, segundo o autor, o projeto de protocolos criptográficos, os quais utilizam as primitivas atômicas para prover funções de segurança úteis (p. ex: confidencialidade, autenticidade, integridade) pode ser transformado em uma ciência, se este for baseado em noções formais de segurança e na noção de reduções da Teoria da Complexidade.

O autor provê argumentos para isto, baseando-se nos ataques apresentados contra esquemas criptográficos na história da criptografia moderna, os quais, em praticamente a sua totalidade, ocorrem sobre os protocolos, e não sobre as primitivas atômicas. Ou seja, estes ataques ocorrem no nível de protocolo, no sentido de que os ataques apresentados normalmente não dependem das características de uma primitiva específica.

Partindo deste fato, a idéia do autor é de que, supondo que as primitivas atômicas utilizadas no protocolo em questão são seguras, poderíamos, utilizando as noções formais de segurança adequadas para o tipo de protocolo e os princípios da Teoria da Complexidade, reduzir a segurança do protocolo à segurança da primitiva subjacente, de modo que a *única* forma de quebrar o protocolo seja através da quebra da primitiva subjacente.

O *modelo padrão* consiste na construção de uma redução entre um problema considerado computacionalmente difícil e um ataque ao esquema (protocolo) em questão. Tais reduções seguem os princípios da Teoria da Complexidade, portanto são aceitas somente reduções polinomiais: os tempos de execução envolvidos devem ser polinomiais no parâmetro de segurança, dentre eles o tempo de execução do adversário e a complexidade de tempo da redução. Além disso, a probabilidade de sucesso do adversário, também deve ser *desprezível* (veja a definição de Função Desprezível em 3.1.3). Esquemas que possuem reduções deste tipo são ditos *demonstrados seguros*. O conceito de segurança demonstrável encontra-se dentro da classe segurança computacional, pois a segurança dos esquemas sob este modelo dependem da dificuldade de um problema considerado computacionalmente difícil.

A grosso modo, podemos interpretar a redução da seguinte maneira. Seja A o problema computacional e B o problema de quebrar o esquema, a partir do momento que uma redução polinomial de A para B é produzida, obtemos a implicação: "Se B pode ser quebrado em tempo polinomial t (sob a noção de segurança apropriada) com probabilidade ϵ, então A pode ser resolvido em tempo polinomial t' com probabilidade ϵ'". Da lei

contrapositiva, temos que a implicação anterior é logicamente equivalente a "Se A não pode ser resolvido em tempo polinomial t' com probabilidade ϵ', então B não pode ser quebrado em tempo polinomial t com probabilidade ϵ".

Tomando a hipótese da dificuldade do problema computacional (A não pode ser resolvido em tempo polinomial), também chamada de *hipótese computacional*, e a última implicação (obtida da lei contrapositiva), podemos concluir que B não pode ser quebrado em tempo polinomial, concluindo a demonstração de segurança do esquema.

Como a demonstração de segurança obtida através da técnica de segurança demonstrável se baseia na dificuldade de quebra da primitiva subjacente, ou seja, a demonstração não garante segurança absoluta, o uso do termo "demonstração" para se referir a este tipo de argumentação é considerado um exagero para alguns autores, pois este termo tem um significado consagrado e forte em Matemática que não se adequa bem ao seu uso neste contexto. Segundo o próprio Bellare, um dos criadores do conceito de segurança demonstrável, um termo mais adequado seria *segurança reducionista* [Bel98]. Menezes e Koblitz preferem o termo *argumento reducionista* [KM07].

Segundo [KM06], a idéia de Segurança Demonstrável tem sido utilizada também como metodologia de projeto de protocolos criptográficos. Tradicionalmente, criptógrafos obtinham um esquema que *acreditavam* ser seguro, e somente então tentavam demonstrar a segurança do protocolo. Com o advento da Segurança Demonstrável, muitos criptógrafos passaram a iniciar o projeto de protocolos com um objetivo de segurança reducionista (uma conjectura) que eles gostariam de demonstrar. Então, no caminho para a demonstração, eles adicionam um componente após o outro ao protocolo tal que a demonstração prossiga corretamente, até que, finalmente, a demonstração seja atingida, e a conjectura torna-se portanto um teorema. Em outras palavras, o objetivo de segurança reducionista "guia" a construção do protocolo.

3.5.1 Segurança Exata, Segurança Prática e Eficiência da Redução

Segundo Bellare [Bel98], apesar do potencial de *Segurança Demonstrável* em impactar o projeto de protocolos criptográficos práticos seja muito grande, o seu uso na prática até aquele momento havia sido muito pequeno.

Uma primeira razão para isto seria o fato de os protocolos demonstrados seguros propostos até então não utilizarem cifras de bloco como primitivas atômicas, as quais eram muito populares e já possuíam implementações muito eficientes na época. Ao invés disto, estes protocolos utilizavam primitivas baseadas em problemas difíceis em Teoria dos Números, que até então não possuíam implementações eficientes ou estas ainda não estavam sendo amplamente utilizadas.

Uma segunda razão é o fato de que os protocolos demonstrados seguros eram signi-

ficativamente mais ineficientes em comparação com os protocolos utilizados na prática, devido ao uso de primitivas menos eficientes, como explicado anteriormente.

Uma terceira razão encontra-se na determinação de valores para os parâmetros do procolo, quando este é instanciado. A demonstração de segurança do protocolo provê uma expressão que descreve um limitante inferior para a complexidade de tempo ($t(k)$) e a probabilidade ($\epsilon(k)$) de um adversário quebrar o esquema, em função da complexidade de tempo ($t'(k)$) e a probabilidade ($\epsilon'(k)$) do adversário "quebrar" a primitiva subjacente, onde k é o parâmetro de segurança. Normalmente são duas expressões, $t(k) \geq T(t'(k))$ e $\epsilon(k) \leq E(\epsilon'(k))$, onde T e E são funções que representam a complexidade das respectivas transformações. A complexidade da redução é dada pelo par de transformações (T, E). Pelo fato de se basear em reduções da Teoria de Complexidade, é somente exigido que a redução tenha complexidade polinomial no parâmetro de segurança, ou seja, que T e E sejam polinomiais em k.

Implementadores de protocolos criptográficos precisam de números precisos sobre a segurança de um protocolo em função do valor do parâmetro de segurança, de modo a determinar valores que serão empregados em diversos componentes do esquema a ser instanciado, por exemplo, o tamanho em bits das chaves. Porém, o uso de *notação assintótica* na descrição de polinômios e polinômios inversos (nas probabilidades), que por definição é uma notação frouxa (*loose*) para funções em geral, torna difícil obter valores justos (*tight*) para os parâmetros do protocolo.

Esta dificuldade em estimar "bons" valores a partir da redução obtida na demonstração fez com que os implementadores muitas vezes deixassem até mesmo de ler as demonstrações de segurança dos protocolos propostos, pois, mesmo se estes entendessem as demonstrações (as quais usualmente são difíceis de entender para pesquisadores fora da área), ainda assim não conseguiriam utilizá-las para instanciar o protocolo.

Nas demonstrações baseadas em Teoria da Complexidade, também denominadas *demonstrações reducionistas*, a complexidade de tempo da redução é normalmente apresentada em notação assintótica, e é somente requerido que esta seja polinomial. Porém, em criptografia a eficiência da redução é muito importante pois, como veremos a seguir, uma redução polinomial não é suficiente para garantir a segurança se o polinômio tem grau (ou constantes) muito grande.

A redução provê um algoritmo para resolver o problema computacional (problema A) a partir de um algoritmo para quebrar o esquema criptográfico (problema B). Seja Alg_A^{red} um algoritmo para A obtido deste modo. Suponha que a complexidade de tempo de Alg_A^{red} é $T_{Alg_A^{red}}(|params_A|)$, onde $params_A$ são os parâmetros do problema A, e $|params_A|$ é o tamanho total destes parâmetros. Suponha também que o algoritmo mais eficiente (Alg_A^*) conhecido para o problema computacional A tem complexidade $T_{Alg_A^*}(|params_A|)$.

A redução do problema B para o problema A normalmente é trivial. Seja Alg_B^{red} um

algoritmo para B obtido deste modo, e $T_{Alg_B}(|params_B|)$ a complexidade de tempo deste algoritmo.

Como argumentado por Pointcheval [Poi05], nos primeiros esquemas criptográficos demonstrados seguros no modelo padrão, e para parâmetros de segurança suficientemente pequenos, a redução polinomial obtida indica que todos os adversários contra o esquema levarão pelo menos vários anos para quebrá-lo (com probabilidade não desprezível). Porém na prática pode existir um ataque contra o esquema que utiliza o melhor algoritmo para o problema computacional (A), e que pode quebrar o esquema (com probabilidade não desprezível) dentro de algumas horas.

Quanto maior o valor do parâmetro de segurança, mais próximo ficará o tempo previsto pela redução e aquele obtida pelo melhor algoritmo para o problema computacional A, portanto, a redução provê uma estimativa precisa para valores muito grandes do parâmetro de segurança.

Ainda segundo Pointcheval [Poi05], para que esquemas com demonstração de segurança sejam *de fato* seguros para valores pequenos do parâmetro de segurança (valores úteis na prática), é necessário que as reduções obtidas nas demonstrações sejam eficientes (justas).

Portanto, para evitar grandes disparidades entre a segurança "teórica"obtida através de reduções, e a segurança "na prática", é muito importante obter reduções eficientes. De preferência o mais eficiente possível, de modo que a diferença entre a complexidade do melhor algoritmo para o problema computacional e a complexidade do algoritmo para a quebra do esquema (este último é obtido a partir da redução) seja pequena. Ou, equivalentemente, a redução deve ser tal que quebrar o esquema dentro de um certo intervalo de tempo implica na capacidade de resolver o problema difícil dentro de um intervalo de tempo *bastante próximo* ao intervalo anterior.

Se estas diferenças forem pequenas para todos os tamanhos de instâncias possíveis do problema e, consequentemente, para todo valor do parâmetro de segurança (já que o tamanho das instâncias é derivado deste valor), melhor será.

Na prática, no entanto, é suficiente que esta diferença seja pequena apenas para instâncias pequenas do problema, pois estas são as instâncias que serão efetivamente utilizados nos esquemas instanciados. Podemos ver isto da seguinte maneira: das instâncias do problema computacional corresponderá os valores atribuídos aos parâmetros do esquema (quando este for instanciado) e, para o esquema ser utilizado na prática, ele deve ser eficiente, o que significa que os valores atribuídos aos parâmetros do esquema (e consequentemente o tamanho da instância do problema computacional subjacente) devem ter os menores tamanhos possíveis para o nível de segurança pretendido.

As afirmações dos parágrafos anteriores foram feitas sobre a complexidade de tempo, mas, como os algoritmos tratados são probabilísticos, estas afirmações devem valer também

para a probabilidade de sucesso dos algoritmos. Em particular, a probabilidade de resolver o problema computacional também deve ser *bastante próxima* da probabilidade de quebrar o esquema.

Esquemas demonstrados seguros e cuja redução é eficiente, no sentido dos parágrafos anteriores, atingem a assim chamada *segurança prática* [Poi05]. Na busca por reduções mais eficientes, foram definidos outros conceitos semelhantes ao conceito de segurança prática, denominados *segurança concreta* [OO98] e *segurança exata* [BR96].

Com o passar do tempo, com objetivo de aumentar a eficiência de seus esquemas, os projetistas de esquemas no modelo padrão passaram a utilizar hipóteses computacionais mais fortes, e portanto menos seguras, normalmente baseadas em problemas computacionais menos estudados. Do outro lado, projetistas de esquemas no modelo ROM procuraram aumentar a segurança dos seus esquemas através do uso de hipóteses computacionais mais fracas [Poi05].

Apesar destes esforços, esquemas no modelo padrão com hipóteses computacionais fortes ainda eram ineficientes para uso prático [Poi05]. Segundo [Poi05], em 1998, o esquema mais eficiente para cada uma das categorias de esquemas criptográficos tinha somente demonstração no modelo ROM.

Tanto quanto saibamos, atualmente os esquemas no modelo ROM são os mais eficientes no contexto de assinatura, e são a maioria entre os mais eficientes no contexto de encriptação [Abd11]:

- No contexto de esquemas PKE baseados na dificuldade de problemas da família Diffie-Hellman, DHIES [BR97, MR99, ABR01] (também conhecido como DHAES) está entre os mais rápidos, e se baseia na dificuldade do Gap-CDH no modelo ROM. Uma variante do DHIES proposta em [CKS08] é baseada na dificuldade do CDH no modelo ROM, uma hipótese mais fraca do que o Gap-CDH.

- No contexto de esquemas PKE baseados na dificuldade do problema RSA no modelo ROM, o RSA-OAEP [BR95] ainda está entre os mais eficientes.

- No contexto de esquemas PKE baseados na dificuldade da fatoração com demonstração no modelo padrão, o esquema Hofheinz-Kiltz [HK09] é o mais promissor, utilizando somente 2 exponenciações para encriptação e 1 exponenciação para decriptação.

- No contexto de esquemas PKS, esquemas com demonstração no modelo ROM, como por exemplo FDH [BR93] e PSS [BR96], ainda são os mais eficientes.

3.5.2 Um Outro Olhar sobre Segurança Demonstrável

Ataques não cobertos pelos modelos de segurança

A técnica de Segurança Demonstrável não garante segurança contra ataques que estão fora do modelo de segurança adotado. Em particular, em quase todos os modelos de segurança de esquemas criptográficos na literatura, ataques por canais secundários (*side-channel attacks*) não são considerados, portanto demonstrações de esquemas nestes modelos não garantem segurança contra ataques deste tipo.

Atualmente existem algumas técnicas para a implementação de primitivas criptográficas ou operações criptográficas básicas que garantem, com demonstração formal de segurança, a resistência dos algoritmos resultantes a certos tipos de ataques por canais secundários. Trabalhos nesta direção incluem: exponenciação [CJ01], AES [BGK05], [RDP08].

Tentativas de ataque ao modelo ROM

Diversos trabalhos tentaram apontar fraquezas no modelo ROM, argumentando que este não seria um modelo adequado para demonstrações de segurança [KM07].

Dentre estes trabalhos, Canneti et al [CGH04] apresentaram exemplos de esquemas criptográficos demonstrados seguros no ROM, mas que são inseguros quando instanciados com qualquer função hash. Bellare et al [BBP04] apresentaram um esquema de encriptação híbrido demonstrado seguro no ROM, mas que é inseguro quando instanciado com qualquer função hash. Goldwasser e Kalai [GK03] apresentaram uma classe de esquemas de assinatura demonstrados seguros no ROM, mas inseguros quando instanciados com qualquer função hash. Halevi e Krawczyk [HK07] mostra um esquema de encriptação que é seguro no ROM, mas inseguro no modelo padrão.

Em todos os trabalhos anteriores, os autores se esforçaram para construir exemplos de esquemas criptográficos que *separassem* o modelo ROM do modelo padrão, porém, o máximo que obtiveram foram esquemas criptográficos artificiais, distantes da realidade, que de fato separam o modelo ROM do modelo padrão.

Segundo [KM07], o fato dos pesquisadores não conseguirem mostrar a separação entre o modelo ROM e o modelo padrão utilizando esquemas criptográficos realistas, apesar de muito esforço, reforça a confiança no modelo ROM. Um problema em aberto, portanto, é construir um esquema criptográfico realista que separe o modelo ROM do modelo padrão.

Resultados contrários para um mesmo problema

Boneh e Venkatesan [BV98] apresentaram um resultado de que é improvável que exista uma redução eficiente da Fatoração para o RSA. O trabalho, cujo título é "Breaking RSA May Not Be Equivalent to Factoring", mostra que, para o problema RSA com expoente e

pequeno, se uma redução (algébrica) eficiente da fato existisse, Fatoração seria fácil (uma contradição, pois supomos que Fatoração é difícil).

Segundo Koblitz e Menezes [KM06], o resultado dos autores podem ter outra interpretação, contrária a anterior: dado que os autores se esforçaram para obter a inequivalência para o caso geral, mas somente conseguiram mostrar a inequivalência para expoentes pequenos, enquanto o problema RSA geral (com expoentes e arbitrários) tem sido estudado a muito tempo e não há outro método conhecido para resolvê-lo além da Fatoração, o resultado obtido pelos autores apenas reforça o fato de que o problema RSA geral é muito provavelmente equivalente à Fatoração.

O resultado de Brown [Bro05b], cujo título é "Breaking RSA May Be As Difficult as Factoring" mostra, através de um argumento reducionista, um resultado contrário aquele de [BV98], também considerando o problema RSA com expoentes públicos pequenos.

Em esquemas de assinatura do tipo ElGamal: (1) deve-se realizar o padding aleatório a mensagem e então aplicar o hash ($H(m||r)$), ou (2) deve-se aplicar o hash a mensagem e depois aplicar o padding aleatório ($H(m)||r$)?. Por exemplo, esquemas de assinatura baseados em assinaturas Schnorr usam (1), enquanto os esquemas DSA e ECDSA usam (2). Na literatura foram apresentados argumentos reducionistas [PS96, PS00, KM07] (redução de DLP para falsificação de assinatura Schnorr) que mostram que (1) é segura, e também argumentos reducionistas [Bro05b, Bro05a] (redução do problema do semi-logaritmo adaptativo para ataque ataques CCA de forja universal sobre ECDSA) que mostram o contrário. Mais uma vez, o uso de argumentos reducionistas proporcionou resultados contrários para o mesmo problema.

A partir dos exemplos anteriores, Koblitz e Menezes [KM06] concluem: a existência de resultados, obtidos através de argumentos reducionistas, contrários para um mesmo problema mostra o valor limitado dos argumentos reducionistas.

Reduções justas são necessárias?

Definição 13 *(**Redução Justa (Tight) (Definição Informal)**) Seja Alg_A um algoritmo probabilístico para resolver o problema A que leva no máximo tempo T e tem probabilidade pelo menos ϵ de sucesso (ou seja, é bem sucedido para um proporção pelo menos ϵ de execuções), onde T e ϵ são funções do parâmetro de segurança k. Uma redução de um problema B para um problema A é um algoritmo Alg_B^{red} que resolve B, o qual utiliza um algoritmo Alg_A para o problema A como subrotina. Seja t' e ϵ' respectivamente o tempo máximo e a probabilidade de sucesso do algoritmo Alg_B^{red} assim obtido, onde estes valores são funções do parâmetro de segurança.*

*Uma redução é dita **justa** (ou **eficiente**) se, e somente se, $T' \approx T$ e $\epsilon' \approx \epsilon$. Uma redução é dita **não-justa** se $T' \gg T$ ou $\epsilon' \ll \epsilon$.*

Em segurança demonstrável, as demonstrações de protocolos devem, idealmente, obter reduções justas (tight) de um problema considerado difícil (problema A) para um ataque bem sucedido ao protocolo (problema B). Isto significa que um adversário que quebra o esquema com sucesso o faz com tempo e probabilidade *aproximadamente* iguais ao tempo e probabilidade de um adversário que quebra o problema difícil subjacente. Entretanto, grande parte dos protocolos demonstrados seguros apresentados na litetura estão longe deste ideal, pois trazem reduções frouxas que necessitam de valores imensos para os parâmetros para que sejam seguras, o que as torna ineficientes para uso prático.

Koblitz e Menezes [KM06] apresentam um protocolo de identificação (artificial, não-realista) que pode ser demonstrado seguro (com redução não-justa), mas na prática é inseguro, quando instanciado com um valor realista para o parâmetro de segurança.

Bellare e Rogaway [BR93] propuseram o esquema de assinatura FDH-RSA (Full Domain Hash RSA), no qual a função de hash não é aplicada sobre bits aleatório algum: $H(m)^d$, onde m é a mensagem e d é o expoente privado. Posteriormente, os mesmos autores propuseram o esquema PSS (Probabilistic Signature Scheme) [BR96], no qual a função de hash é aplicada sobre uma string de bits aleatórios cujo tamanho k_0 é dependente do parâmetro de segurança: $H(m, r)^d$, onde $r \in \{0, 1\}^{k_0}$ é a string de bits aleatórios. Katz e Wang [KW03] apresentaram o esquema KW-RSA, no qual a função de hash é aplicada sobre um bit aleatório apenas $H(m, b)^d$, onde b é o bit aleatório).

Coron [Cor00] obteve uma redução (do problema RSA para a falsificação do esquema) mais eficiente para o FDH-RSA do que a redução obtida por Bellare e Rogaway [BR93]. Em [Cor02] Coron mostrou que a redução obtida em [Cor00] é ótima (no sentido de ser a melhor possível). A redução obtida por Coron não é justa, mas, por ser ótima, facilita muito a determinação de valores adequados para os parâmetros para instanciar o esquema [KM06]. Portanto, o esquema FDH-RSA não pode ter reduções justas.

Ambos os esquemas PSS e KW-RSA possuem demonstrações de segurança com reduções justas (do problema RSA para a falsificação do esquema), porém o KW-RSA consegue, surpreendentemente, obter uma redução justa utilizando apenas um bit aleatório na função de hash.

A diferença na segurança de um esquema que utiliza apenas um bit aleatório no hash (KW-RSA) em relação a um esquema que não utiliza nenhum bit aleatório no hash (FDH-RSA), é levantada por Koblitz e Menezes [KM06] como mais uma evidência da necessidade de se analisar o impacto da segurança obtida por demonstrações em Segurança Demonstrável na segurança prática dos esquemas criptográficos. Os autores argumentam que este resultado (KW-RSA) não traz mais segurança na prática do que o (FDH-RSA), apesar do primeiro possuir uma redução justa.

Koblitz e Menezes [KM06] deixam em aberto o seguinte problema, o qual supostamente ajudaria a esclarer a necessidade (ou não) de reduções eficientes em Segurança

Demonstrável: encontre um exemplo de um protocolo criptográfico natural e realista que tem uma demonstração de segurança reducionista *não-justa* correta e que seja inseguro quando instanciado com parâmetros de tamanho usual (prático).

Se o problema anterior for resolvido, então fica clara a necessidade de reduções justas em demonstrações de segurança. Caso contrário, se o problema anterior não puder ser resolvido após muito esforço dos pesquisadores, então reduções justas *provavelmente* não são necessárias.

3.6 Técnicas de Demonstração

3.6.1 Sequências de Jogos

A técnica de sequências de jogos aplicada a demonstrações de segurança possui duas vertentes, a primeira foi proposta por Shoup [Sho06] e consiste em um método computacional cujo objeto fundamental analisado em cada jogo, e na transição entre estes, é o *espaço de probabilidade*.

A segunda vertente é a proposta por Bellare e Rogaway [BR04, BR08], baseada em métodos formais. Neste trabalho iremos tratar exclusivamente da abordagem de Shoup, que de fato é a mais utilizada na literatura. Apresentamos sucintamente esta abordagem nos parágrafos seguintes. Para maiores detalhes, incluindo exemplos, referimos o leitor a [Sho06].

As demonstrações de segurança de esquemas tradicionais (não baseadas em sequências de jogos) consistem de um jogo entre um adversário e um desafiante, em que o segundo utiliza o primeiro para resolver um problema supostamente difícil.

Por outro lado, as demonstrações baseadas em sequências de jogos de Shoup se baseiam na idéia de que, ao invés de apenas um jogo, há uma sequência de jogos distintos, onde o primeiro jogo é o jogo real, do qual deseja-se saber a probabilidade de sucesso do adversário, e o último jogo é um jogo cuja probabilidade de sucesso do adversário é conhecida com exatidão. Esta última probabilidade tem valor exatamente igual ao valor desejável para a probabilidade do primeiro jogo, chamada *probabilidade alvo* (ϵ_T).

Cada jogo intermediário é obtido do jogo anterior por meio de uma modificação simples e facilmente verificável. Esta modificação é tal que a probabilidade de sucesso do adversário no jogo anterior é *muito próxima* (*desprezivelmente próxima*) da probabilidade de sucesso do adversário neste jogo.

A metodologia geral de demonstrações baseadas em sequências de jogos é a seguinte:

1. Contrói-se uma sequência de jogos: $Jogo_0$, ..., $Jogo_n$, onde $Jogo_0$ é o jogo original aplicado ao esquema criptográfico original e com o adversário original, e, para todo

i, $0 < i \leq n$, o $Jogo_{i+1}$ é obtido a partir do jogo $Jogo_i$ por meio de uma *pequena* transformação. Obs: As possíveis tranformações e o significado de *pequena* será apresentado mais a frente.

2. Para cada $Jogo_i$, chame de S_i o evento de segurança *natural* para este jogo. Para cada $i > 0$, S_i é a adaptação *natural* do evento S do $Jogo_0$ para o contexto do $Jogo_i$.

3. Mostre que $Pr[S_i]$ é desprezivelmente próxima (ou igual) a $Pr[S_{i+1}]$, para todo $i = 0, ..., n-1$ e que $Pr[S_n]$ é desprezivelmente próxima (ou igual) a ϵ_T.

4. A partir da demonstração no passo anterior e do fato de que n é constante (ou polinomial no parâmetro de segurança k), temos que $Pr[S]$ é desprezivelmente próxima a ϵ_T, concluindo a demonstração.

Os possíveis tipos de transição entre os jogos são os seguintes:

- **Indistinguibilidade**. Neste tipo de transição é feita uma pequena alteração no jogo corrente ($Jogo_i$), a qual a princípio modifica a probabilidade de sucesso do adversário, para se obter o jogo seguinte ($Jogo_{i+1}$). Esta alteração é tal que, se fosse detectada pelo adversário, isto implicaria em um método para distinguir entre duas distribuições de probabilidade, $Pr[S_i]$ e $Pr[S_{i+1}]$, que por hipótese são computacionalmente indistinguíveis.

  Sejam P_1 e P_2 duas distribuições de probabilidade computacionalmente indistinguíveis. Para provar que $Pr[S_i] - Pr[S_{i+1}]$ é desprezível, constrói-se um *algoritmo de distinção distinguishing algorithm* D que *interpola* o $Jogo_i$ e o $Jogo_{i+1}$. Isto é, quando D recebe como entrada um elemento da distribuição P_1, ele devolve 1 com probabilidade $Pr[S_i]$, e quando D recebe como entrada um elemento de P_2, ele devolve 1 com probabilidade $Pr[S_{i+1}]$. A partir da existência deste algoritmo e da hipótese da indistinguibilidade de P_1 e P_2, concluimos que $|Pr[S_i] - Pr[S_{i+1}]$ é desprezível.

- **Eventos de Falha**. Neste tipo de transição é mostrado que o $Jogo_i$ e $Jogo_{i+1}$ procedem identicamente, a menos que um certo evento F, chamado *evento de falha* ocorra. Ou seja, $S_i \cap \overline{F} = S_{i+1} \cap \overline{F}$.

  Para que a argumentação sobre a transição seja clara, [Sho06] recomenda que os jogos sejam definidos sobre o mesmo espaço de probabilidade, tal que a única diferença entre eles sejam as regras para computar certas variáveis aleatórias.

  Na demonstração de que a transição por evento de falha praticamente não altera a probabilide de sucesso do adversário, ou seja, de que $Pr[S_i]$ é desprezivelmente próxima a $Pr[S_{i+1}]$, é usualmente utilizado o Lema 2. Se tomarmos $A := S_i$ e

$B := S_{i+1}$ neste lema, para mostrar que $|Pr[S_i] - Pr[S_{i+1}]|$ é desprezível, basta mostrar que $Pr[F]$ é desprezível.

- **Passos de Ligação** (*Bridging steps*). Este tipo de transição simplesmente define um modo diferente de computar certas quantidades, mas este novo modo é equivalente ao modo original. Ou seja, esta mudança é puramente conceitual, não afeta a probabilidade do evento de interesse: $Pr[S_i] = Pr[S_{i+1}]$. Esta transição é feita simplesmente para tornar mais clara a argumentação sobre a transição seguinte.

Lema 2 *(**Lema da Diferença**). Sejam A, B e F eventos definidos em alguma distribuição de probabilidade, e suponha que $A \cap \overline{F} = B \cap \overline{F}$. Então $|Pr[A] - Pr[B]| \leq Pr[F]$.*

Prova:

$$\begin{aligned} |Pr[A] - Pr[B]| &= |Pr[A \cap F] + Pr[A \cap \overline{F}] - Pr[B \cap F] - Pr[B \cap \overline{F}]| \\ &= |Pr[A \cap F] - Pr[B \cap F]| \\ &\leq Pr[F]. \end{aligned}$$

3.7 Emparelhamentos Bilineares

Sejam q um número primo, $\mathbb{G}_1$ e $\mathbb{G}_2$ grupos cíclicos aditivos de ordem prima q e $\mathbb{G}_T$ é um grupo multiplicativo cíclico de ordem prima q. Os grupos $\mathbb{G}_1$, $\mathbb{G}_2$ e $\mathbb{G}_T$ tem elementos neutros $0_{\mathbb{G}_1}$, $0_{\mathbb{G}_2}$ e $1_{\mathbb{G}_T}$, respectivamente. Um emparelhamento bilinear admissível é um mapa $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, com as seguintes propriedades:

1. (bilinearidade): dados $Q, L \in \mathbb{G}_1$ e $Z, R \in \mathbb{G}_2$, temos
   $e(Q, Z + R) = e(Q, Z)\ e(Q, R)$ e
   $e(Q + L, Z) = e(Q, Z)\ e(L, Z)$.
   Portanto, para todo $a, b \in \mathbb{Z}_q$:
   $e(aQ, bR) = e(Q, R)^{ab} = e(abQ, R) = e(Q, abR), etc.$

2. (não-degeneração)
   Para todo gerador P de $\mathbb{G}_1$ e todo gerador Q de $\mathbb{G}_2$, $e(P, Q) \neq 1_{G_T}$;

3. (eficiência) O mapa e é eficientemente computável.

Na prática, G_1 e G_2 são grupos de pontos em uma curva elíptica sobre um corpo finito $\mathbb{F}$, e G_T é um subgrupo do grupo multiplicativo de um corpo finito relacionado à $\mathbb{F}$. Dentre os emparelhamentos mais comuns estão o de Weil e o de Tate.

3.7.1 Considerações sobre Instanciação

Segundo [GPS08], diferentes instanciações de emparelhamentos produzem emparelhamentos com características de desempenho e funcionalidade distintas. Se $\mathbb{G}_1 = \mathbb{G}_2$, o emparelhamento é dito *simétrico*, ou Tipo 1, e é implementado usando curvas elípticas supersingulares. Quando $\mathbb{G}_1 \neq \mathbb{G}_2$, o emparelhamento é dito assimétrico, e há dois tipos: (Tipo 2) emparelhamentos onde há um homomorfismo eficiente $\varphi : \mathbb{G}_2 \to \mathbb{G}_1$ e (Tipo 3) emparelhamentos onde não há tal homomorfismo.

3.8 Problemas Difíceis e Hipóteses Criptográficas

Definição 14 *Discrete Logarithm Problem (DLP)*
Dados $(\mathbb{G}, +)$ um grupo aditivo cíclico de ordem prima q, P um gerador de $\mathbb{G}$e $Q \in \mathbb{G}$. Determine $a \in \mathbb{N}$ tal que $Q = aP$.

Definição 15 *Diffie-Hellman Problem (DHP) ou Computational Diffie-Hellman Problem (CDHP)*
Sejam $(\mathbb{G}, +)$ um grupo aditivo cíclico de ordem prima q, P um gerador de $\mathbb{G}$, e $a, b \in \mathbb{Z}_q$. O problema DHP em $\mathbb{G}$é: dados (P, aP, bP), determine abP.

Definição 16 *Decision Diffie-Hellman Problem (DDHP)*
Sejam $(\mathbb{G}, +)$ um grupo aditivo cíclico de ordem prima q, P um gerador aleatório de $\mathbb{G}$e a, b, c elementos aleatórios de $\mathbb{Z}_q$. O problema DDHP em $\mathbb{G}$ é: dados (P, aP, bP, cP), determinar se $cP \stackrel{?}{=} abP$. Ou, equivalentemente, dados P, aP, bP, cP, distinguir entre as distribuições $< P, aP, bP, abP >$ e $< P, aP, bP, cP >$, onde $a, b, c \in \mathbb{Z}_q$.

A partir destas definições, podemos afirmar que o DDHP não é mais difícil do que o DHP, e também que o DHP não é mais difícil do que o DLP, pois existem reduções polinomiais do DDHP para o DHP e do DHP para o DLP. No trabalho [JN01], os autores argumentam que, em grupos gerais, a afirmação anterior é tudo o que podemos dizer sobre a relação entre as dificuldades destes problemas.

Em [MW99] são apresentados argumentos fortes (não provas) de que o DLP e o DHP são equivalentes em tempo polinomial, o que leva a crer que se o DLP é forte em um dado grupo, podemos assumir que, muito provavelmente, o DHP também é forte neste grupo.

A relação entre a dificuldade do DHP e o DDHP parece ser mais fraca, em [JN01] são construídos grupos de pontos em curvas elípticas onde o DDHP é fácil e o DHP é considerado equivalente ao DLP, mostrando uma separação entre ambos os problemas.

Definição 17 *Bilinear Computational Diffie-Hellman Problem (BCDHP)*
Sejam $\mathbb{G}_1$ e $\mathbb{G}_2$ grupos de ordem prima q, P um gerador de $\mathbb{G}_1$ e $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_2$

um emparelhamento admissível. O BCDHP em $(\mathbb{G}_1, \mathbb{G}_2, e)$ *é: dados* (P, aP, bP, cP)*, para* $a, b, c \in \mathbb{Z}_q^*$*, computar* $e(P,P)^{abc} \in \mathbb{G}_2$.

Definição 18 *Bilinear Decision Diffie-Hellman Problem (BDDHP)*
Sejam $\mathbb{G}_1$ *e* $\mathbb{G}_2$ *grupos de ordem prima* q*,* P *um gerador de* $\mathbb{G}_1$ *e* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_2$ *um emparelhamento admissível. O BDDHP em* $(\mathbb{G}_1, \mathbb{G}_2, e)$ *é: dados* (P, aP, bP, cP, T)*, para* $a, b, c \in \mathbb{Z}_q^*$*, determinar se* $T = e(P,P)^{abc} \in \mathbb{G}_2$.

Definição 19 *Generalized Bilinear Computational Diffie-Hellman Problem (GBCDHP)* [1]
Seja $\mathbb{G}_1$ *um grupo aditivo e* $\mathbb{G}_2$ *um grupo multiplicativo, ambos de ordem prima* q*,* P *um gerador de* $\mathbb{G}_1$ *e* $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ *um emparelhamento bilinear.*
O GBDHP em $(\mathbb{G}_1, \mathbb{G}_2, e)$ *é definido por: dados* (P, aP, bP, cP)*, onde* $a, b, c \in_R \mathbb{Z}_q^*$*, devolver um par* (Q, R)*, onde* $Q \in \mathbb{G}_1^*$ *e* $R = e(P,Q)^{abc} \in \mathbb{G}_2$.

3.8.1 A Classe de Problemas Lacunares

Os problemas *lacunares* (Gap) surgiram em [OP01]. Segundo os autores, alguns esquemas criptográficos importantes não continham demonstrações de segurança (até a publicação do artigo), também não haviam vulnerabilidades conhecidas, e também acreditava-se que estes não poderiam ser demonstrados seguros com as hipóteses criptográficas tradicionais baseadas na dificuldade de problemas tradicionais, dentre outros, da fatoração e do logaritmo discreto. Os autores então consideraram que seriam necessárias algumas hipóteses de segurança mais fortes para que estes esquemas pudessem ser demonstrados seguros.

Intuitivamente, um problema lacunar significa resolver um problema de inversão com a ajuda de um oráculo para o problema de decisão relacionado. Por exemplo, seja P um problema e seja f uma relação entre uma instância x para P e a solução correspondente y, ou seja, $f(x,y) = 1$ se, e somente se, y é solução de x segundo o problema P. Então, o problema lacunar de P é: dado a instância x, encontre y satisfazendo $f(x,y) = 1$ (y é solução de x), com a ajuda de um oráculo que responde se $f(x', y') \stackrel{?}{=} 1$ para uma consulta (x', y') qualquer.

Seguem abaixo os problemas Diffie-Hellman lacunares que serão utilizados nesta dissertação:

Definição 20 *Gap (Computational) Diffie-Hellman Problem (G-DHP ou G-CDHP)*
Seja $\mathbb{G}$*um grupo aditivo cíclico de ordem prima* q*, e* P *um gerador de* $\mathbb{G}$*. O problema G-CDHP em* G *é: dados* (P, aP, bP)*, para* $a, b \in Z_q^*$*, determinar* abP*, tendo a ajuda de um oráculo de decisão que, para uma consulta* (P, aP, bP, C)*, responde com* 1 *se* $C = abP$*, e com* 0*, caso contrário.*

[1] também conhecido como Generalized Bilinear Diffie-Hellman Problem

Definição 21 *Gap Computational Diffie-Hellman Problem With Bilinear Decisional Diffie-Hellman Oracle (G-CDHP-BDDH)*
Sejam G_1 e G_T grupos de ordem prima p, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ um emparelhamento, e P um gerador de G_1. O problema G-CDHP-DBDH é definido por: dados (P, aP, bP), para $a, b \in Z_q^*$, determinar abP, tendo a ajuda de um oráculo que resolve o problema Bilinear Decision Diffie-Hellman (BDDHP).

Definição 22 *Gap Bilinear (Computational) Diffie-Hellman Problem (G-BDHP ou G-BCDHP)*
Sejam $\mathbb{G}_1$ e $\mathbb{G}_2$ grupos de ordem prima q, P um gerador de $\mathbb{G}_1$ e $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_2$ um emparelhamento admissível. O BDHP em $(\mathbb{G}_1, \mathbb{G}_2, e)$ é: dados (P, aP, bP, cP), para $a, b, c \in \mathbb{Z}_q^*$, computar $e(P, P)^{abc} \in \mathbb{G}_2$, tendo a ajuda de um oráculo de decisão que responde $C \stackrel{?}{=} e(P, P)^{abc}$, para uma consulta (P, aP, bP, C).

Capítulo 4

Criptografia de Chave Pública Sem Certificados

4.1 Introdução

No modelos de segurança para os diversos tipos de esquemas sem certificados (CLE, CLS, CLSC, CLKA, etc), são considerados dois tipos de adversário:

- **Tipo I**: é um usuário típico do sistema. Não tem acesso à chave secreta mestra (msk), ou seja, não tem o poder da KGC. Dado que no paradigma CL-PKC não há certificação explícita, assume-se que este tipo de adversário pode substituir a chave pública de qualquer usuário do sistema (inclusive de si mesmo) [1].
- **Tipo II**: é a KGC. Tem acesso à chave secreta mestra (msk). Supomos que este tipo de adversário não substitui a chave pública dos usuários do sistema.

4.2 Encriptação Sem Certificados (CLE)

Apresentamos na Figura 4.1 o modelo original para CLE [ARP03], composto por sete algoritmos.

Obviamente, um esquema CLE deve ser consistente. Isto é, para toda as escolhas possíveis de $params$, para toda entidade A (identidade ID_A, par de chaves legítimo $((S_A, P_A))$, e para toda mensagem m, temos: $Decripta(params, Encripta(params, m, ID_A, P_A), S_A) = m$.

[1] Na prática, a substituição da chave pública pode ocorrer de diversas maneiras, dentre elas: a substituição da chave de um usuário em um ataque de man-in-the-middle quando a chave pública deste for utilizada por outro usuário, ou a substituição da chave pública de um usuário que está presente em um diretório com as chaves públicas de todos os usuários.

1. **Inicializa** (KGC)
   `Entrada`: 1^k, para um parâmetro de segurança k.
   `Saída`: parâmetros públicos *params* e chave mestra *msk*.

2. **Extrai-Chave-Privada-Parcial** (KGC)
   `Entrada`: *params*, chave mestra msk e identificador $ID_A \in \{0,1\}^*$ de uma entidade A.
   `Saída`: chave privada parcial D_A. Envia D_A para A através de um canal confidencial e autêntico.

3. **Escolhe-Valor-Secreto** (A)
   `Entrada`: *params*, identificador ID_A da entidade A.
   `Saída`: valor secreto x_A.

4. **Define-Chave-Privada** (A)
   `Entrada`: *params*, chave privada parcial D_A e valor secreto x_A.
   `Saída`: chave privada completa S_A.

5. **Define-Chave-Pública** (A)
   `Entrada`: *params*, valor secreto x_A.
   `Saída`: chave pública P_A.

6. **Encripta** (B)
   `Entrada`: *params*, mensagem m, identificador ID_A, chave pública P_A.
   `Saída`: texto cifrado c ou o símbolo $\perp$ indicando falha.

7. **Decripta** (A)
   `Entrada`: *params*, texto cifrado c, chave privada S_A.
   `Saída`: mensagem m ou o símbolo $\perp$ indicando falha.

Figura 4.1: Esquema Genérico para Encriptação Sem Certificados

Podemos ver que o algoritmo *Define-Chave-Pública* pode ser executado antes ou depois de *Define-Chave-Privada*, pois ele depende apenas do valor secreto (além de *params*), o qual é obtido de *Escolhe-Valor-Secreto*. Esta é a propriedade de CLE que permite as aplicações do tipo *workflow criptográfico*.

Os algoritmos *Escolhe-Valor-Secreto*, *Define-Chave-Privada* e *Define-Chave-Pública* podem ser executados em sequência, depois que a chave privada parcial D_A tiver sido recebida da *CA*. Além disto, a maioria dos esquemas CLE (e também alguns esquemas de outros tipos) utilizam como informação privada do usuário o valor secreto e a chave privada parcial (separadamente), ao invés de utilizar a chave privada completa (S_A). A partir destas observações, podemos juntar estes três algoritmos em apenas um, *Define-Chaves*, obtendo assim um esquema genérico para CLE com apenas cinco algoritmos. Esta definição equivalente de CLE foi proposta por Hu et al [HWZD06], e pode ser vista na Figura 4.2.

1. **Inicializa** (KGC)
   `Entrada`: 1^k, para um parâmetro de segurança k.
   `Saída`: parâmetros públicos *params* e chave mestra *msk*.

2. **Extrai-Chave-Privada-Parcial** (KGC)
   `Entrada`: *params*, chave mestra *msk* e identificador $ID_A \in \{0,1\}^*$ de uma entidade A.
   `Saída`: chave privada parcial D_A. Envia D_A para A através de um canal confidencial e autêntico.

3. **Define-Chaves** (A)
   `Entrada`: *params*, identificador ID_A e chave privada parcial D_A.
   `Saída`: valor secreto x_A e chave pública P_A.

4. **Encripta** (B)
   `Entrada`: *params*, mensagem m, identificador ID_A, chave pública P_A de A.
   `Saída`: texto cifrado c ou o símbolo $\perp$ indicando falha.

5. **Decripta** (A)
   `Entrada`: *params*, texto cifrado c, chave privada S_A.
   `Saída`: mensagem m ou o símbolo $\perp$ indicando falha.

Figura 4.2: Esquema Genérico Simplificado para Encriptação Sem Certificados

4.2.1 Modelo de Segurança para CLE

A noção padrão para a segurança de esquemas de encriptação de chave pública se refere à indistinguibilidade de textos cifrados contra adversários adaptativos que realizam ataques de texto cifrado escolhido (IND-CCA2). São apresentados mais detalhes sobre esta noção na seção 3.

O modelo IND-CCA2 tradicional foi modificado em [BF01] para atender aos requisitos de segurança de IBE. Dentre as alterações estão o tratamento de adversários que podem extrair chaves privadas de entidades arbitrárias (exceto da entidade de desafio, ID_{CH}) e a escolha pelo adversário da identidade alvo do desafio (ID_{CH}).

O trabalho [ARP03] estende o modelo de [BF01] para a configuração de CLE, permitindo que os adversários extraiam chaves privadas parciais ou chaves privadas (ou ambas) de identidades escolhidas por estes. Além disso, dado que em CL-PKC não há certificados, o modelo obtido pelos autores também considera adversários que podem substituir a chave pública de qualquer entidade por qualquer valor.

Apresentamos na Figura 4.3 as consultas (oráculos) que estão disponíveis ao adversário ($\mathcal{A}$) e são simuladas pelo desafiante ($\mathcal{C}$). [2]

O desafiante deve responder a consulta *Decripta* mesmo se a chave pública de A foi substituída. No modelo de segurança aqui discutido [ARP03], os autores consideram duas opções: $\mathcal{C}$ escolhe um valor secreto aleatório para decifrar (neste caso, a decifragem obtida provavelmente estará incorreta, devido ao valor secreto aleatório), ou $\mathcal{C}$ decifra C corretamente, de algum modo.

A primeira opção é razoável: como o desafiante conseguirá decifrar um texto cifrado por uma chave pública escolhida livremente pelo adversário? A segunda opção está mais distante da realidade, em termos de segurança, mas é factível: em alguns esquemas, o desafiante consegue construir as respostas às consultas do jogo de indistinguibilidade de forma a ser capaz de decifrar até mesmo textos cifrados por chaves públicas escolhidas pelo adversário.

A segunda opção é, obviamente, a mais forte em relação a segurança, e é a opção escolhida pelos autores. A demonstração do esquema CLE concreto de AlRiyami e Paterson [ARP03] (Seção 4.2.3) utiliza esta segunda hipótese. Oráculos *Decripta* do primeiro tipo são chamados de *oráculos de decriptação fortes*, enquanto os outros são chamados de *oráculos de decriptação fracos*.

Demonstrações de segurança que empregam oráculos fortes de encriptação (assinatura) utilizam um mecanismo específico ao esquema em questão chamado *extrator de conhecimento* (*Knowledge Extractor*) [ARP03] que permite a decriptação (assinatura)

[2] Neste trabalho as consultas à oráculos são representada por Nome-da-Consulta (Entidade-Remetente, Parâmetro1, Parâmetro2...), onde Entidade-Remetente é a entidade que submeteu a consulta.

- **Extrai-Chave-Privada-Parcial** (A)
  $\mathcal{C}$ executa `Extrai-Chave-Privada-Parcial` para gerar a chave privada parcial D_A para a entidade A, se esta ainda não foi gerada. Caso contrário, apenas devolve D_A.

- **Extrai-Chave-Privada** (A)
  Se a chave pública de A não foi substituída, então $\mathcal{C}$ executa `Escolhe-Valor-Secreto` (se não tiver sido executado anteriormente) e então usa o valor secreto obtido para executar `Define-Chave-Privada`. Finalmente, devolve a chave obtida.
  Se a chave pública de A foi substituída, então $\mathcal{C}$ pára a simulação e devolve *FALHA*.

- **Requisita-Chave-Pública** (A)
  Se esta foi a primeira requisição pela chave pública de A, então $\mathcal{C}$ executa `Define-Chave-Pública` (após executar `Escolhe-Valor-Privado`) para obter a chave pública P_A de A. Senão, devolve a chave pública obtida na primeira consulta.

- **Substitui-Chave-Pública** (A, P_A)
  Substitui a chave pública corrente P_A da entidade A por P'_A.

- **Decripta** (A, C)
  Se a chave pública de A não foi substituída por $\mathcal{A}$, o desafiante $\mathcal{C}$ executa o algoritmo `Decripta` com o texto cifrado C e a chave privada S_A (executando `Define-Chave-Privada` antes, se necessário).

Figura 4.3: Oráculos Disponíveis em CLE

de uma mensagem onde a chave pública do usuário foi substituída, e o valor privado correspondente não foi fornecido.

Seguem abaixo os tipos de adversário e as restrições a que estão sujeitos:

Adversário CLE do Tipo I ($\mathcal{A}_I$): este adversário não tem acesso à chave mestra msk. Ele pode realizar todas as consultas acima, inclusive substituição de chaves públicas, sujeito às seguintes restrições:

- $\mathcal{A}_I$ não pode extrair a chave privada de ID_{CH} em momento algum.

- $\mathcal{A}_I$ não pode extrair a chave privada de uma entidade cuja chave pública foi substituída.

- $\mathcal{A}_I$ não pode fazer estas duas consultas em um mesmo jogo (pode fazer uma ou outra, mas não ambas): substituir a chave pública da identidade de desafio ID_{CH} antes da fase de desafio e extrair a chave privada parcial de ID_{CH} em alguma fase.

- Na fase 2, $\mathcal{A}_I$ não pode fazer uma consulta `Decripta` para o texto cifrado de desafio C^* e a identidade ID_{CH}, se a chave pública corrente de ID_{CH} (P_{CH}) é a mesma chave pública utilizada por $\mathcal{C}$ para encriptar M_b e obter C^*.

Adversário CLE do Tipo II ($\mathcal{A}_{II}$): este adversário tem acesso à chave mestra msk, mas não pode substituir chaves públicas dos usuários. Dado que $\mathcal{A}_{II}$ tem acesso à msk, então este é capaz de computar sozinho as chaves privadas parciais dos usuários e portanto a consulta `Extrai-Chave-Privada-Parcial` não é necessária. As restrições de $\mathcal{A}_{II}$ são as seguintes:

- $\mathcal{A}_{II}$ não pode substituir chaves públicas de quaisquer entidades, em qualquer momento.

- $\mathcal{A}_{II}$ não pode extrair a chave privada de ID_{CH}, em qualquer momento.

- Na fase 2, $\mathcal{A}_{II}$ não pode fazer uma consulta `Decripta` para o texto cifrado de desafio C^* e a identidade ID_{CH}, se a chave pública corrente de ID_{CH} (P_{CH}) é a mesma chave pública utilizada por $\mathcal{C}$ para encriptar M_b e obter C^*.

Segurança de texto cifrado escolhido (IND-CCA) [3] para CLE: um esquema CLE é semanticamente seguro contra ataques adaptativos de texto cifrado escolhido (IND-CCA seguro) se nenhum adversário $\mathcal{A}$ polinomialmente limitado do tipo I ou tipo II tem vantagem não-desprezível contra o desafiante $\mathcal{C}$ no seguinte jogo:

- **Inicializa**:
  $\mathcal{C}$ recebe 1^k, para um parâmetro de segurança k e executa o algoritmo `Inicializa` com argumento 1^k, obtendo os parâmetros do sistema *params* e *msk*. Se $\mathcal{A}$ é do Tipo I então $\mathcal{C}$ envia *params* para $\mathcal{A}$. Se $\mathcal{A}$ é do Tipo II então $\mathcal{C}$ envia *params* e *msk* para $\mathcal{A}$.

- **Fase 1**:
  $\mathcal{A}$ faz uma sequência de consultas a $\mathcal{C}$, respeitando as restrições. As consultas são respondidas de forma adaptativa, ou seja, $\mathcal{A}$ recebe a resposta da última consulta antes de fazer a consulta seguinte.

- **Fase de Desafio**:
  Eventualmente, $\mathcal{A}$ decide que já fez o número de consultas suficiente na Fase 1, e devolve uma identidade de desafio ID_{CH} e duas mensagens M_0 e M_1 de mesmo comprimento.

[3] Esta é de fato a noção de segurança IND-CCA2, mas preferimos manter o termo IND-CCA, o qual é o termo adotado pelo artigo original.

Supomos que o adversário respeita todas as restrições, e portanto a identidade ID_{CH} escolhida por $\mathcal{A}$ deve respeitar a restrição acima de que a chave privada correspondente não tenha sido extraída na Fase 1.
A seguir, $\mathcal{C}$ escolhe um bit aleatório $b \in \{0,1\}$, calcula $C^* :=$ `Encripta`(*params*, M_b, P_{CH}, ID_{CH}), onde P_{CH} é a chave pública corrente da entidade ID_{CH}, e envia C^* para $\mathcal{A}$.

- **Fase 2**
  Tal como na Fase 1, $\mathcal{A}$ faz uma sequência de consultas à $\mathcal{C}$ de forma adaptativa, respeitando as restrições.

- **Chute** [4]
  Finalmente, $\mathcal{A}$ devolve um bit $b' \in \{0,1\}$. $\mathcal{A}$ vence o jogo se $b = b'$. A vantagem de $\mathcal{A}$ no jogo é $\text{Adv}(\mathcal{A}) := 2(Pr[b = b'] - \frac{1}{2})$

O modelo de segurança para CLE que incorpora ataques de KGC maliciosa [AMC+07] (Seção 4.6) tem como principais diferenças em relação ao modelo de segurança para CLE acima [ARP03]:

- Se o adversário em questão é do tipo II, a fase `Inicializa` no jogo de segurança acima passa a ser: $\mathcal{C}$ executa $\mathcal{A}$ (1^k, *master-key-gen*), onde *master-key-gen* é uma flag que diz a $\mathcal{A}$ que ele deve gerar os parâmetros do sistema. $\mathcal{A}$ devolve a chave pública mestra *mpk*.

- Novamente supondo que o adversário é do tipo II, na *Fase 1* (*Fase 2*), $\mathcal{C}$ executa $\mathcal{A}$ passando o valor [5] *choose1* (*choose2*) como parâmetro, informando ao adversário a fase em que este se encontra.

Um outra diferença do modelo [AMC+07] para CLE, que não é detalhada aqui, é o novo oráculo `CreateUser` (não existente no modelo tradicional), o qual foi criado pelos autores para tornar explícito o poder da KGC em controlar a geração das chaves privadas parciais dos usuários (*ataques de KGC maliciosa na geração de chaves privadas parciais*, Seção 4.6).

4.2.2 O Modelo Baek-Safavi-Susilo [BSNS05]

O modelo CLE de Al-Riyami e Paterson [ARP03], daqui em diante chamado modelo CLE Al-Riyami-Paterson, apresenta o problema de negação de decriptação (*denial of decryption*) [LAS07]. Este problema tem origem no fato de que a chave pública de um

[4] do inglês Guess. Pode ser traduzido também como "adivinhação".
[5] do inglês tag.

usuário receptor (B) não pode ser explicitamente autenticada por um usuário emissor (A), e consiste no seguinte: se um adversário $\mathcal{A}$ produzir uma chave pública falsa P' para B, e A utilizar tal chave para encriptar uma mensagem m para B, obtendo o texto cifrado c, então B não conseguirá decriptar c (nem mesmo $\mathcal{A}$ será capaz de decriptar c). Apesar deste ataque comprometer somente a disponibilidade, não havendo comprometimento do sigilo, ainda assim é um ataque bastante significativo contra o modelo CLE.

Na direção de resolver o problema de negação de decriptação em CLE, foi proposto o modelo CLE de Baek-Safavi-Susilo [BSNS05]. A diferença entre este modelo e o modelo Al-Riyami-Paterson está na exigência da obtenção da chave privada parcial como requisito para o cálculo da chave pública deste. Para mais detalhes sobre este modelo, referimos o leitor a [BSNS05, Den08].

4.2.3 Um Esquema Concreto

No trabalho de Al-Riyami e Paterson [ARP03] são propostos dois esquemas para CLE: BasicCL-PKE e FullCL-PKE. O esquema BasicCL-PKE é baseado no esquema IBE BasicIdent de [BF01] e não atinge o nível de segurança IND-CCA, servindo apenas como introdução para o esquema seguinte. O esquema FullCL-PKE também se baseia no esquema IBE FullIdent de [BF01] e é demonstrado IND-CCA seguro no modelo do oráculo aleatório sob a hipótese da dificuldade do GBDHP.

Apresentamos o esquema FullCL-PKE na Figuras 4.4 e 4.5, o problema GBDHP está definido na Seção 3.8. A chave privada parcial D_A computada no passo 2b é exatamente a chave privada do IBE BasicIdent. Esta é uma característica comum em adaptações de esquemas baseados em identidade para o paradigma sem certificados.

Podemos ver com facilidade que o esquema FullCL-PKE é consistente, ou seja, que o algoritmo `Decripta` desfaz o que o algoritmo `Encripta` faz. Se $C = (U, V, W)$ é uma encriptação de m usando ID_A e P_A, então o algoritmo `Decripta` calcula

$$\begin{aligned}
m' &= W \oplus H_4(\sigma') \\
&= m \oplus H_4(\sigma) \oplus H_4(V \oplus H_2(e(S_A, U))) \\
&= m \oplus H_4(\sigma) \oplus H_4(\sigma \oplus H_2(e(Q_A, Y_A)^r) \oplus H_2(e(x_A s Q_A, rP))) \\
&= m \oplus H_4(\sigma) \oplus H_4(\sigma \oplus H_2(e(Q_A, Y_A)^r) \oplus H_2(e(Q_A, x_A s P)^r)) \\
&= m \oplus H_4(\sigma) \oplus H_4(\sigma \oplus H_2(e(Q_A, Y_A)^r) \oplus H_2(e(Q_A, Y_A)^r)) \\
&= m.
\end{aligned}$$

No trabalho [ARP03] foi demonstrado que se as funções de hash H_1, H_2, H_3 e H_4 se comportam como oráculos aleatórios e não há algoritmo de tempo polinomial que possa resolver o GBDHP em $(\mathbb{G}_1, \mathbb{G}_2, e)$, então o esquema FullCL-PKE é IND-CCA seguro sob

o modelo de segurança para CLE definido na Seção 4.2.1.

Em [AMC+07] foi mostrado que uma KGC maliciosa é capaz de montar um ataque contra um usuário específico do esquema FullCL-PKE, sendo capaz de decifrar todas as mensagens cifradas com a chave pública deste, pois ela consegue computar o valor secreto deste usuário logo após ele publicar a sua chave pública. Portanto, este esquema não é seguro sob o modelo mais forte de segurança para CLE que incorpora ataques de KGC maliciosa (Seção 4.2.1).

4.3 Assinatura Sem Certificados (CLS)

4.3.1 Introdução

O primeiro esquema de assinatura sem certificados foi o esquema de Al-Riyami e Paterson [ARP03], proposto no mesmo trabalho que deu origem ao paradigma sem certificados.

Apresentamos o esquema genérico para CLS na Figura 4.6. Exceto os algoritmos *Assina* e *Verifica*, os outros algoritmos são idênticos aos do esquema CLE Genérico (Figura 4.1), por isso apresentamos somente estes dois.

Mostramos nas Figuras 4.7 e 4.8 o esquema CLS concreto de Al-Riyami e Paterson [ARP03]. Este esquema possui *verificação explícita* da chave pública (Passo 2a). Tal como os esquemas de encriptação BasicCL-PKE e FullCL-PKE [ARP03] (Seção 4.2.3), ele também se baseia no esquema BasicIdent [BF01].

Este esquema não continha demonstração de segurança, e foi mostrado por Huang, Susilo e Mu [HSMZ05] que um adversário do Tipo I é capaz de forjar assinaturas, pois, dentre outras fatores, a *verificação* da chave pública no Passo 2a é insuficiente. Neste mesmo trabalho os autores propõem uma correção e demonstram que o esquema resultante é EU-CMA seguro no modelo ROM.

O esquema, tal como apresentado no artigo que o propôs [ARP03] contém um pequeno erro: é dito que H (Passo 1d) é a única função hash necessária, porém isto não é verdade, pois é necessária uma função hash para computar Q_A (Passos 2a e 2b), a qual está inclusa nesta versão que apresentamos com o nome H_1 (Passo 1c).

4.3.2 Tipos de Ataques

As noções de segurança para esquemas de assinatura de chave pública mais aceitas na literatura são aquelas que foram propostas no trabalho de GoldWasser, Micali e Rivest [GMR88]. Este trabalho enumerou os principais tipos de ataque, os diferentes níveis de "quebra" (no quesito severidade) de um esquema de assinatura, e também propôs um

1. **Inicializa**:
   `Entrada`: 1^k, para um parâmetro de segurança k.

   (a) Escolhe grupos $\mathbb{G}_1$ e $\mathbb{G}_2$ de ordem prima q e emparelhamento $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.

   (b) Escolhe um gerador $P \in \mathbb{G}_1$ e uma chave mestra $s \in_R \mathbb{Z}_q^*$. Faz $P_0 := sP$.

   (c) Escolhe o comprimento em bits das mensagens (n), em função de k.

   (d) Escolhe funções de hash criptográficas:
   $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$,
   $H_2 : \mathbb{G}_2 \to \{0,1\}^n$,
   $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$,
   $H_4 : \{0,1\}^n \to \{0,1\}^n$.

   (e) Os parâmetros do sistema são *params* $:= (\mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, H_1, H_2, H_3, H_4)$. O espaço de mensagens é $\mathcal{M} = \{0,1\}^n$ e o espaço de textos cifrados é $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^{2n}$.

2. **Extrai-Chave-Privada-Parcial**:
   `Entrada`: identificador $ID_A \in \{0,1\}^*$.

   (a) Faz $Q_A := H_1(ID_A) \in \mathbb{G}_1^*$

   (b) Devolve $D_A := sQ_A \in \mathbb{G}_1^*$.
   (Obs: a entidade A pode verificar se o valor D_A recebido da KGC está correto fazendo $e(D_A, P) \stackrel{?}{=} e(Q_A, P_0)$).

3. **Escolhe-Valor-Secreto**
   `Entrada`: identificador ID_A.

   (a) Escolhe e devolve $x_A \in_R \mathbb{Z}_q^*$.

4. **Define-Chave-Privada**
   `Entrada`: chave privada parcial D_A e valor secreto x_A.

   (a) Devolve $S_A := x_A D_A \in G_1^*$.

5. **Define-Chave-Pública**
   Entrada: valor secreto x_A.

   (a) Faz $X_A := x_A P$ e $Y_A := x_A P_0$.

   (b) Devolve $P_A := (X_A, Y_A)$.

Figura 4.4: Esquema FullCL-PKE (Parte 1)

6 **Encripta**
`Entrada`: mensagem $m \in \mathcal{M}$, identificador $ID_A \in \{0,1\}^*$ e chave pública $P_A = (X_A, Y_A)$.

(a) Verifica se $X_A, Y_A \in \mathbb{G}_1^*$ e $e(X_A, P_0) \stackrel{?}{=} e(Y_A, P)$. Se não for, devolve $\perp$.
(b) Faz $Q_A := H_1(ID) \in \mathbb{G}_1^*$.
(c) Escolhe $\sigma \in \{0,1\}^n$.
(d) Faz $r := H_3(\sigma, M)$.
(e) Faz $U := rP$, $V := \sigma \oplus H_2(e(Q_A, Y_A)^r)$, $W := M \oplus H_4(\sigma)$.
(f) Devolve $C := (U, V, W)$

7 **Decripta**
`Entrada`: texto cifrado $C = (U, V, W) \in \mathcal{C}$, chave privada S_A da entidade A.

(a) Calcula $\sigma' := V \oplus H_2(e(S_A, U))$.
(b) Calcula $m' := W \oplus H_4(\sigma')$.
(c) Faz $r' := H_3(\sigma', m')$.
(d) Verifica se $U \stackrel{?}{=} r'P$. Se não for, devolve $\perp$.
(e) Devolve m'.

Figura 4.5: Esquema FullCL-PKE (Parte 2)

1. **Assina** (Usuário A)
`Entrada`: *params*, mensagem $m \in \mathcal{M}$ e chave privada S_A.
`Saída`: assinatura $sig \in \mathcal{S}$.

2. **Verifica**
`Entrada`: *params*, mensagem $m \in \mathcal{M}$, assinatura $sig \in \mathcal{S}$, identificador ID_A e chave pública P_A.
`Saída`: *válida*, *inválida* ou $\perp$ (falha).

Figura 4.6: Esquema Genérico de Assinatura Sem Certificados

1. **Inicializa** (KGC)
   `Entrada`: 1^k, para um parâmetro de segurança k.
   (a) Escolhe um primo q, grupos $\mathbb{G}_1$ e $\mathbb{G}_2$ de ordem q e um emparelhamento $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.
   (b) Escolhe um gerador P de $\mathbb{G}_1$, uma chave mestra $s \in_R \mathbb{Z}_q^*$ e faz $P_0 := sP$.
   (c) Escolhe uma função de hash $H_1 : \{0,1\}^* \to \mathbb{G}_1$.
   (d) Escolhe uma função de hash $H : \{0,1\}^* \times \mathbb{G}_2 \to \mathbb{Z}_q^*$.
   (e) Escolhe o inteiro n para ser o comprimento em bits das mensagens.
   (f) O espaço de mensagem é $\mathcal{M} = \{0,1\}^n$, o espaço de assinaturas é $\mathcal{S} = \mathbb{G}_1 \times \mathbb{Z}_q^*$, e *params* é $(\mathbb{G}_1, \mathbb{G}_2, n, e, P, P_0, H)$

2. **Extrai-Chave-Privada-Parcial** (KGC)
   `Entrada`: ID_A.
   (a) Calcula $Q_A := H_1(ID_A) \in \mathbb{G}_1^*$.
   (b) Devolve $D_A := sQ_A \in \mathbb{G}_1^*$.
   (Obs: a entidade A pode verificar se o valor D_A recebido está correto fazendo $e(D_A, P) \stackrel{?}{=} e(Q_A, P_0)$).

3. **Escolhe-Valor-Secreto** (A)
   `Entrada`: *params*, ID_A.
   Escolhe $x_A \in_R \mathbb{Z}_q^*$ e devolve x_A.

4. **Define-Chave-Privada** (A)
   `Entrada`: *params*, chave privada parcial D_A, x_A.
   Devolve $S_A := x_A D_A$

5. **Define-Chave-Pública** (A)
   `Entrada`: *params*, x_A.
   Calcula $X_A := x_A P \in \mathbb{G}_1$ e $Y_A := x_A P_0 \in \mathbb{G}_1$.
   Devolve $P_A := (X_A, Y_A)$.

Figura 4.7: Esquema de Assinatura Sem Certificados de Al-Riyami e Paterson

1. **Assina**
   `Entrada`: *params*, mensagem m, chave privada S_A.
   (a) Escolhe $a \in_R \mathbb{Z}_q^*$. Calcula $r := e(aP, P) \in \mathbb{G}_2$.
   (b) Faz $v := H(m, r) \in \mathbb{Z}_q^*$ e $U := vS_A + aP \in \mathbb{G}_1$
   (c) Devolve $\sigma := (U, v)$.

2. **Verifica** (B)
   `Entrada`: *params*, mensagem m, $\sigma = (U, v) \in \mathbb{G}_1 \times \mathbb{Z}_q^*$, identidade ID_A da entidade A, chave pública (X_A, Y_A).
   (a) Verifica se $e(X_A, P_0) \stackrel{?}{=} e(Y_A, P)$. Se não for, devolve $\perp$.
   (b) Calcula $Q_A := H_1(ID_A) \in \mathbb{G}_1^*$ e $r := e(U, P)e(Q_A, -Y_A)^v$.
   (c) Verifica se $v \stackrel{?}{=} H(m, r)$. Se for, devolve m. Senão, devolve $\perp$

Figura 4.8: Esquema de Assinatura Sem Certificados de Al-Riyami e Paterson (Cont.)

esquema de assinatura seguro sob a noção mais forte de segurança proposta no trabalho, a segurança EU-CMA (discutida a diante).

Duas grandes classes de ataques podem ser identificadas:

- **Ataques Apenas com a Chave**: o adversário conhece apenas a chave pública do assinante.
- **Ataques de Mensagem**: o adversário pode examinar assinaturas correspondentes a mensagens conhecidas ou escolhidas por este.

Nas definições seguintes, A denota o usuário alvo do ataque. Os ataques de mensagem podem ser divididos, em ordem de severidade crescente, em:

- **Ataques de Mensagem Conhecida**: o adversário tem acesso a uma lista de mensagens (m_1, m_2,..., m_n) e as assinaturas correspondentes a estas mensagens (σ_1,σ_2,..., σ_n), mas não é capaz de escolher as mensagens desta lista.
- **Ataques Genéricos de Mensagem Escolhida**: o adversário pode escolher as mensagens que vão compor a lista de mensagens a serem assinadas, somente *antes* de ter acesso à chave pública de A. Após compor a lista mensagens, o adversário submete esta lista ao assinador, que devolve para ele a lista das assinaturas. Neste ponto é fornecido ao adversário acesso à chave pública de A, e ele pode então iniciar a tentativa de quebra do esquema.
- **Ataques Dirigidos de Mensagem Escolhida**: semelhante ao tipo de ataque anterior, porém o adversário tem acesso à chave pública de A *antes* de escolher as mensagens que vão compor a lista de mensagens a serem assinadas.
- **Ataques Adaptativos de Mensagem Escolhida**: além das capacidades dadas ao adversário pelo tipo de ataque anterior, neste tipo de ataque o adversário pode submeter as mensagens ao assinador de maneira adaptativa, ou seja, após escolher e submeter uma mensagem ao assinador, e depois de receber a assinatura correspondente, o adversário pode escolher e submeter a próxima mensagem a ser assinada. Em outras palavras, a escolha feita pelo adversário da mensagem a ser assinada pode depender do valor da assinatura de todas as mensagens anteriormente assinadas. O assinador age como um oráculo para o adversário.

Os *ataques adaptativos de mensagem escolhida* constituem o tipo mais genérico e forte de ataque à esquemas de assinatura, e também são ataques que podem ser realizados na prática. Portanto, é desejável que esquemas de assinatura seguros sejam resistentes a ataques deste tipo.

Um inimigo pode comprometer um esquema de assinatura e esta quebra pode ter um impacto maior ou menor dependendo da informação que o adversário foi capaz de obter ou o que este foi capaz de fazer. Segue abaixo em ordem decrescente de severidade os tipos de comprometimentos alcançados pelo adversário:

- **Quebra Total**: consegue computar a informação secreta de A.
- **Falsificação Universal**: consegue obter um algoritmo de assinatura funcionalmente equivalente ao algoritmo de assinatura autêntico de A, possivelmente utilizando uma informação secreta diferente daquela usada pelo algoritmo de A.
- **Falsificação Seletiva**: o adversário primeiramente escolhe uma mensagem e então falsifica uma assinatura válida para esta mensagem.
- **Falsificação Existencial**: o adversário consegue obter uma assinatura válida para pelo menos uma mensagem, mas não tem controle sob a mensagem que foi assinada. Esta mensagem é aleatória, ou não faz sentido.

O tipo de comprometimento de menor impacto é a *falsificação existencial*, portanto, para garantir que um esquema de assinatura tenha um alto nível de segurança basta garantir que este não seja *existencialmente inforjável*.

Combinando o tipo de comprometimento de menor impacto com o tipo mais forte de ataque, obtemos a *noção padrão de segurança* para esquemas de assinatura, a de um esquema *existencialmente inforjável sob ataques de mensagem escolhida adaptativos* (EU-CMA).

A noção padrão de segurança EU-CMA pode ser capturada através de um jogo de segurança entre um adversário ($\mathcal{A}$) e um desafiante ($\mathcal{C}$). Um adversário capaz de quebrar um esquema EU-CMA seguro é capaz de vencer o seguinte jogo com probabilidade não-desprezível, rodando por um tempo polinomial (no parâmetro de segurança) [Dan08]:

Definição 23 *(Jogo EU-CMA). Um esquema de assinatura $\mathcal{S}$ é existencialmente inforjável sob ataques adaptativos de mensagem escolhida (EU-CMA) se, dado um parâmetro de segurança k, qualquer adversário $\mathcal{A}$ tem chance desprezível (em k) de vencer o seguinte jogo contra o desafiante $\mathcal{C}$:*

1. *$\mathcal{C}$ gera os parâmetros do sistema (*params*).*
2. *$\mathcal{A}$ escolhe o usuário alvo $\mathcal{U}^*$.*
3. *$\mathcal{C}$ gera a chave pública PK^* de $\mathcal{U}^*$.*

4. *$\mathcal{C}$ executa $\mathcal{A}$ (*params*, PK^*).*
   $\mathcal{A}$ executa por um tempo polinomial em k, tendo acesso a um oráculo $\mathcal{O}^S$ que computa assinaturas de mensagens escolhidas por este, utilizando para isto a chave privada SK^* correspondente à chave pública PK^*.

5. *$\mathcal{A}$ devolve um par (σ, m)*

$\mathcal{A}$ vence o jogo se o algoritmo de verificação de $\mathcal{S}$ aceita σ como assinatura válida de $\mathcal{U}^*$ em m.

Uma das diferenças dos esquemas de assinatura sem certificados (CLS) frente aos esquemas de assinatura de chave pública explicitamente certificados é o fato de que neste paradigma as chaves públicas não são explicitamente certificadas. Ou seja, existe a possibilidade do adversário substituir as chaves públicas dos usuários por valores de sua escolha, e portanto, tal possibilidade deve ser incorporada ao modelo de segurança, como um oráculo disponível ao adversário.

Ao incorporarmos esta característica, surge a necessidade de dividir os adversários de CLS em dois tipos: KGC e não-KGC. Se o adversário é do **Tipo II** (KGC), então supomos que este não tem acesso ao oráculo de substituição de chave pública, pois, caso contrário, o esquema CLS (qualquer que seja ele) seria quebrado, pois a KGC poderia personificar qualquer usuário através da geração de um novo par de chaves válido para o *usuário-alvo* (pois conhece a chave privada parcial do usuário-alvo), e posterior substituição da chave pública deste pela nova chave pública computada. Por outro lado, se o adversário é do **Tipo I** (não-KGC), supomos que este tem acesso ao oráculo de substituição de chave pública.

Por causa desta divisão dos adversários em dois tipos, as demonstrações de segurança para CLS usualmente contém uma demonstração para adversários do Tipo I e outra para adversários do Tipo II.

4.3.3 Modelo Formal de Segurança

Podemos definir formalmente a segurança de esquemas CLS com dois jogos de segurança: um jogo I entre o desafiante e um adversário do Tipo I ($\mathcal{A}_I$) e um jogo II entre um desafiante e um adversário do Tipo II ($\mathcal{A}_{II}$), restringindo as consultas que o adversário é capaz de realizar, dependendo do seu tipo.

Apresentamos na definição seguinte os dois jogos, mostrando explicitamente as diferenças entre eles.

Definição 24 *(Segurança EU-CMA contra Adversários do Tipo I e do Tipo II) [Dan08]. Seja k um parâmetro de segurança e $\mathcal{A}$ um adversário do Tipo I ou do Tipo II. O jogo de segurança é:*

1. *$\mathcal{C}$ gera a chave privada* msk *da KGC e os parâmetros públicos* params *(incluindo* mpk*). Se $\mathcal{A}$ é do tipo I, faz aux :=$\perp$, senão ($\mathcal{A}$ é do tipo II), faz aux :=* `msk`*.*

2. *$\mathcal{C}$ executa $\mathcal{A}$ (1^k,*params,aux*). Durante a execução $\mathcal{A}$ tem acesso aos oráculos definidos a seguir, segundo as restrições abaixo.*

3. *$\mathcal{A}$ devolve (ID^*,m^*,σ^*), onde ID^* é a identidade do usuário-alvo escolhido por $\mathcal{A}$.*

$\mathcal{A}$ vence o jogo se ***Verifica****(*params*, ID^*, P_ID^*, m^*, σ^*) =* "aceita" *e as seguintes restrições sobre as consultas são satisfeitas:*

- *Qualquer que seja o tipo de $\mathcal{A}$, ele está sujeito a:*
  - *a consulta* ***Assina****(ID^*, m^*) não foi feita;*
- *se $\mathcal{A}$ é do tipo I, então ele está sujeito à seguinte restrição adicional:*
  - *a consulta* ***RevelaChavePrivadaParcial****(ID^*) não foi feita.*
- *se $\mathcal{A}$ é do tipo II, então ele está sujeito às seguintes restrições adicionais:*
  - *a consulta* ***SubstituiChavePública****(ID^*,.) não foi feita;*
  - *a consulta* ***RevelaValorSecreto****(ID^*) também não foi feita.*

Um esquema CLS é dito seguro contra adversários do tipo I (ou tipo II) se nenhum adversário $\mathcal{A}$ que executa por um tempo polinomial em k tem probabilidade não-desprezível (em k) de vencer o jogo acima.

Os adversários tem os seguintes oráculos a disposição:

- **RevelaChavePública** (identidade ID_A):
  Devolve a chave pública P_A do usuário A, gerando-a primeiro se necessário.
- **RevelaChavePrivadaParcial** (identidade ID_A):
  Devolve a chave privada parcial D_A do usuário A, gerando-a primeiro se necessário.
- **RevelaValorSecreto** (identidade ID_A):
  Se a chave pública do usuário A não foi substituída, devolve o valor secreto x_A correspondente à chave pública P_A. Se a chave pública de A foi substituída, e o $\mathcal{C}$ não conhece o valor secreto correspondente à chave pública corrente, devolve $\perp$.
- **SubstituiChavePública** (identidade ID_A, nova chave pública P'_A):
  O desafiante faz com que a chave pública de A passe a ser P'_A.
  O adversário pode, opcionalmente, fornecer para $\mathcal{C}$ o valor secreto x'_A correspondente à P'_A.

- **Assina** (identidade ID_A, mensagem m):
  Devolve a assinatura σ de A sobre m.

No trabalho de Al-Riyami e Paterson [ARP03], durante a discussão sob o modelo de segurança para CLE, é argumentado que o desafiante deve ser capaz de responder corretamente à consulta `Decripta`, até mesmo para usuários cuja chave pública tenha sido substituída e o valor secreto correspondente não seja conhecido (veja a Seção 4.2.1).

Transportando a definição de oráculos de decriptação fortes e fracos (4.2.1) de CLE para o contexto de CLS, obtemos dois tipos de oráculos de assinatura: *oráculos de assinatura fortes*, que são capazes de assinar mensagens para usuários cuja chave pública tenha sido modificada, e os *oráculos de assinatura fracos*, que não são capazes de fazer isto.

De agora em diante, estaremos considerando oráculos de assinatura fortes nas discussões de segurança de CLS.

O modelo de segurança para CLS que incorpora ataques de KGC maliciosa de Au et al [AMC+07] tem diferenças em relação ao modelo de segurança para CLS acima [ARP03] que são análogas às diferenças entre o modelo para CLE de Au et al [AMC+07] e o de AlRiyami e Paterson [ARP03]. Para mais detalhes, referimos o leitor para o trabalho de Au et al [AMC+07].

4.4 Cifrassinatura Sem Certificados (CLSC)

O conceito de cifrassinatura surgiu em 1997 no trabalho de Zheng [Zhe97a, Zhe97b], como uma nova primitiva criptográfica de chave pública (modelo explicitamente certificado) que provê simultaneamente três dos principais requisitos de segurança: confidencialidade, autenticidade e irretratabilidade. A idéia de cifrassinatura sem certificados (CLSC) foi introduzida por Barbosa e Farshim [BF08], os quais propuseram um modelo de segurança para este tipo de esquema e também o primeiro esquema CLSC.

No capítulo 5 apresentamos o desenvolvimento do conceito de cifrassinatura, os modelos de segurança para esta primitiva, esquemas de cifrassinatura no modelo sem certificados (CLSC) e uma revisão de esquemas CLSC na literatura.

4.5 Acordo de Chaves Sem Certificados (CLKA)

4.5.1 Introdução aos Protocolos de Estabelecimento de Chaves

Um protocolo de *estabelecimento de chave* é um protocolo criptográfico no qual duas ou mais partes obtém acesso a um segredo compartilhado. Se o número de partes for

dois, o protocolo é dito bilateral (*two-party*), se for três, o protocolo é chamado trilateral (*tripartite*), se for maior do que três, o protocolo é chamado protocolo de estabelecimento de chave de *conferência* ou *grupo* (Conference (or Group) key establishment protocol).

Os protocolos de estabelecimento de chave subdividem-se também em: *interativos*, nos quais os participantes trocam informações entre si para obter a chave de sessão, e *não-interativos*, no qual não há troca de informações, somente o iniciador envia mensagens.

Dentre os protocolos de estabelecimento de chave distinguem-se os esquemas de *distribuição de chaves de sessão* e os esquemas de *acordo de chaves*:

- Em um protocolo de distribuição de chave de sessão (SKDP), também chamado *esquema de transporte de chave de sessão*, uma TA online (ou um dos usuários) escolhe chaves de sessão e as distribui, através de um canal seguro, aos usuários do sistema. Nestes esquemas somente uma das partes está envolvida na obtenção do valor secreto.

- Em um protocolo de acordo de chave (KAP), duas (ou mais) partes estabelecem em conjunto uma chave secreta de sessão, comunicando-se através de um canal público. Neste tipo de protocolo o valor da chave é determinado como uma função das entradas providas por cada uma das partes (duas ou mais), as quais contém valores secretos gerados por estas partes.

Nesta subseção consideraremos somente protocolos KAP.

Seguem algumas definições necessárias para discussão posterior. Um rodada (execução) de um KAP é chamada de *sessão*, e o segredo acordado entre os participantes é chamado de *chave de sessão*. Os protocolos KAP geralmente assumem que os usuários possuem *chaves de longa duração* (LL-keys), as quais são segredos estáticos (não mudam durante a existência do sistema). As chaves de longa duração são normalmente pré-computadas, e depois são armazenadas em segurança. É também muito comum nestes protocolos o uso de segredos obtidos aleatoriamente durante cada sessão, por cada um dos usuários, chamadas *chaves de curto prazo* ou *chaves efêmeras* (ephemeral keys ou short-term keys).

Há dois tipos de protocolos de acordo de chaves: aqueles baseados em criptografia de chave privada, nos quais cada par de usuários compartilha uma chave de longa duração, e aqueles basedos em criptografia de chave pública, onde cada usuário tem a sua própria chave de longa duração.

Alguns protocolos KAP também assumem a existência de uma autoridade confiável (TA), a qual possui uma informação secreta, chamada *master secret key*. Assumimos que toda comunicação entre a TA e os seus usuários é realizada por meio de um canal seguro (sigiloso e autêntico), enquanto a comunicação dos usuários entre si é realizada por uma canal aberto (público).

Podemos também classificar os protocolos KAP de acordo com as garantias que estes provêem:

- *Acordo de Chave Autenticado*(AKAP): uma parte tem certeza de que somente a outra parte (ou partes) com que deseja compartilhar o segredo será capaz de obter (computar) a chave de sessão.
- *Acordo de Chave com Confirmação de Chave*(KACP): uma parte tem certeza de que a outra parte (ou partes) de fato obteve a chave de sessão, ou pelo menos tem todas as informações necessárias para obtê-la).

Se um protocolo KAP apresenta ambas as garantias acima (autenticação da outra parte e confirmação da obtenção da chave de sessão), dizemos que este é um protocolo de *acordo de chave autenticado com confirmação de chave* (AKACP).

De agora em diante, estaremos considerando somente protocolos de acordo de chave bilateral no modelo sem certificados.

4.5.2 Propriedades de Segurança Desejáveis

- *Key-compromise impersonation (KCI) security* (Segurança contra personificação a partir do comprometimento da chave, também chamada *resiliência à KCI*): mesmo que a chave de longo prazo de um usuário A tenha sido comprometida por um atacante, ainda assim este atacante será incapaz de personificar ("se passar por") um usuário qualquer (B) para A. Obviamente, de posse da LL-key de A, o atacante é capaz de personificar A para B, qualquer que seja o usuário B. Esta propriedade visa minimizar o dano causado pelo comprometimento da LL-key.

  Um exemplo de ataque onde o dano causado pode ser grande se o protocolo KAP não possui esta propriedade ocorre quando o usuário B é um banco, e o atacante (E) personifica B para A. Neste caso, E pode solicitar informações secretas de A, as quais somente o banco real (B) solicitaria, como por exemplo o número PIN de A.

  É importante notar que qualquer protocolo KAP que utilize somente segredos estáticos (individuais ou compartilhados entre os usuários), sem qualquer interação entre os usuários para o cálculo da chave de sessão, está sujeito a este tipo de ataque. Um dos requisitos básicos para se obter segurança contra este tipo de ataque é a utilização de *chaves efêmeras*.

- *Known session-specific temporary information security* (Segurança contra conhecimento de informação específica da sessão): se o atacante obtiver acesso às chaves

efêmeras de uma dada execução do protocolo, ele deve ser incapaz de computar a chave de sessão correspondente. A propriedade mais fraca, *weak known session-specific temporary information security*, assume também que a KGC não tem acesso às chaves efêmeras.

Na prática, as implementações de protocolos criptográficos normalmente envolvem a computação ou armazenamento inseguros das chaves efêmeras [MT07], portanto esta propriedade de segurança é fundamental em protocolos KAP.

- *Known session key security*: se um atacante aprende as chaves de sessão de diversas sessões, este atacante não é capaz de computar a chave de sessão de outras sessões.

- *Forward Secrecy*: se em um dado momento um atacante tem acesso às LL-keys de um ou mais usuários do sistema, mas não tem acesso às chaves efêmeras utilizadas durante a obtenção das chaves de sessão (para todas as sessões estabelecidas pelos usuários antes do incidente), então este atacante não é capaz de determinar estas chaves de sessão.

  A propriedade de forward secrecy mais forte, *perfect forward secrecy*, garante a propriedade acima mesmo se o atacante teve acesso às LL-keys de *todos* os usuários do sistema. Se a propriedade é satisfeita desde que o atacante não tenha comprometido as LL-keys de todos os usuários do sistema, então há *partial forward secrecy*.

- *Controle da Chave* (*Key control*): Os usuários do protocolo devem ter a mesma quantidade de controle sobre o valor da chave de sessão resultante. Isto é difícil de ser obtido na prática, dado que uma das partes, chamada *iniciadora*, deve iniciar o protocolo (escolhendo a sua chave efêmera para a sessão), a outra parte, depois que obtém a mensagem da primeira, tem mais informações para escolher sua chave efêmera, de modo a tentar controlar alguns dos bits da chave de sessão que será obtida. Por isso, em protocolos KAP é somente exigido que a quantidade de controle sobre a chave de sessão pelas partes seja *quase* igual.

4.5.3 Tipos de Ataques

Abaixo definimos alguns tipos de ataques relevantes a protocolos KAP. Os ataques descritos a seguir, bem como as propriedades de segurança descritas na Seção 4.5.2 são importantes para se ter uma noção das capacidades dos adversários presentes em esquemas CLKA. Porém, um esquema CLKA seguro deve ser resistente a adversários capazes de executar não somente um tipo de ataque, mas sim um subconjunto dos ataques descritos, ou até mesmo todos os ataques, possivelmente de forma intercalada ou simultaneamente.

Para ser considerado seguro sob estas condições, um protocolo CLKA deve ter sido demonstrado seguro.

- *Ataque de falsificação* (Forgery attack): o adversário procura forjar uma chave de sessão de uma sessão que possivelmente não existiu entre as partes legítimas, através da observação de mensagens no canal e do conhecimento dos parâmetros públicos do sistema.

- *Ataque de intercalação* (Interleaving attack): o adversário utiliza mensagens de sessões anteriores ou em execução do protocolo para tentar personificar algum usuário em outra sessão. Dois subtipos comuns deste ataque são:

  - *Ataque de reexecução* (Replay attack): o adversário reenvia em um nova sessão do protocolo (possivelmente com algumas modificações) mensagens de usuários em sessões anteriores, para tentar se passar por um destes usuários nesta nova seção.

  - *Ataque de reflexão* (Reflection attack): o adversário começa a coletar as mensagens enviadas por A em uma sessão π do protocolo entre A e B (a qual foi iniciada por A). Chame de $m_{A,\pi,i}$ a i-ésima mensagem enviada por A em π (a primeira mensagem é $m_{A,\pi,1}$). A seguir o adversário inicia uma sessão π' com A, enviando $m_{A,\pi,1}$ para A. Seja $m'_{A,\pi',1}$ a resposta de A a esta mensagem. Agora, em π, o adversário envia $m'_{A,\pi',1}$ para A, recebendo $m_{A,\pi,2}$ como resposta. Em seguida, em π', o adversário envia $m_{A,\pi,2}$ para A, recebendo $m'_{A,\pi',2}$ como resposta.

    O adversário continua repassando as mensagens de uma sessão para a outra deste modo, até que a chave de sessão seja estabelecida. Agindo desta maneira, o adversário, se passando por B, tenta estabelecer uma chave de sessão com A em π, ou seja, o adversário personifica B para A em π.

    No Protocolo 1 temos uma representação gráfica deste ataque, considerando apenas uma mensagem originária de A em π.

4.5.4 Modelo Formal de Segurança

Apresentamos aqui o modelo de segurança para CLKA de Lippold, Boyd e Nieto [LBG09]. Este modelo é uma versão fortalecida do modelo de Swanson [Swa08], que por sua vez é baseado no modelo de segurança Canneti-Krawczyk estendido (eCK) de LaMacchia, Lauter e Mityagin [LLM07].

O protocolo pode ser executado por quaisquer duas partes dentro de um conjunto de n entidades. Cada parte possui uma chave pública baseada em identidade, a qual é

Protocolo 1 Ataque de Reflexão

π :	1. $A \longrightarrow Adv : m_{A,\pi,i}$
	2. $Adv \longrightarrow A : m'_{A,\pi',i}$
	3. $A \longrightarrow Adv : m_{A,\pi,i+1}$
π' :	1. $Adv \longrightarrow A : m_{A,\pi,i}$
	2. $A \longrightarrow Adv : m'_{A,\pi',i}$

derivada do seu identificador. Há um centro de geração de chaves (KGC) que gera chaves privadas baseadas em identidade para as partes, e as envia através de um canal seguro. Supomos também que as partes geram seus próprios valores secretos e chaves públicas sem certificados.

O adversário tem total controle sobre a rede utilizada para a troca de mensagens. A t-ésima sessão do protocolo entre a parte i e a parte j é representada por $\pi^t_{i,j}$. Como de costume em protocolos no modelo sem certificados, permitimos ao adversário substituir as chaves públicas sem certificados utilizadas no protocolo. Além disto, assumimos que o adversário não precisa revelar a chave privada correspondente a chave pública que substitui a chave pública anterior.

Toda sessão do protocolo inicia no estado *inicial*. Se, em uma dada sessão $\pi^t_{i,j}$, ambas as partes já computaram a chave de sessão $SK^t_{i,j}$ e a aceitaram, então dizemos que a sessão passou do estado *inicial* para o estado *aceita*. Uma sessão pode terminar antes de entrar no estado *aceita*, seja porque alguma verificação feita no protocolo falhou, seja por algum outro motivo. Neste caso dizemos que a sessão passou do estado *inicial* para o estado *rejeita*. Supomos que a informação sobre o estado corrente da sessão é pública.

Para cada sessão $\pi^t_{i,j}$ atribuímos um valor chamado *ID dos parceiros* (Partner ID): $pid = (ID_i, ID_j)$. Também associamos à esta sessão o *ID da sessão*, *sid*, a sequência ordenada de todas mensagens trocadas entre i e j. Duas sessões $\pi^t_{i,j}$ e $\pi^u_{j,i}$ são ditas *emparelhadas* (*matching*), ou tem *conversações emparelhadas*, se elas tem o mesmo *pid* (e *sid*). Ou seja, duas sessões estão *emparelhadas* se as mensagens enviadas em uma sessão são exatamente as mensagens recebidas pela outra (e vice-versa).

O jogo de segurança tem duas fases. Durante a primeira fase, o adversário $\mathcal{A}$ pode realizar consultas de maneira adaptativa aos seguintes oráculos:

- **Send**($\pi^t_{i,j}$, x): se a sessão $\pi^t_{i,j}$ não existe e $x = \lambda$, então ela será criada uma sessão sobre a parte i, e esta parte será a *iniciadora* (*initiator*) do protocolo. Por outro lado, se a sessão não existe e $x \neq \lambda$, então ela será criada sobre a parte i, e esta será a parte que responderá (*responder*). Se as partes i e j ainda não foram inicializadas,

ou seja, as chaves ainda não foram geradas, elas são inicializadas neste momento. Requeremos que $i \neq j$, ou seja, uma parte não irá executar sessão do protocolo consigo mesma.

O oráculo responde com (m_R, σ), onde m_R é a mensagem de resposta da entidade i à mensagem x do adversário (ou λ, se não houver mensagem de resposta) e $\sigma \in \{inicial, aceita, rejeita\}$ é a decisão atual de i.

No caso particular de protocolos de uma rodada (*one-round*), a parte i se comporta da seguinte maneira: [6]

- Se $x = \lambda$: gera uma valor efêmero para esta sessão, calcula a mensagem de resposta m_1 e devolve $(m_1, inicial)$.
- Se $x \neq \lambda$: se i é uma parte iniciadora, ela responde com (λ, σ), onde $\sigma \in \{aceita, rejeita\}$. Pelo contrário, se i é uma parte *responder*, ela gera um valor efêmero para esta sessão e responde com (m_2, σ), onde $\sigma \in \{aceita, rejeita\}$

- **RevealMasterKey**: devolve a chave secreta mestra.
- **RevealSessionKey**($\pi_{i,j}^t$): se a sessão está no estado *aceita*, devolve a chave de sessão. Caso contrário, devolve $\perp$.
- **RevealIDSecret**(i): devolve a chave privada baseada em identidade de i.
- **RevealSecretValue**(i): devolve o valor secreto x_i de i correspondente a chave pública sem certificados P_i. Se a chave pública de i foi substituída (a consulta ReplacePublicKey(i,$\cdot$) foi realizada), devolve $\perp$.
- **ReplacePublicKey**(i, P_i'): substitui a chave pública de i por P_i'.
- **RevealEphemeralKey**($\pi_{i,j}^t$): devolve o segredo efêmero usado na sessão $\pi_{i,j}^t$, se este segredo já foi gerado. Caso contrário, devolve $\perp$.

As consultas acima podem ser divididas em três tipos: (**1**) consultas que revelam chaves privadas da parte do protocolo que é baseada em identidades (`RevealMasterKey` e `RevealIDSecret`), (**2**) consultas relativas à parte sem certificados do protocolo (`RevealSecretValue` e `ReplacePublicKey`), e (**3**) consultas que revelam informações específicas de uma dada sessão (`RevealEphemeralKey`).

[6] Adotamos a convenção em sistemas distribuídos de que uma rodada (*round*) entre um conjunto de participantes corresponde ao intervalo de tempo entre o *início* do envio de uma mensagem por cada um dos participantes (o momento em que o participante mais adiantado começa a enviar a sua mensagem deste round), e o *término* do recebimento das mensagens enviadas anteriormente por cada um dos participantes (o momento em que o participante mais atrasado recebe a sua mensagem deste round.

Dizemos que uma dada sessão está *completamente corrompida*, se esta sessão já recebeu pelo menos uma consulta de cada tipo, sobre quaisquer partes (i ou j). Uma sessão está *aberta* se a consulta `RevealSessionKey` já foi realizada sobre ela, caso contrário, a sessão está *fechada*. Eventualmente o adversário decide que já realizou um número suficiente de consultas (polinomial no parâmetro de segurança k) nesta fase, e então inicia a fase de desafio. Para tal, ele escolhe uma sessão *limpa* (do inglês *fresh*) $\pi_{i,j}^t$, chamada sessão de teste, e realiza uma consulta **Test** sobre esta sessão. Seguem abaixo as definições necessárias:

Definição 25 *Uma sessão $\pi_{i,j}^t$ está* limpa *se todas as afirmações abaixo são verdadeiras:*

1. *$\pi_{i,j}^t$ está no estado* aceita*;*
2. *$\pi_{i,j}^t$ está* fechada*;*
3. *$\pi_{i,j}^t$ não está* completamente corrompida*;*
4. *não há sessão aberta $\pi_{j,i}^u$ que tenha conversação emparelhada com $\pi_{i,j}^t$.*

Test$(\pi_{i,j}^t)$: o oráculo verifica se a sessão $\pi_{i,j}^t$ está limpa. Se não estiver, o oráculo pára a simulação. Caso contrário, ele escolhe um bit $b \in_R \{0,1\}$. Se $b = 0$, devolve a chave da sessão $\pi_{i,j}^t$. Se $b = 1$, escolhe aleatoriamente uma chave de sessão dentro do conjunto de chaves de sessão válidas, e devolve esta chave.

Após receber a resposta da consulta **Test**$(\pi_{i,j}^t)$, inicia-se a segunda fase do jogo. Nesta fase o adversário pode continuar fazendo as mesmas consultas da primeira fase, desde que a sessão escolhida para a consulta **Test**, $\pi_{i,j}^t$, se mantenha limpa.

Eventualmente o adversário $\mathcal{A}$ devolve um chute (*guess*) b'. Se $b' = b$, nós dizemos que o adversário vence. A vantagem de $\mathcal{A}$ em vencer o jogo é definida por $\text{Adv}(\mathcal{A}) := 2(Pr[b = b'] - \frac{1}{2})$.

Seguem abaixo as restrições para cada tipo de adversário:

Definição 26 ***Strong Type I Secure Key Agreement Scheme****. Um esquema de acordo de chaves é seguro contra adversários do tipo I (forte) se todo adversário probabilístico com tempo polinomial $\mathcal{A}$ tem vantagem desprezível no jogo definido acima, sujeito às seguintes restrições:*

- *$\mathcal{A}$ pode realizar no máximo dois tipos de consulta sobre cada parte envolvida na sessão de teste. Para cada tipo de consulta realizado sobre uma parte, dizemos que o $\mathcal{A}$ corrompeu um segredo desta parte. (Obs: $\mathcal{A}$ pode substituir chaves públicas de qualquer parte, porém isto conta como um segredo corrompido);*

- *$\mathcal{A}$ não pode fazer consultas **RevealSecrectValue** sobre partes que já tiveram a chave pública substituída (**ReplacePublicKey**);*
- *$\mathcal{A}$ pode fazer consultas **RevealSessionKey** até mesmo para chaves de sessão computadas por identidades sobre as quais $\mathcal{A}$ substituiu a chave pública;*
- *$\mathcal{A}$ pode substituir chaves públicas de qualquer parte após a consulta **Test** ter sido feita.*

Definição 27 ***Strong Type II Secure Key Agreement Scheme**. Um esquema de acordo de chaves é seguro contra adversários do tipo II (forte) se todo adversário probabilístico com tempo polinomial $\mathcal{A}$ tem vantagem desprezível no jogo definido acima, sujeito às seguintes restrições:*

- *$\mathcal{A}$ recebe a chave secreta mestra s no início do jogo;*
- *$\mathcal{A}$ pode realizar no máximo um tipo de consulta sobre cada parte envolvida na sessão de teste. Para cada tipo de consulta realizado sobre uma parte, dizemos que o $\mathcal{A}$ corrompeu um segredo desta parte. (Obs: $\mathcal{A}$ pode substituir chaves públicas de qualquer parte, porém isto conta como um segredo corrompido);*
- *$\mathcal{A}$ não pode fazer consultas **RevealSecrectValue** sobre partes que já tiveram a chave pública substituída (**ReplacePublicKey**);*
- *$\mathcal{A}$ pode fazer consultas **RevealSessionKey** até mesmo para chaves de sessão computadas por identidades sobre as quais $\mathcal{A}$ substituiu a chave pública;*
- *$\mathcal{A}$ pode substituir chaves públicas de qualquer parte após a consulta **Test** ter sido feita.*

4.5.5 Um Esquema CL-KAP Seguro

Apresentamos a seguir o esquema de acordo de chaves com autenticação implícita, sem certificados, e com uma rodada de Lippold, Boyd e Nieto [LBG09]. Este esquema é demonstrado seguro no modelo do oráculo aleatório sob o modelo de segurança definido na seção anterior.

Por se tratar de um esquema com somente uma rodada, a autenticação é implícita, ou seja, não há confirmação do estabelecimento de chave. Segundo Krawczyk [Kra05], a autenticação explícita em protocolos KAP é possível com três meias-rodadas (half-rounds) (cada passo de envio ou recepção de mensagem conta como uma meia-rodada). Os autores alegam que a confirmação pode ser adicionada ao protocolo da mesma maneira que o KAP HMQV [Kra05] foi transformado em HMQV-C [Kra05].

- **Inicializa**

  (Executado pela KGC)

  - escolhe grupos $\mathbb{G}$ e $\mathbb{G}_T$ de ordem prima p. Escolhe um gerador P de $\mathbb{G}$ e um emparelhamento bilinear admissível $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Escolhe um inteiro $n > 0$, o comprimento em bits da chave de sessão.
  - Escolhe $s \in_R \mathbb{Z}_p$ como chave secreta mestra e define a sua chave pública como $P_{KGC} = sP$.
  - Escolhe as funções de hash criptográficas:
    $H_1 : \{0,1\}^* \to \mathbb{G}$,
    $H_2 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{G}^8 \times \mathbb{G}^6 \to \{0,1\}^n$ e
    $H_3 : \mathbb{G} \to \mathbb{G}$.

  (Executado por cada parte U)

  - Escolhe um valor secreto $x_U :\stackrel{R}{=} \mathbb{Z}_p$ e calcula a chave pública $x_U P \in \mathbb{G}$;
  - Obtém da KGC a chave privada parcial (chave privada baseada em identidade): $D_U = (D_{U,1}, D_{U,2}) = (sH_1(ID_U), sH_3(H_1(ID_U))) \in \mathbb{G}^2$.

- **Protocolo**:

  Supomos que uma parte A está iniciando uma sessão do protocolo com uma parte B.

  (Executado por A)

  Escolhe segredo efêmero $r_A :\stackrel{R}{=} \mathbb{Z}_p$.
  Faz $R_A := r_A P$.
  Envia $E_A := (R_A, P_A)$ para B.

  (Executado por B)

  Recebe E_A.
  Escolhe segredo efêmero $r_B :\stackrel{R}{=} \mathbb{Z}_p$.
  Faz $R_B := r_B P$.
  Envia $E_B := (R_B, P_B)$ para A.
  Executa o procedimento "Derivação da Chave de Sessão" (descrito abaixo).

  (Executado por A)

Recebe E_B.
Executa o procedimento "Derivação da Chave de Sessão".

- **Derivação da Chave de Sessão**:

  (Executado pela parte A)
  (Obs: A execução pela parte B é análoga, substituindo A por B e vice-versa)

  $K_A := e(H_1(ID_B), P_{KGC})^{r_A} e(D_{A,1}, R_B)$
  $K'_A := e(H_3(H_1(ID_B)), P_{KGC})^{r_A} e(D_{A,2}, R_B)$
  $L_A := e(H_1(ID_B), P_{KGC})^{x_A} e(D_{A,1}, P_B)$
  $L'_A := e(H_3(H_1(ID_B)), P_{KGC})^{x_A} e(D_{A,2}, P_B)$
  $N_A := e(H_1(ID_B), D_{A,1})$
  $N'_A := e(H_3(H_1(ID_B)), D_{A,2})$

  $SK := H_2(A, B, E_A, E_B, r_A R_B, x_A P_B, r_A P_B, x_A R_B, K_A, K'_A, L_A, L'_A, N_A, N'_A)$

Apesar de ser demonstrado seguro e ter somente uma rodada, o custo computacional do protocolo é bastante elevado, exigindo 10 emparelhamentos e 5 exponenciações. Segundo os autores, a função de hash H_3 foi utilizada para conseguir uma demonstração utilizando somente as hipóteses da dificuldade dos problemas CDHP e BDHP (segurança contra adversário Tipo I supõe CDHP e segurança contra adversário tipo II supõe CDHP e BDHP). Se for assumido que o problema Diffie-Hellman bilinear com Gap (GBDHP) é difícil (Seção 3.8.1), então pode-se remover os termos que utilizam a função H_3, restando K_A, L_A e N_A, para um total de 5 emparelhamentos e 3 exponenciações. Segundo os autores, otimizações adicionais são possíveis para algumas aplicações.

4.6 Ataques de KGCs Maliciosas

Ataque Ativo: Substituição de Chave Pública

Os modelos de segurança para CL-PKC (p.ex. modelo de segurança para CLE, CLS, CLSC, CLKA), supõem que a KGC não substitui a chave pública de usuários, esta é a única restrição ao seu comportamento. A razão para isto é que, se a KGC substitui a chave pública de algum usuário, ela conseguirá personificá-lo. Isto é fácil de ver, pois a KGC poderia escolher uma valor secreto x^*_A, computar P^*_A usando x^*_A, computar a chave privada parcial D^*_A usando s e ID_A (observe que $D^*_A = D_A$, onde D_A é a chave privada parcial legítima de A, logo a KGC não está computando um valor novo, e finalmente substituir a chave pública legítima de A (P_A) por P^*_A. Deste modo,

ela conseguiria decifrar e assinar como se fosse A, utilizando a chave privada completa $S_A^* := DefineChavePrivada(D_A^*, x_A^*)$, ou os valores x_A^* e D_A^* individualmente.

No modelo padrão de CL-PKC, especificamente na geração de chaves, o ataque anterior pode ser detectado, pois em um dado momento haverá duas públicas chaves diferentes para um mesmo usuário, e isto não deveria acontecer. Entretanto, não é possível distinguir quem foi o perpetrador do ataque, a KGC ou o usuário A, pois ambos poderiam ter gerado o novo par de chaves (P_A^*, S_A^*), pois não há envolvimento de informação secreta da KGC (a chave privada mestra msk) na geração deste par de chaves. Dado que a KGC consegue realizar este ataque sem ser responsabilizada por isto, o modelo CL-PKC original não atinge o nível 3 de Girault (o nível em que está a PKI), ficando apenas com o nível 2.

AlRiyami e Paterson [ARP03] identificaram este problema em CL-PKC e propuseram a aplicação do *encapsulamento de chaves públicas* no processo de geração de chaves, para que o modelo de CL-PKC resultante atinja o nível 3 de Girault. A técnica de *encapsulamento de chaves públicas* consiste em vincular (*bind*) a chave pública de um usuário à sua chave privada parcial (e consequentemente à sua chave privada completa). Nesta técnica, a identidade de um usuário, do ponto de vista do esquema, passa ser a concatenação da sua chave pública (escolhida arbitrariamente pelo usuário) com a sua identidade real. Se a identidade de um usuário é ID_A, este usuário define a sua chave pública P_A (em função de x_A), e a sua nova identidade é $ID'_A := ID_A||P_A$. Esta nova identidade é então utilizada pela KGC no lugar de ID_A no algoritmo *ExtraiChavePrivadaParcial*, de modo que agora $D_A := ExtraiChavePrivadaParcial(msk, ID'_A) = ExtraiChavePrivadaParcial(msk, ID_A||P_A)$. Logo, a chave privada parcial gerada pela KGC para um dado usuário está diretamente ligada à chave pública escolhida por este usuário.

Com a aplicação desta técnica, a ataque de substituição de chave pública pela KGC ocorre da seguinte maneira: a KGC escolhe x_A^*, computa P_A^*, faz $ID_A^* := ID_A||P_A^*$, computa $D_A^* := sH_1(ID_A^*) = sH_1(ID_A||P_A^*)$, e por fim, substitui a chave pública de A por P_A^*. Observe que, com a aplicação do encapsulamento de chave pública, $D_A^* \neq D_A$, pois D_A^* depende da chave pública escolhida anteriormente, neste caso, P_A^*, a qual difere de P_A.

Se for detectada a presença de duas chaves públicas válidas para um mesmo usuário (onde *válida* significa que a chave privada completa correspondente é conhecida), digamos, P_A e P_A^*, a KGC será implicada em ter gerado uma destas duas chaves (digamos, P_A^*), pois, para haver duas chaves públicas válidas, é necessária a participação da KGC na geração da chave privada parcial correspondente à chave pública falsa (veja discussão no parágrafos anteriores). Logo, se a KGC gerou mais de uma chave privada parcial para um mesmo usuário, ela está agindo de forma maliciosa, e deve ser responsabilizada por isto. Portanto, o emprego desta técnica eleva CL-PKC ao nível 3 de Girault.

Uma desvantagem do uso da técnica de *encapsulamento de chave pública* é o fato de que agora existe dependência temporal entre as chaves pública e privada, com a segunda podendo ser calculada somente depois de a primeira ter sido escolhida. Devido a esta restrição, muitos autores consideram os esquemas que utilizam tal técnica, ou outras técnicas que impõem algum tipo de restrição temporal na geração das chaves públicas e privadas, "certificateless não-estritos" ou autocertificados (*self-certified*). Referimos o leitor a [ARP03] para maiores detalhes sobre a técnica de encapsulamento de chaves públicas. No restante desta dissertação estaremos supondo o modelo CL-PKC original sem a incorporação desta técnica.

Ataques Passivos: Geração Maliciosa dos Parâmetros do Sistema

No trabalho [AMC+07], os autores levantaram uma importante questão sobre o nível de confiança depositado na KGC em esquemas no paradigma CL-PKC. Supõe-se que a KGC sempre gera os parâmetros do sistema (públicos e privados) corretamente (seguindo estritamente a especificação do esquema). Mas, em princípio, existe a possibilidade da KGC modificar a geração dos parâmetros de forma a tentar atacar os usuários do sistema. Chamamos tais ataques de *ataques de KGC maliciosa na geração dos parâmetros do sistema*.

Estes ataques, se possíveis, poderiam ser genéricos, atingindo quaisquer usuários do sistema (Tipo A), ou poderiam ser dirigidos a usuários específicos, ou somente um usuário (Tipo B). Por exemplo, se a KGC maliciosa sabe que haverá no sistema um usuário com a identidade "Bob", então ela poderia gerar os parâmetros do sistema de forma a tentar atacar somente este usuário. Dependendo de como estes ataques são construídos, os parâmetros maliciosos gerados podem ter distribuição computacionalmente indistinguível da distribuição dos parâmetros legitimamente gerados, tornando muito difícil para os usuários do sistema detectarem que a KGC está agindo de forma maliciosa apenas observando os valores produzidos pelos algoritmos do esquema.

Conforme argumentado em AlRiyami e Paterson [ARP03], um ataque do tipo B pode ser prevenido através da técnica de *encapsulamento de chave pública*. Quando esta técnica é empregada, a KGC não é mais capaz de determinar a identidade do usuário no momento da geração dos parâmetros do sistema, pois ela não é capaz de prever o valor da chave pública que o usuário escolherá.

Apesar disto, a KGC poderia, a princípio, personificar um usuário-alvo A através da substituição da chave pública deste usuário, conforme ataque descrito na seção anterior. Neste ataque, após a substituição da chave pública legítima (P_A) de A pela chave pública falsa (P'_A), existirão duas chaves públicas válidas (pois os algoritmos que utilizam estas chaves funcionarão corretamente com qualquer uma delas), porém, a existência de mais de uma chave pública válida para um mesmo usuário indica que a KGC trapaceou, pois

somente ela conseguiria gerar a segunda chave privada parcial (correspondente a segunda chave pública). Logo, neste tipo de ataque, um juiz consegue responsabilizar a KGC, e portanto, ataque do Tipo B é prevenido por esta técnica.

Já para um ataque do tipo A, até onde sabemos, não existe uma técnica genérica que garanta que o esquema seja resistente a este tipo de ataque. Se o modelo de segurança adotado para CL-PKC supõe que a KGC pode realizar ataques do tipo A, então CL-PKC com este modelo de segurança não atinge o nível 3 de Girault, somente o nível 2.

Os modelos de segurança tradicionais de CL-PKC supõem que, se o adversário em questão é do Tipo I, o desafiante simula um oráculo que gera a chave privada parcial dos usuários. E se o adversário é do Tipo II (KGC), não é necessário um oráculo, pois o próprio adversário pode computar a chave privada parcial dos usuários de seu interesse, dado que ele sabe a chave secreta mestra.

O ponto chave é que os modelos de segurança tradicionais de CL-PKC não consideram que um adversário do Tipo II tem o controle total sobre as chaves privadas parciais dos usuários (pelo menos não explicitam esta capacidade do adversário), e portanto supõem que este gera as chaves privadas parciais corretamente. Mas, a princípio, tais adversários poderiam modificar a geração de chaves privadas parciais dos usuários de modo a tentar atacar algum usuário específico ou todos eles. Chamamos estes ataques de *ataques de KGC maliciosa na geração da chave privada parcial*. Alguns esquemas possuem validação explícita da chave privada parcial pelo usuário, por exemplo, o esquema FullCL-PKE (Figura 4.4), os quais previnem ataques deste tipo.

O trabalho de Au et al [AMC+07] propõe modelos de segurança para CLE e CLS que capturam ataques de KGC maliciosa (tanto os ataques na geração dos parâmetros do sistema quanto os ataques na geração da chave privada parcial), e também um esquema concreto seguro sob este novo modelo de segurança. Os autores também fornecem demonstrações de segurança sob este modelo de segurança mais forte para o esquema CLS genérico construído a partir de IBS e PKS de Hu et al [HWZD06] e também para o esquema CLE genérico construído a partir do IBE e PKE de Libert e Quisquater [LQ06].

Capítulo 5

Cifrassinatura Sem Certificados

5.1 Introdução

O conceito de cifrassinatura surgiu em 1997 no trabalho de Zheng [Zhe97a, Zhe97b], como uma nova primitiva criptográfica de chave pública que provê simultaneamente dois dos principais requisitos de segurança para sistemas criptográficos: confidencialidade e autenticidade. Além disto, muitos esquemas de cifrassinatura também provêem irretratabilidade (Definição 12), o qual pode ser necessário em algumas aplicações.

Segundo Zheng, uma das influências que o levou a pensar na idéia de misturar encriptação e assinatura foi o surgimento e rápida difusão da técnica de *modulação codificada* em sistemas de comunicação, a qual misturou a etapa de correção de erros e a etapa de modulação em uma só etapa, chamada modulação codificada. Esta nova etapa juntou as funções de cada uma das anteriores, a alta confiabilidade na transmissão (correção de erros), e a alta eficiência na transmissão por meio do uso de pouca largura de banda (modulação), trazendo, dentre outros benefícios, a redução no tempo de processamento.

Dentre as razões para a criação desta primitiva, encontra-se aquela que está presente no próprio título do trabalho "Digital signcryption or how to achieve cost(signature & encryption) << cost (signature) + cost (encryption)" está a possibilidade de se obter os requisitos de confidencialidade e autenticidade com custo computacional e/ou custo de comunicação menores do que aqueles que podem ser obtidos pela combinação de encriptação e assinatura (primeiro a encriptação da mensagem depois a assinatura do texto encriptado, ou vice-versa)

Zheng também argumenta que, quando é necessário proteger o sigilo de uma mensagem com a encriptação, normalmente deseja-se garantir também a autenticidade da mesma. Mas, os esquemas de encriptação e assinatura de chave pública tem um custo computacional várias vezes superior ao custo dos esquemas de chave privada correspondentes, o que torna custoso o uso da combinação simples (composição) destes esquemas

para se obter esquemas de cifrassinatura, surgindo assim uma oportunidade para combinar eficientemente estas operações e obter cifrassinaturas eficientes.

No contexto da criptografia simétrica, Jutla [Jut01] estudou a junção da encriptação de chave privada com a autenticação de chave privada, dando origem ao conceito de *encriptação autenticada* (*authenticated encryption* ou *authenticryption*).

Desde a proposta do primeiro esquema de cifrassinatura não trivial (não obtido por composição) por Zheng [Zhe97a], esquemas de cifrassinatura tiveram pouquíssimo uso em protocolos criptográficos no mundo real. Mas, recentemente, projetistas e implementadores de sistemas criptográficos começaram a usá-los, motivados também pelo recente esforço iniciado em 2007 de padronização de esquemas de cifrassinatura pela ISO [ISO08].

5.2 Modelos de Segurança

A primeira tentativa de definição de um modelo de segurança adequado para cifrassinatura surgiu no trabalho de Steinfeld e Zheng [SZ00], no qual foi proposto um modelo de segurança para cifrassinatura, e no qual os autores tentaram demonstrar a segurança de um certo esquema de cifrassinatura neste modelo. Porém os autores somente conseguiram demonstrar a inforjabilidade (autenticidade) de um esquema de cifrassinatura proposto no trabalho, faltava conseguir demonstrar a confidencialidade deste esquema no modelo de segurança recém criado.

Posteriormente, no trabalho de Baek, Steinfeld e Zheng [BSZ02], foi estabelecido um modelo de segurança forte para esquemas de cifrassinatura no cenário de múltiplos usuários (*Multi-Usuário*), chamado BSZ. Os autores deste trabalho também conseguiram obter a demonstração de segurança completa (autenticação e confidencialidade) de um esquema de cifrassinatura sob este modelo de segurança, o esquema Baek-Steinfeld-Zheng proposto no mesmo artigo.

Independentemente do trabalho anterior, An, Dodis e Rabin [ADR02] propuseram um modelo de segurança para cifrassinatura no cenário de dois usuários (*Dois-Usuários*), analisaram a segurança neste modelo dos esquemas de cifrassinatura obtidos pela composição genérica de esquemas de assinatura com esquemas de encriptação.

5.2.1 Esquema de Cifrassinatura Baek-Steinfeld-Zheng

O esquema de cifrassinatura Baek-Steinfeld-Zheng [BSZ02] foi o primeiro esquema de cifrassinatura explicitamente certificado (PKSC) demonstrado seguro, o qual foi obtido a partir de mudanças feitas no esquema de Zheng [Zhe97a], para que os autores conseguissem demonstrar a sua segurança no modelo BSZ proposto no mesmo trabalho [BSZ02]. A Figura 5.1 ilustra o esquema.

1. **Setup**
   (a) Escolhe um primo grande p e um fator primo q de $p-1$.
   (b) Escolhe um inteiro $g \in \mathbb{Z}_p^*$ com ordem q módulo p.
   (c) Escolhe um esquema de encriptação simétrico $\pi_{SE} = (E, D)$, com espaço de textos cifrados $\mathcal{C}$ e espaço de chaves $\mathcal{K}$.
   (d) Escolhe funções de hash criptográficas:
   $G : \{0,1\}^* \to \mathcal{K}$;
   $H : \{0,1\}^* \to \mathbb{Z}_q$.
   (e) Devolve $param := (p, q, g, \pi_{SE}, G, H)$.
2. **KeyGen** (ID_A)
   (a) Escolhe a chave privada $x_A \in_R \mathbb{Z}_p^*$.
   (b) Calcula a chave pública $y_A := g^{x_A} \ mod \ p$.
   (c) Devolve (x_A, y_A).
3. **Signcrypt** (m, ID_A: identidade do emissor, ID_B: identidade do destinatário)
   (a) Escolhe $x \in_R \mathbb{Z}_q^*$.
   (b) $k := y_B^x \ mod \ p$, $\tau := G(k)$.
   (c) $c := E_\tau(m)$, $r := H(m, y_A, y_B, k)$.
   (d) Se $r + x_A \equiv 0 \ (mod \ q)$, volta para o início do algoritmo.
   (e) $s := \frac{x}{r+x_A} \ mod \ q$.
   (f) Devolve $\sigma := (c, r, s)$.
4. **Unsigncrypt** (σ, ID_A: identidade do emissor, ID_B: identidade do destinatário)
   (a) $(c, r, s) := \sigma$.
   (b) $k' := (y_A g^r)^{s x_B} \ mod \ p$.
   (c) $\tau' := G(k')$.
   (d) $m' := D_{\tau'}(c)$.
   (e) Se $H(m', y_A, y_B, k') = r$, devolve m'.
   (f) Caso contrário, devolve *REJEITA*.

Figura 5.1: Esquema de Cifrassinatura Baek-Steinfeld-Zheng

5.2.2 Cenários Dois-Usuários e Multi-Usuário

No cenário *Dois-Usuários* [ADR02], supõe-se que no sistema criptográfico há somente dois usuários, digamos, A e B. Se o esquema criptográfico é baseado em chave privada (*encriptação autenticada*), então A e B compartilham a chave privada x. Por outro lado, se o esquema criptográfico é baseado em chave pública, então supomos que A possui *dois* pares de chaves: (sk_S^A, pk_S^A) para envio, e (sk_R^A, pk_R^A) para recebimento de cifrassinaturas, onde sk_S^A e sk_R^A são chaves privadas e as chaves restantes são públicas. Analogamente, B possui (sk_S^B, pk_S^B) e (sk_R^B, pk_R^B).

Esta hipótese de que cada usuário tem dois pares de chaves pode parecer um pouco *estranha*, pois a maioria dos esquemas criptográficos de chave pública assume apenas um par de chaves, mas, alguns modelos importantes de segurança para cifrassinatura de chave pública assumem dois pares. Entretanto, existem modelos de segurança que definem cifrassinatura utilizando somente um par de chaves por usuário, como veremos na Seção 5.2.3.

No cenário *Multi-Usuário* [ADR02], supõe-se que no sistema criptográfico há um número arbitrário de usuários. Portanto, consideramos somente esquemas de cifrassinatura de chave pública neste cenário. Para cada usuário A, supomos que A possui dois pares de chaves: (sk_S^A, pk_S^A) para envio, e (sk_R^A, pk_R^A) para recebimento de cifrassinaturas.

Segundo An et al [ADR02], esta separação entre os cenários é necessária para estudar o ganho de poder do adversário quando *saltamos* do cenário de chave privada para o cenário de chave pública, e também para definir com clareza o modelo de segurança.

Este ganho de poder fica evidente quando observamos que no cenário de chave privada (simétrico), cada par de usuários constitui um *sistema* no qual os dois usuários se conhecem e confiam um no outro, e portanto estão preocupados apenas com atacantes *externos* ao sistema. [1]

Por outro lado, em sistemas assimétricos, por hipótese, os usuários não se conhecem e não há confiança mútua entre eles, pois cada usuário sabe a identidade dos usuários para (dos) os quais deseja enviar (receber) mensagens cifrassinadas, mas tem acesso a chave pública destes usuários publicada em algum servidor, sem garantia de que estas são as chaves públicas legítimas destes usuários. Portanto, em sistemas assimétricos o atacante pode ser um dos usuários do sistema, e então o poder do adversário é maior neste cenário.

O trabalho de An et al [ADR02] analisou a segurança de esquemas de cifrassinatura obtidos a partir da composição genérica de encriptação e assinatura, e propôs modelos

[1] Chamamos de "sistema" criptográfico a instanciação de um esquema criptográfico. Onde "instanciação" significa atribuição específica de chaves aos usuários do sistema. Um sistema simétrico é uma instanciação de um esquema simétrico, onde o algoritmo de geração de chaves é executado somente uma vez e os dois usuários compartilham a chave gerada. Um sistema assimétrico é uma instanciação de um esquema assimétrico, onde o algoritmo de geração de chaves foi executado para cada usuário.

de segurança para este tipo de esquema no cenário Dois-Usuários e Multi-Usuário. Neste artigo também foram obtidas condições necessárias para a segurança destes esquemas, e foram determinadas certas condições em que ocorre a *amplificação* da segurança de um dos requisitos, confidencialidade ou inforjabilidade, do esquema componente (encriptação ou assinatura) para o esquema de cifrassinatura construído.

Baek, Steinfeld e Zheng [BSZ02] propuseram um modelo de segurança para cifrassinatura no cenário Multi-Usuário (modelo BSZ) mais forte do que aquele definido em [ADR02] (modelo ADR), suportando também esquemas de cifrassinatura não triviais (aqueles não obtidos a partir da composição genérica de encriptação e assinatura).

O modelo de segurança Barbosa-Farshim para cifrassinatura sem certificados (contexto Multi-Usuário) apresentado na seção seguinte é baseado no modelo de segurança [BSZ02]. Para maiores detalhes sobre os modelos BSZ e ADR e a relação entre eles, referimos o leitor a Dent e Zheng [DZ10].

5.2.3 Modelo de Segurança Barbosa-Farshim para CLSC

A idéia de cifrassinatura sem certificados (CLSC) foi introduzida por Barbosa e Farshim [BF08], os quais propuseram um modelo de segurança para este tipo de esquema e também o primeiro esquema CLSC. Este modelo para CLSC provê irretratabilidade, além de confidencialidade e autenticidade.

Antes deste modelo de cifrassinatura sem certificados, o que havia de mais próximo à cifrassinatura no paradigma sem certificados era a encriptação autenticada sem certificados (*authenticated certificateless encryption*) proposta em [CC05], a qual não garante irretratabilidade e cujo modelo de segurança é mais fraco do que o modelo de segurança Barbosa-Farshim para CLSC. Apresentamos na Figura 5.2 o esquema CLSC genérico.

O modelo de segurança Barbosa-Farshim para CLSC é um modelo Multi-Usuário que foi baseado nos modelos de segurança [Boy03, CML05] para cifrassinatura baseada em identidade (IBSC). Segundo os autores do modelo, cifrassinatura sem certificados é *intrinsecamente* multi-usuário, diferentemente de PKSC (cifrassinatura de chave pública explicitamente certificada), no qual pode ser definido também um modelo Dois-Usuários, e analogamente à IBSC, onde o modelo Multi-Usuário é necessário. Segundo eles, a razão é de que o algoritmo *Dsc* necessita da identidade do emissor (S) e da identidade do destinatário (R), e portanto o emissor deverá enviar ao receptor estas informações em conjunto com a cifrassinatura.

Seguindo o modelo de segurança para IBSC, os autores não consideram como ataques válidos aqueles em que as identidades dos usuários receptor e emissor são as mesmas ($ID_S = ID_R$). Para tal, os oráculos de cifrassinatura e decifrassinatura não permitem consultas deste tipo, e também não são aceitas cifrassinaturas c^* onde $ID_S^* = ID_R^*$.

1. **Setup** (KGC)
   `Entrada`: 1^n, para um parâmetro de segurança n.
   `Saída`: chave secreta mestra *msk* da KGC e parâmetros públicos *params* (inclui a chave pública mestra *mpk*).

2. **Extract-Partial-Private-Key** (KGC)
   `Entrada`: *params*, *msk*, identificador ID_A do usuário A.
   `Saída`: chave privada parcial D_A de A.

3. **Generate-User-Keys** (A)
   `Entrada`: *params*, ID_A.
   `Saída`: valor secreto x_A e chave pública P_A.

4. **Set-Private-Key** (A)
   `Entrada`: *params*, chave privada parcial D_A e valor secreto x_A.
   `Saída`: chave privada completa S_A.

5. **Signcrypt (Sc)** (S)
   `Entrada`: *params*, mensagem m; informações do emissor (S): ID_S, P_S, S_S; informações do receptor (R): ID_R, P_R; valor aleatório r.
   `Saída`: cifrassinatura c ou símbolo de erro $\perp$.

6. **De-signcrypt (Dsc)** (R)
   `Entrada`: *params*, cifrassinatura c; informações do receptor (R): ID_R, P_R, S_R; informações do emissor (S): ID_S, P_S.
   `Saída`: uma mensagem m ou o símbolo de falha na verificação $\perp$.

Figura 5.2: Esquema CLSC Genérico

Para cada propriedade (confidencialidade e autenticidade), segue o jogo de segurança e a entrada/saída dos oráculos.

Confidencialidade

Abaixo está representado o jogo de segurança que captura a propriedade de confidencialidade de um esquema CLSC contra ambos os tipos de adversário (o tipo é indicado pelo parâmetro x), adversários do tipo I (se $x =$ "I") (não-KGC) e adversários do tipo II (se $x =$ "II") (KGC).

Jogo CLSC-IND-CCA-x

1. $(Msk, params) := Setup(1^n)$.

2. Se $x =$ "II", $aux := Msk$.
3. Senão, $aux := \emptyset$;
4. Fase 1. $(m_0, m_1, ID_S^*, ID_R^*, state) := \mathcal{A}\ _1^{O_1}(params, aux)$.
5. Desafio. $b \in_R \{0,1\}$; $c^* := Sc(m_b, S_S^*, ID_S^*, P_S^*, ID_R^*, P_R^*, params)$.
6. Fase 2. $b' := \mathcal{A}\ _2^{O_2}(c^*, state)$.

A vantagem do adversário $\mathcal{A} = (\mathcal{A}\ _1, \mathcal{A}\ _2)$ neste jogo é definida por:

$$Adv^{CLSC-IND-CCA-x}(A) := |2Pr[b' = b] - 1|$$

No jogo acima, m_0 e m_1 devem ser mensagens de igual comprimento e ID_S^* e ID_R^* devem ser distintos. $\mathcal{A}$ pode guardar alguma informação de estado ($state$) na Fase 1 para utilizar na Fase 2.

Os oráculos O_1 e O_2 que $\mathcal{A}$ tem acesso nas Fases 1 e 2, respectivamente, são os seguintes (diferenças entre o mesmo oráculo em fases diferentes são apresentadas na lista de restrições do oráculo):

1. **Oracle-Request-Public-Key**
   `Entrada`: identidade ID.
   Devolve a chave pública P do usuário cuja identidade é ID. Se ainda não foi criada a chave pública deste usuário, executa o algoritmo `Generate-User-Keys` antes.

2. **Oracle-Replace-Public-Key**
   `Entrada`: identidade ID e chave pública P.
   Substitui a chave pública corrente do usuário ID por P.

3. **Oracle-Extract-Partial-Secret-Key**
   `Entrada`: identidade ID.
   Devolve a chave privada parcial D_{ID} associada a ID. Se a chave privada parcial ainda não foi criada para este usuário, gera esta chave com `Extract-Partial-Private-Key`.

4. **Oracle-Extract-Private-Key**
   `Entrada`: identidade ID.
   Devolve a chave privada completa S_{ID} do usuário ID. Se esta chave ainda não foi criada, gera-a com `Set-Private-Key` (executando `Extract-Partial-Private-Key` antes, se necessário).

5. **Oracle-De-signcrypt**
   `Entrada`: cifrassinatura c, identidade do emissor ID_S, identidade do receptor ID_R. Devolve $m := Dsc(c, S_R, ID_R, ID_S, PK_S, params)$.

Seguem as restrições sobre as consultas aos oráculos para cada tipo de usuário.

Restrições para Adversário Tipo I

1. Em `Oracle-Replace-Public-Key`:
   - Não há restrições.
2. Em `Oracle-Extract-Partial-Private-Key`
   - $\mathcal{A}$ não pode realizar consultas para identidades $ID_S = ID_S^*$ ou $ID_R = ID_R^*$, se a chave pública correspondente a alguma destas identidades foi substituída antes desta consulta. Se a noção *segurança contra insiders* foi adotada, então a restrição aplica-se somente a ID_R^*, consultas para ID_S^* estão liberadas.
3. Em `Oracle-Extract-Private-Key`:
   - $\mathcal{A}$ não pode consultar este oráculo, em momento algum, com as identidades alvo ID_S^* e ID_R^*. Se foi adotada a noção *segurança contra insiders*, então esta restrição se aplica somente a ID_R^*, consultas cuja identidade do emissor é ID_S^* estão liberadas.
   - $\mathcal{A}$ não pode realizar esta consulta para usuário ID tal que a chave pública de ID foi substituída antes desta consulta (digamos, por P'). Esta última restrição é considerada necessária neste modelo, pois, para $\mathcal{C}$ computar a valor secreto a partir da chave pública, ele precisaria de tempo não-polinomial. (Veja as considerações abaixo sobre a exigência de fornecimento do valor secreto correspondente a chave pública substituta)
4. Em `Oracle-De-signcrypt`:
   - Não são permitidas consultas tais que $ID_S = ID_R$.
   - Na Fase 2 não são permitidas consultas tais que $c = c^*$, $ID_S = ID_S^*$, $ID_R = ID_R^*$ e ambas P_S^* e P_R^* não foram substituídas após o desafio ter sido lançado (e antes da consulta a este oráculo).
   - Se a chave pública do receptor foi substituída (digamos, por P_R') antes da consulta a este oráculo, então é exigido que $\mathcal{A}$ forneça o valor secreto x_R' correspondente a P_R', para que $\mathcal{C}$ possa executar `Dsc` (Veja as considerações abaixo sobre

a exigência de fornecimento do valor secreto correspondente a chave pública substituta).

Restrições para Adversário Tipo II

1. Em `Oracle-Replace-Public-Key`:
   - $\mathcal{A}$ não pode substituir as chave públicas de ID_S^* ou ID_R^* antes do desafio ser lançado. Se foi adotada a noção *segurança contra insiders*, então esta restrição se aplica somente a ID_R^*.
2. Em `Oracle-Extract-Partial-Private-Key`
   - Não há restrições.
3. Em `Oracle-Extract-Private-Key`:
   - Mesmas restrições do oráculo respectivo para adversário Tipo I.
4. Em `Oracle-De-signcrypt`:
   - Mesmas restrições do oráculo respectivo para adversário Tipo I.

Considerações sobre a exigência de $\mathcal{A}$ prover o valor secreto correspondente à chave pública substituta

Segundo os autores do modelo de segurança para CLSC apresentado nesta seção [BF08], esta hipótese é realista, pois é impossível para $\mathcal{C}$ computar em tempo polinomial o valor secreto de um usuário cuja chave pública foi substituída. Segundo Dantas [Dan08], no contexto de CLS, esta hipótese significa, no fundo, que o único meio que o adversário tem para conseguir uma chave pública válida para um usuário é através da obtenção do valor secreto, e então, de posse deste valor, computando a chave pública.

Entretanto, conforme indicado em [Dan08], alguns esquemas de CLS demonstrados seguros que faziam esta hipótese em seu modelo de segurança foram posteriormente atacados, e tais ataques violavam exatamente esta hipótese. Ou seja, estes esquemas eram vulneráveis, e os ataques obtidos estavam fora do modelo segurança no qual os esquemas haviam sido demonstrados.

Esta hipótese é *aparentemente* realista, pois teoricamente o desafiante tem liberdade para simular os oráculos como desejar, e portanto, este poderia, *de algum modo*, simular os oráculos para fazer com seja fácil para ele computar o valor secreto a partir da chave pública.

Devido a este problema, talvez esta hipótese também não seja adequada para o modelo de segurança de CLSC. Em um esquema CLSC, quando for necessário tratar este aspecto da demonstração de segurança do esquema, informaremos qual dos tipos será considerado.

Autenticidade

Abaixo está representado o jogo de segurança que captura a propriedade de autenticidade de um esquema CLSC contra ambos os tipos de adversário (o tipo é indicado pelo parâmetro x), adversários do tipo I (se $x =$ "I") (não-KGC) e adversários do tipo II (se $x =$ "II") (KGC).

Jogo CLSC-sEUF-CMA-x

1. $(Msk, params) := Setup(1^n)$
2. Se $x =$ "II", $aux := Msk$.
3. Senão, $aux := \emptyset$;
4. $(c^*, ID_S^*, ID_R^*) := \mathcal{A}^{\,O}(params, aux))$

A vantagem do adversário $\mathcal{A}$ neste jogo é definida por:

$$Adv^{CLSC-sEUF-CMA-x}(A) := Pr[m^* \neq \perp \wedge (m^*, ID_S^*, P_S^*, ID_R^*, P_R^*, c^*) \notin L],$$

onde $m^* = Dsc(c^*, S_R^*, ID_R^*, P_R^*, ID_S^*, P_S^*, param)$, e L é a lista de consultas e respectivas respostas obtidas por $\mathcal{A}$ ao oráculo `Oracle-Signcrypt`. Os elementos de L são da forma $(m, ID_S, P_S, ID_R, P_R, c)$, onde c é a resposta que o oráculo `Oracle-Signcrypt` retornou para a consulta $(m, ID_S, P_S, ID_R, P_R)$.

O adversário tem acesso a todos os oráculos do Jogo CLSC-IND-CCA-x, e mais o oráculo de cifrassinatura (`Oracle-Signcryption`).

1. **Oracle-Signcryption**.
   `Entrada`: mensagem m, identidade do emissor ID_S, identidade do receptor ID_R.
   Devolve $c := Sc(m, S_S, ID_S, P_S, ID_R, P_R, params)$.

Seguem as restrições sobre as consultas aos oráculos para cada tipo de usuário.

Restrições para Adversário Tipo I

1. Em `Oracle-Replace-Public-Key`
   - Não há restrições.
2. Em `Oracle-Extract-Partial-Private-Key`
   - $\mathcal{A}$ não pode realizar consultas para identidades $ID_S = ID_S^*$ ou $ID_R = ID_R^*$, se a chave pública correspondente a alguma destas identidades foi substituída antes desta consulta. Se a noção *segurança contra insiders* foi adotada, então a restrição aplica-se somente a ID_S^*, consultas para ID_R^* estão liberadas.

3. Em `Oracle-Extract-Private-Key`:

   - $\mathcal{A}$ não pode consultar este oráculo, em momento algum, com as identidades alvo ID_S^* e ID_R^*. Se foi adotada a noção *segurança contra insiders*, então esta restrição se aplica somente a ID_S^*, consultas cuja identidade do emissor é ID_R^* estão liberadas.
   - $\mathcal{A}$ não pode realizar esta consulta para usuário ID tal que a chave pública de ID foi substituída antes desta consulta (digamos, por P'). (Veja as considerações sobre a exigência de fornecimento do valor secreto correspondente à chave pública substituta)

4. Em `Oracle-Signcrypt`:

   - Não são permitidas consultas tais que $ID_S = ID_R$.
   - Se a chave pública de ID_S foi substituída (digamos, por P'_S) antes da consulta a este oráculo, então é exigido que $\mathcal{A}$ forneça o valor secreto x'_s correspondente a P'_S, para que $\mathcal{C}$ possa executar `Sc` (Veja as considerações sobre a exigência de fornecimento do valor secreto correspondente à chave pública substituta).

5. Em `Oracle-De-signcrypt`:

   - Não são permitidas consultas tais que $ID_S = ID_R$.
   - Se a chave pública de ID_R foi substituída (digamos, por P'_R) antes da consulta a este oráculo, então é exigido que $\mathcal{A}$ forneça o valor secreto x'_r correspondente a P'_R, para que $\mathcal{C}$ possa executar `Sc` (Veja as considerações sobre a exigência de fornecimento do valor secreto correspondente à chave pública substituta).

Restrições para Adversário Tipo II

1. Em `Oracle-Replace-Public-Key`:

   - $\mathcal{A}$ não pode substituir a chave pública de ID_S^* (ou ID_R^*). Se foi adotada a noção *segurança contra insiders*, então esta restrição se aplica somente a ID_S^*.

2. Em `Oracle-Extract-Partial-Private-Key`

   - Não há restrições.

3. Em `Oracle-Extract-Private-Key`:

   - Mesmas restrições do oráculo respectivo para adversário Tipo I.

4. Em `Oracle-Signcrypt`:

- Mesmas restrições do oráculo respectivo para adversário Tipo I.

5. Em `Oracle-De-signcrypt`:

   - Mesmas restrições do oráculo respectivo para adversário Tipo I.

5.3 Revisão da Literatura

5.3.1 CLSC Barbosa-Farshim [BF08]

O esquema CLSC Barbosa-Farshim [BF08] foi o primeiro esquema de CLSC. Este esquema utiliza o paradigma *Encrypt-then-Sign* [ADR02], porém esta composição não é genérica, pois é feito *reuso de aleatoriedade* (randomness reuse) entre o esquema de encriptação e o esquema de assinatura.

A componente do esquema responsável pela assinatura é baseada no CLS Zhang-Wong-Xu-Feng [ZWXF06], o qual por sua vez é derivado do IBS Libert-Quisquater [LQ04]. A demonstração de segurança original do CLS Zhang-Wong-Xu-Feng estava incorreta, e foi mostrado que o esquema é seguro em um modelo de segurança mais fraco do que o modelo tradicional para CLS.

A componente responsável pela encriptação é baseada no CLE Cheng-Comley [CC05], o qual foi demonstrado seguro, porém em um modelo de segurança para CLE mais fraco que o usual.

Apesar de ambos os esquemas-base do CLSC Barbosa-Farshim, CLS Zhang-Wong-Xu-Feng e CLE Cheng-Comley, não possuirem demonstrações em modelos de segurança fortes, segundo os autores, a técnica de reuso de aleatoriedade permitiu obter um esquema com segurança superior a segurança obtida individualmente com cada componente. Porém, apesar de existir uma demonstração de segurança para o CLSC Barbosa-Farshim, foi encontrado um ataque de falsificação contra ele [SVR10b], mostrando que o esquema é inseguro.

O esquema CLSC Barbosa-Farshim é apresentado nas Figuras 5.3 e 5.4.

Análise de Segurança

O artigo original [BF08] do CLSC Barbosa-Farshim continha uma demonstração de segurança no modelo ROM para este esquema. No requisito confidencialidade, foi demonstrado que o esquema é IND-iCCA-I e IND-iCCA-II seguro contra atacantes insiders (por isso o "i") sob a hipótese de intratabilidade do G-BDHP em (G_1, G_T). No requisito autenticidade, foi demonstrado que o CLSC Barbosa-Farshim é sEUF-iCMA-I e sEUF-iCMA-II seguro sob a hipótese de intratabilidade do G-CDHP-BDDH.

1. **Setup**(1^n)
   (a) Escolhe grupo aditivo G_1 de ordem prima p e grupo multiplicativo G_T de ordem p. Para todos estes grupos, supõe-se que as operações de grupo são eficientemente computáveis.
   (b) Escolhe gerador P de G_1.
   (c) $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ é um emparelhamento bilinear não-degenerado eficientemente computável.
   (d) Escolhe funções hash criptográficas

$$H_1 : \{0,1\}^* \to \mathbb{G}_1; H_2 : \{0,1\}^* \to \{0,1\}^\kappa;$$
$$H_3 : \{0,1\}^* \to \mathbb{G}_1; H_4 : \{0,1\}^* \to \mathbb{G}_1.$$

   (e) Escolhe $Msk \in_R \mathbb{Z}_p$. Faz $Mpk := Msk \cdot P$.
   (f) O espaço de mensagens é $\mathcal{M} = \{0,1\}^\kappa$ e o espaço de textos cifrados é $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^\kappa \times \mathbb{G}_1$.
   (g) Devolve $params := (Mpk, G_1, G_T, P, H_1, H_2, H_3, H_4)$.
2. **Extract-Partial-Private-Key**$(params, ID, Msk)$
   (a) Devolve $D := Msk \cdot H_1(ID)$.
3. **Generate-User-Keys**$(ID, params)$
   (a) Escolhe $x_{ID} \in_R \mathbb{Z}_p$ como o valor secreto.
   (b) Calcula a chave pública $P_{ID} := x_{ID}P$.
   (c) Devolve (x_{ID}, P_{ID}).
4. **Set-Private-Key**$(params, D, x)$
   (a) Devolve $S_{ID} := (x, D)$.

Figura 5.3: Esquema CLSC Barbosa-Farshim (Parte 1)

1. **Sc**$(params, m, S_S, ID_S, P_S, ID_R, P_R)$
   (a) $r \in_R \mathbb{Z}_p$; $U := rP$; $T := e(Mpk, Q_R)^r$
   (b) $h := H_2(U, T, rP_R, ID_R, P_R)$
   (c) $V := m \oplus h$
   (d) $H := H_3(U, V, ID_S, P_S)$
   (e) $H' := H_4(U, V, ID_S, P_S)$
   (f) $(x_S, D_S) := S_S$
   (g) $W := D_S + rH + x_S H'$
   (h) Devolve $c := (U, V, W)$
2. **Dsc**$(params, c, S_R, ID_R, P_R, ID_S, P_S)$
   (a) $(U, V, W) := c$
   (b) $H := H_3(U, V, ID_S, P_S)$
   (c) $H' := H_4(U, V, ID_S, P_S)$
   (d) Se $e(Mpk, Q_S)e(U, H)e(P_S, H') \neq e(P, W)$, Devolve $\perp$
   (e) $(x_R, D_R) := S_R$
   (f) $T := e(D_R, U)$
   (g) $h := H_2(U, T, x_R U, ID_R, P_R)$
   (h) Devolve $m := V \oplus h$

Figura 5.4: Esquema CLSC Barbosa-Farshim (Parte 2)

1. Durante o jogo CLSC-sEUF-CMA-I (ou CLSC-sEUF-CMA-II), $\mathcal{A}$ escolhe o usuário alvo ID_S^*, o usuário alvo ID_R^*, um usuário arbitrário A e uma mensagem m.

2. $\mathcal{A}$ realiza a consulta `Oracle-Signcryption`(m, ID_S^*, ID_A), obtendo a resposta $c = (U, V, W)$.

3. $\mathcal{A}$ submete $c^* := c$ a $\mathcal{C}$ como a cifrassinatura falsificada, alegando que esta é uma cifrassinatura de ID_S^* para ID_R^*.

4. $\mathcal{C}$ então verifica se c^* é cifrassinatura de ID_S^* para ID_R^*. De fato, isto é verdade, c^* é uma cifrassinatura de $m^* = m \oplus h \oplus h^*$ de ID_S^* para ID_R^*, onde $h^* = H_2(U, T^*, x_R^*U, ID_R^*, P_R^*)$, $h = H_2(U, T, x_AU, ID_A, P_A)$ e $T^* = e(D_R^*, U)$, pois:

   (a) $\mathcal{C}$ faz $H^* := H_3(U, V, ID_S^*, P_S^*)$ e $H'^* := H_4(U, V, ID_S^*, P_S^*)$.
   (Observe que $H^* = H$ e $H'^* = H'$, onde H e H' são os valores análogos obtidos por $\mathcal{A}$ na simulação de `Oracle-Signcryption` no Passo 2).

   (b) a cifrassinatura tem verificação positiva:

$$\begin{aligned} e(Mpk, Q_S^*)e(U^*, H^*)e(P_S^*, H') &= e(Mpk, Q_S^*)e(U, H)e(P_S, H') \\ &= e(P, W) \\ &= e(P, W^*) \end{aligned}$$

Figura 5.5: Ataque de Selvi, Vivek e Rangan contra CLSC Barbosa-Farshim

Apesar de conter uma demonstração de segurança, o CLSC Barbosa-Farshim foi criptanalisado por Selvi et al [SVR10b], os quais mostraram um ataque de falsificação sobre o esquema. A falsificação obtida é somente existencial, ou seja, o adversário não tem controle sobre a mensagem (m) contida na cifrassinatura (c). Este ataque é descrito na Figura 5.5.

O adversário não tem controle sobre a mensagem m^* que foi cifrassinada, pois todos os bits de m^* dependem de h^*, cujo valor $\mathcal{A}$ não tem controle (h^* depende do aleatório r usado na simulação de `Oracle-Signcryption` (Passo 2)).

Segundo [SVR10b], a razão por trás do ataque é: em [ADR02] foi provado que esquemas de cifrassinatura seguros sob o paradigma *Encrypt-then-Sign* devem ter (1) a identidade do emissor (ID_S) *ligada* à encriptação e (2) a identidade do receptor (ID_R) *ligada* à assinatura. O esquema CLSC Barbosa-Farshim falha no requisito 2, propiciando o ataque descrito. No entanto, conforme nós observamos, o esquema também falha no requisito 1, o que o torna (muito provavelmente) vulnerável também a ataques contra a confidencialidade, apesar de um ataque contra este requisito ainda não ter sido levantado.

Correção Proposta

Seguindo as recomendações de [ADR02] para cifrassinatura sob o paradigma *Encrypt-then-Sign*, fazemos a seguintes mudanças no esquema:

1. a função H_2 passa a também receber ID_S e P_S como parâmetros, isto é, em `Sc`, $h := H_2(U, T, rP_R, ID_S, P_S, ID_R, P_R)$ e em `Dsc`, $h := H_2(U, T, x_R U, ID_S, P_S, ID_R, P_R)$. Isto satisfaz o requisito (1).

2. as funções H_3 e H_4 passam a também receber ID_R e P_R como parâmetros, isto é, em `Sc` e `Dsc`, H e H' são calculados da seguinte forma: $H := H_3(U, V, ID_S, P_S, ID_R, P_R)$ e $H' := H_4(U, V, ID_S, P_S, ID_R, P_R)$. Isto satisfaz o requisito (2).

Eficiência

Sc: 1 emp.
Dsc: 5 emp. (1 deles pode ser pré-computado)

5.3.2 CLSC Aranha-Castro-López-Dahab [ACLD08]

O esquema [ACLD08] é uma extensão de um eficiente esquema de cifrassinatura baseada em identidades (IBSC) proposto em [MB04], herdando deste o recurso de verificação pública. O esquema pode ser instanciado utilizando emparelhamentos simétricos ou assimétricos, e é apresentado na Figura 5.6.

Análise de Segurança

O esquema foi criptanalisado por Selvi et al [SVR10b], os quais mostraram ataques sobre a confidencialidade (Figura 5.7) e autenticidade (5.8).

Eficiência

Sc: 0 emp.
Dsc: 2 emp.

5.3.3 CLSC Barreto et al [BDC+08]

Este esquema é baseado no PKSC de Zheng [Zhe97a], em assinatura do tipo Schnorr [Sch91], e no IBS BLMQ [BLMQ05]. Ele possui um algoritmo para validação da chave pública, o qual deve ser executado em `Sc` para validar P_R, e em `Dsc` para validar P_S. Após ter sido executado uma vez para uma dada chave pública, o resultado pode ser armazenado para ser reutilizado, ao efetuar a validação da mesma chave pública.

1. **Setup**(1^n)

   (a) Dado um parâmetro de segurança n, a KGC escolhe um número primo q de n-bits, grupos bilineares $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ de ordem q com geradores $P \in \mathbb{G}_1$ e $Q \in G_2$, um mapa bilinear admissível e, as funções de hash
   - $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$,
   - $H_2 : \mathbb{G}_T \times \mathbb{G}_1 \times \{0,1\}^* \to \{0,1\}^l$ e
   - $H_3 : \{0,1\}^l \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_1 \times \{0,1\}^* \to \mathbb{Z}_q^*$,

   e a chave mestra $Msk \in_R \mathbb{Z}_q^*$;

   (b) Calcula $Mpk := Msk \cdot P$ e $g = e(P, Q)$;

   (c) Devolve $params := (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, Q, e, g, Mpk, H_1, H_2, H_3)$.

2. **Extract-Partial-Private-Key**$(params, ID_A)$

   (a) $y_A := H_1(ID_A)$;

   (b) Devolve $D_A := (y_A + Msk)^{-1}Q$.

3. **Generate-User-Keys**$(params, ID_A, D_A)$

   (a) Escolhe valor secreto $x_A \in_R \mathbb{Z}_q^*$;

   (b) Calcula a chave privada $S_A := x_A^{-1} D_A$;

   (c) Calcula a chave pública $P_A := x_A(y_A P + P_{pub})$;

   (d) Devolve (P_A, S_A). Observe que $e(P_A, Q_A) = e(P, Q) = g$.

4. **Sc**$(m, ID_A, S_A, P_A, ID_B, P_B)$

   (a) Escolhe $r \in_R \mathbb{Z}_q^*$. Calcula $u := r^{-1}$, $U := g^u$, $c := m \oplus H_2(U)$;

   (b) $h := H_3(c, rP_A, uP_B)$, $T := (r + h)^{-1} S_A$;

   (c) Devolve $(c, R, S, T) := (c, rP_A, uP_B, T)$.

5. **Dsc**$((c, R, S, T), ID_A, P_A, ID_B, S_B, P_B)$

   (a) $h' := H_3(c, R, S)$, $V := e(R + h'P_A, T)$;

   (b) Se $V \neq g$, devolve $\perp$;

   (c) $U' := e(S, S_B)$, $m' := c \oplus H_2(U')$;

   (d) Devolve m'.

Figura 5.6: Esquema CLSC Diego-Castro-Lopez-Dahab

Um adversário do Tipo I consegue forjar uma cifrassinatura válida sobre uma mensagem m arbitrária de um usuário ID_A para um usuário ID_B (arbitrários) do seguinte modo:

1. $\mathcal{A}$ escolhe $r \in_R \mathbb{Z}_q^*$, faz $u := r^{-1}$.
2. $U := g^u$ e $c := m \oplus H_2(U)$.
3. $T := uQ$, $R := rP - P$ e $S := uP_B$.
4. $h := H_3(c, R, S)$.
5. $P'_A := h^{-1}P$.
6. Substitui a chave pública de ID_A por P'_A.
7. Devolve $\sigma := (c, R, S, T)$, alegando ser uma cifrassinatura de ID_A para ID_B.

Podemos ver que σ é de fato uma cifrassinatura válida de ID_A para ID_B, pois:

$$\begin{aligned} e(R + hP'_A, T) &= e(rP - P + hh^{-1}P, uQ) \\ &= e(rP, uQ)e(-P + P, uQ) \\ &= e(P, Q) \\ &= g \end{aligned}$$

Figura 5.7: Ataque à Inforjabilidade do CLSC Diego-Castro-Lopez-Dahab por Selvi et al

Um adversário do Tipo I (ou do Tipo II) consegue determinar a mensagem escolhida no desafio. Segue o ataque:

1. $\mathcal{A}$ escolhe mensagens m_0 e m_1, e usuários ID_A e ID_B. Submete estas informações ao desafiante.

2. Desafiante calcula $\sigma^* := (c^*, R^*, S^*, T^*) = Sc(m_b, ID_A, ID_B)$, onde $b \in \{0, 1\}$ é o bit de desafio.

3. $\mathcal{A}$ constrói uma cifrassinatura σ' sobre a mensagem m_b de ID_C para ID_B (por hipótese, $\mathcal{A}$ sabe a chave privada S_C de ID_C):

   (a) $c' := c^*$;

   (b) Escolhe $r' \in_R \mathbb{Z}_q^*$, e faz $R' := r'P_C$

   (c) $S' := S^*$;

   (d) $h' := H_3(c', R', S')$;

   (e) $T' := (r' + h')^{-1} S_C$;

   (f) $\sigma' := (c', R', S', T')$.

4. $\mathcal{A}$ consulta σ' ao oráculo `Oracle-De-signcrypt`, obtendo m_b como resposta, pois dado que $U' = e(S', S_B) = e(S^*, S_B) = U^*$ e $S' = S^*$, logo $m' = c' \oplus H_2(U') = c^* \oplus H_2(U^*) = m_b$. Com isto, $\mathcal{A}$ determina o valor do bit b.

Este ataque não utiliza a chave secreta mestra, e nem a substituição de chaves públicas, logo é válido para adversários do Tipo I ou do Tipo II.

Figura 5.8: Ataque à Confidencialidade do CLSC Diego-Castro-Lopez-Dahab por Selvi et al

O CLSC Barreto et al difere de outros esquemas CLSC no que se refere da técnica peculiar empregada por este para a certificação das chaves públicas. Em CLSC as chaves públicas são implicitamente certificadas, e o CLSC Barreto et al em particular combina assinaturas do tipo Schnorr com assinaturas do IBS BLMQ para obter as assinaturas das chaves públicas dos usuários. Portanto, sempre que a chave pública de um usuário precisa ser utilizada, esta é primeiramente validada pelo algoritmo de validação de chave pública, por meio da verificação da respectiva assinatura da chave pública.

Aderência ao Paradigma Sem Certificados

Este esquema é considerado certificateless não estrito, no sentido de que ele não segue o modelo CLE Al-Riyami-Paterson. Este esquema segue o modelo Baek-Safavi-Susilo para CLE (Seção 4.2.2), tornando-o imune a ataques de negação de decriptação. A geração da chave pública depende da chave privada completa (valor secreto + chave privada parcial), ou seja, não há independência temporal na geração destas chaves. Por causa desta restrição, o esquema não suporta "cifrassinatura para o futuro" [2].

Análise de Segurança

Os autores argumentam que a demonstração de segurança do esquema seria idêntica à demonstração de segurança (no ROM) do esquema de cifrassinatura de chave pública (PKSC) de Zheng [Zhe97a] fornecida por Baek et al [BSZ02], se os *valores públicos* y_A e y_B (do emissor e do receptor, respectivamente) fossem chaves públicas explicitamente certificadas, como é no paradigma PKSC.

No entanto, no CLSC Barreto et al a certificação da chave pública é feita através da aplicação do algoritmo de verificação de chave pública sobre a assinatura da chave pública do usuário em questão. Portanto, para mostrar que o esquema é seguro basta mostrar que o mecanismo de assinatura da chave pública é seguro.

Dado que os esquemas componentes, esquema de assinatura de Schnorr [Sch91] e IBS BLMQ [BLMQ05], têm demonstração de segurança no ROM, os autores reduzem ambos os problemas FAPIP [BLMQ05] e falsificação no IBS BLMQ à falsificação da assinatura da chave pública, concluindo assim a demonstração de que o esquema é seguro no ROM.

Eficiência

Sc: 1 emp. (para a validação de P_R).
Dsc: 1 emp. (para a validação de P_S).

[2] Se fosse um esquema CLE, a propriedade seria "encriptação para o futuro".

5.3.4 CLSC Selvi-Vivek-Rangan [SVR10b, SVR11]

Uma característica interessante deste esquema é que ele não utiliza emparelhamentos, o que o torna eficiente. Entretanto, ele não suporta verificação pública de cifrassinatura.

Análise de Segurança

Em [SVR10b] os autores enunciam os teoremas sobre a segurança do esquema no modelo ROM, incluindo a complexidade de cada redução: confidencialidade para adversário Tipo I e Tipo II sob a dificuldade do DLP, e inforjabilidade para adversário Tipo I e Tipo II sob a dificuldade do CDHP. Porém, não são fornecidas demonstrações para estes teoremas.

Em [SVR11] há demonstrações de segurança para os teoremas de [SVR10b] relativos a adversários do Tipo 1.

Eficiência

Sc: 0 emp.
Dsc: 0 emp.

5.3.5 CLSC Xie-Zhang I [XZ10a]

Análise de Segurança

Autores apresentam demonstrações de segurança no ROM para a confidencialidade (IND-CCA2) sob a hipótese G-DHP, e para a autenticidade (EUF-CMA) sob a hipótese G-DLP (Gap Discrete Logarithm Problem) [BSZ02].

Eficiência

Sc: 0 emp.
Dsc: 0 emp.

5.3.6 CLSC Xie-Zhang II [XZ10b]

Análise de Segurança

Continha demonstração de segurança no ROM, porém Selvi et al [SVR10c] mostraram que o esquema é inseguro.

Eficiência

Sc: 0 emp.
Dsc: 2 emp.

5.3.7 CLSC Liu-Hu-Zhang-Ma [LHZM10]

Este trabalho introduziu um modelo de segurança formal que incorpora ataques de KGC maliciosa (passiva), e apresentou um esquema supostamente seguro sob este modelo de segurança, sem o uso de oráculos aleatórios (modelo padrão). Se fosse seguro, este seria o primeiro CLSC resistente a ataques de KGC maliciosa.

Análise de Segurança

Os autores demonstraram, no modelo padrão, a segurança do esquema sob o modelo de segurança assim definido. Apesar da demonstração de segurança, Selvi et al [SVR10c] apresentou um ataque baseado em substituição de chave pública, mostrando que o esquema é inseguro contra Adv. do Tipo I. Independentemente, Weng et al [WYD+11] apresentaram dois ataques ao esquema: um contra a confidencialidade, e o outro sobre a autenticidade.

Eficiência

Sc: 0 emp.
Dsc: 5 emp.

5.3.8 CLSC Wu-Chen [WC08]

Análise de Segurança

Selvi et al [SVR10b] apresentaram um ataque contra o esquema que mostra que este não é semanticamente seguro e as cifrassinaturas são falsificáveis. Independemente, Zhang e Geng [ZG09] apresentaram um ataque à confidencialidade do esquema, diferente do ataque de Selvi et al, sem utilizar quaisquer oráculos encriptação ou decriptação, ou ainda substituição de chave pública. Neste mesmo trabalho [ZG09], os autores propuseram uma correção ao esquema que resiste a este ataque, porém a segurança do esquema resultante ainda não foi demonstrada.

Eficiência

Sc: 1 emp.
Dsc: 3 emp.

5.3.9 CLSC Jin-Wen-Zhang [JWZ10]

O esquema é baseado no CLSC Liu-Hu-Zhang-Ma, com uma modificação feita sobre este para que ele resista ao ataque de substituição de chave pública (Adv. Tipo I) em [SVR10c]. A modificação consiste em adicionar uma assinatura *one-time* do tipo Schnorr [BS07] a cada chave pública de usuário.

Aderência ao Paradigma Sem Certificados

Neste esquema, a chave pública depende da chave privada completa, e portanto, pelas mesmas razões do CLSC Barreto et al (Seção 5.3.3), este esquema é certificateless não estrito.

Análise de Segurança

Os autores apresentam uma demonstração de segurança no modelo padrão.

Eficiência

Sc: 0 emp.
Dsc: 5 emp.

5.3.10 CLSC Li-He-Li-Liu [LHLL10]

Análise de Segurança

Os autores demonstraram a segurança do esquema no ROM. Mais especificamente, o esquema é IND-iCCA-I seguro sob a hipótese de dificuldade do k-CCA [3] [LHLL10], IND-iCCA-II seguro sob a hipótese de dificuldade do mICDH [4] [LHLL10], sEUF-iCMA-I seguro sob a hipótese de dificuldade do k-CCA e sEUF-iCMA-II seguro sob a hipótese de dificuldade do mICDH.

[3] Collusion Attack Algorithm with k-traitors
[4] Modified Inverse Computational Diffie-Hellman

Eficiência

Sc: 0 emp.
Dsc: 2 emp.

5.3.11 Comparação

A Tabela 5.1 apresenta uma comparação dos esquemas CLSC discutidos nas subseções anteriores.

A coluna "Modelo" contém o modelo no qual a demonstração foi realizada, podendo ser: ROM (oráculo aleatório), Padrão (modelo padrão) ou "-" (não há demonstração de segurança).

A coluna "Propriedades"lista propriedades *especiais* de certos esquemas. Neste coluna, "Verificação Pública" se refere a propriedade de que a verificação da autenticidade da cifrassinatura (mas não necessariamente da mensagem contida na cifrassinatura) pode ser realizada utilizando somente informações públicas, não sendo necessário conhecer a chave privada do usuário receptor. Esquemas CLSC com esta característica provêem irretratabilidade.

Alguns esquemas de cifrassinatura possuem algoritmos específicos para verificação da autenticidade da cifrassinatura (ou da própria mensagem) que dependem do valor da mensagem. Em tais esquemas, no caso de um conflito entre emissor e receptor, o algoritmo de verificação deve ser executado, e nesta ocasião a confidencialidade da mensagem contida na cifrassinatura será necessariamente violada. Por causa deste problema, não consideraremos esta propriedade na Tabela 5.1.

A coluna "Eficiência" tem entradas da forma $X + Y$, onde X é o número de emparelhamentos no algoritmo `Signcrypt` e Y é o número de emparelhamentos no algoritmo `De-Signcrypt`. Estamos considerando somente o número de emparelhamentos nesta coluna, pois, apesar de diversos avanços na eficiência da computação de emparelhamentos bilineares, estes ainda continuam tendo tempo de execução ordens de magnitude superior ao tempo de execução das outras operações (cálculo de hash, operações de grupo, etc.), por isso, este tipo de operação *normalmente* domina o tempo de execução dos algoritmos nos quais são empregadas.

5.3.12 Variações sobre CLSC

Até o momento consideramos apenas cifrassinaturas "tradicionais" sob o paradigma sem certificados, mas a literatura sobre CLSC também contém variações do conceito de cifrassinatura convencional, com funcionalidades adicionais ou distintas. Dentre estas variações e esquemas, temos:

Tabela 5.1: Comparação de Esquemas de Cifrassinatura Sem Certificados

Esquema	Modelo	Propriedades	Efic.	Status
Barbosa-Farshim [BF08]	ROM	Verificação Pública	1 + 5	Quebrado
Aranha-Castro-Lopez-Dahab [ACLD08]	-	Verificação Pública	0 + 2	Quebrado
Barreto et al [BDC$^+$08]	ROM	-	1 + 1	OK[a]
Selvi-Vivek-Rangan [SVR10b, SVR11]	ROM	-	0 + 0	OK
Xie-Zhang I [XZ10a]	ROM	-	0 + 0	OK
Xie-Zhang II [XZ10b]	ROM	-	0 + 2	Quebrado
Liu-Hu-Zhang-Ma [LHZM10]	Padrão	Verificação Pública	0 + 5	Quebrado
Wu-Chen [WC08]	ROM	-	1 + 3	Quebrado
Jin-Wen-Zhang [JWZ10]	Padrão	Verificação Pública	0 + 5	OK[b]
Li-He-Li-Liu [LHLL10]	Padrão	-	0 + 2	OK

[a] O esquema não é certificateless estrito (Seção 5.3.3).
[b] O esquema não é certificateless estrito (Seção 5.3.9).

- Certificateless Ring Signcryption (CLRSC)
  Esta variante adiciona a propriedade de anonimato a CLSC, um usuário é capaz de cifrassinar anonimamente uma mensagem para os membros de um anel (grupo), de modo que o receptor não consegue distinguir qual membro do grupo cifrassinou a mensagem. Exemplo de esquema: [WZM07].

- Certificateless Multi-receiver Signcryption (CLMSC)
  Nesta variante de CLSC, um emissor produz uma única cifrassinatura de uma única mensagem para um conjunto de destinatários. Um esquema CLMSC deve prover confidencialidade, autenticidade e irretratabilidade para cada um dos receptores. Exemplos de esquemas: [SVSR08, SVR09, MZZ10].

- Certificateless Generalized Signcryption (CLGSC)
  Esta variante provê cifrassinatura tradicional (encriptação + assinatura), mas também cada operação , encriptação ou assinatura, isoladamente. Exemplo de esquema: [HWL10].

- Certificateless Hybrid Signcryption (CLHSC) ou Certificateless Signcryption with Key Encapsulation Mechanism (CLSC-KEM)
  Esta variante consiste no uso de técnicas híbridas (combinação de criptografia de chave pública com criptografia de chave privada) para a construção de esquemas de cifrassinatura sem certificados. Exemplos de esquemas: [LST09, LBN10, SVR10a].

5.4 Ataque ao IBSC McCullagh-Barreto

Nesta seção descrevemos um ataque contra a confidencialidade do IBSC com verificação pública de cifrassinatura (transferível) de McCullagh e Barreto [MB04], mostrando, portanto, que o esquema é inseguro.

Este ataque é análogo ao ataque de Selvi et al [SVR10b] sobre a confidencialidade do CLSC Aranha-Castro-Lopez-Dahab [ACLD08]. O esquema é apresentado na Figura 5.9, e o ataque na Figura 5.10.

1. **Setup**(1^n)
   (a) Sejam $\mathbb{G}_1$, $\mathbb{G}_2$ e $\mathbb{G}_T$ grupos "pairing-friendly" e $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ um emparelhamento. Os geradores de $\mathbb{G}_1$, $\mathbb{G}_2$ e $\mathbb{G}_T$ são P, Q e $g = e(P, Q)$, respectivamente.
   (b) Escolhe inteiro r e funções hash criptográficas $H_0 : \{0,1\}^* \to \mathbb{Z}_r^*$, e H_1 e H_3 com domínios e contra-domínios apropriados.
   (c) Gera um polinômio aleatório secreto $s(x) = \sum_{i=0}^{d} s_i x^i \in \mathbb{Z}_r[x]$ e o atribui como chave secreta.
   (d) Faz $P_i := s_i P$.
   (e) Os parâmetros públicos são $params = (P, Q, g, P_0, ..., P_d))$.
2. **Keygen**$(params, ID_A)$
   (a) $a := H_0(ID_A)$;
   (b) Calcula a chave privada: $Q_A := s(a)^{-1}Q$;
   (c) Envia Q_A para A através de um canal seguro.
   (d) Observação: A chave pública de A é computada (publicamente) via $P_A := \sum_{i=0}^{d} a^i(s_i P)$.
3. **Sc**(m, Q_A, P_A, ID_B)
   (a) Calcula a chave pública P_B de ID_B.
   (b) Escolhe $x \in_R \mathbb{Z}_r^*$;
   (c) $N := g^{x^{-1}}$, $R := xP_A$, $S := x^{-1}P_B$;
   (d) $c := H_1(N) \oplus m$, $h := H_3(R, S, c)$, $T := (x + h)^{-1}Q_A$;
   (e) Devolve $\sigma := (c, R, S, T)$.
4. **Dsc**$((c, R, S, T), ID_A, Q_B)$
   (a) Calcula a chave pública P_A de ID_A;
   (b) $h := H_3(R, S, c)$, $V := e(R + hP_A, T)$;
   (c) Se $V \neq g$, devolve $\perp$;
   (d) $N := e(S, Q_B)$;
   (e) Devolve $m := H_1(N) \oplus c$.

Figura 5.9: Esquema IBSC McCullagh-Barreto

Sejam $\mathcal{A}$ o adversário e $\mathcal{C}$ o desafiante.

$\mathcal{A}$:

1. Escolhe mensagens m_0 e m_1, e usuários alvo ID_A e ID_B.
2. Submete estes valores para $\mathcal{C}$.

$\mathcal{C}$:

1. Escolhe $i \in_R \{0, 1\}$.
2. Calcula o desafio $\sigma^* := (c^*, R^*, S^*, T^*) = Sc(m_i, Q_A, P_A, ID_B)$, e o envia para $\mathcal{A}$.
3. Observação: se x é o valor aleatório escolhido por $\mathcal{C}$ ao calcular o desafio, então:
   (a) $S^* = x^{-1}P_B$;
   (b) $N^* = g^{x^{-1}}$;
   (c) $c^* = H_1(N^*) \oplus m_i = H_1(g^{x^{-1}}) \oplus m_i$.

$\mathcal{A}$:

1. Constrói uma cifrassinatura σ' sobre m_i de ID_C para ID_B, onde ID_C é a identidade de um usuário cuja chave privada (Q_C) o adversário tem acesso:
   (a) Escolhe $r' \in_R \mathcal{Z}_r^*$.
   (b) $c' := c^*$;
   (c) $R' := r'P_C$;
   (d) $S' := S^*$;
   (e) $h' := H_3(R', S', c')$;
   (f) $T' := (r' + h')^{-1}Q_C$.
   (g) $\sigma' := (c', R', S', T')$.
2. Consulta (σ', ID_C, ID_B) ao oráculo Designcrypt:
   (a) V' satisfaz a equação de verificação, pois: $V' = e(R' + h'P_C, T') = e(r'P_C + h'P_C, (r' + h')^{-1}Q_C) = e((r' + h')P_C, (r' + h')^{-1}Q_C) = e(P_C, Q_C) = g$.
   (b) $\mathcal{C}$ devolve $m' := H_1(e(S', Q_B)) \oplus c' = H_1(e(S^*, Q_B)) \oplus c^* = H_1(e(x^{-1}P_B, Q_B)) \oplus c^* = H_1(g^{x^{-1}}) \oplus H_1(g^{x^{-1}}) \oplus m_i = m_i$
   (c) $\mathcal{A}$ sabe o valor de i.

Figura 5.10: Ataque sobre a indistinguibilidade do IBSC McCullagh-Barreto

Capítulo 6

Considerações Finais

O paradigma sem certificados (CL-PKC) trouxe vantagens claras em relação ao paradigma explicitamente certificado (PKI), dentre elas a não necessidade de um certificado para a autenticação da chave pública. Uma desvantagem da CL-PKC, em sua concepção original, é o fato de que este modelo atinge somente o nível 2 de Girault, enquanto a PKI atinge o nível 3. Entretanto, com a utilização da técnica de encapsulamento de chaves públicas em CL-PKC, este problema pode ser superado, elevando o paradigma CL-PKC ao nível 3 de Girault, porém com o custo da eliminação de algumas das propriedades únicas presentes no paradigma CL-PKC original. Desde o surgimento do paradigma CL-PKC, diversos tipos de esquemas criptográficos no paradigma PKI foram convertidos para CL-PKC, dentre eles: encriptação, assinatura, cifrassinatura e acordo de chaves. Desde então, diversos esquemas CL-PKC foram propostos para cada uma destas operações.

Tradicionalmente, as primitivas, os algoritmos e os esquemas criptográficos eram tão seguros quanto quão longo fosse o período de tempo pelo qual eles estivessem resistindo à criptanálise. Nas últimas décadas, com um número crescente de propostas de esquemas criptográficos, e o seu uso para a segurança de informações críticas, não tem sido aceitável depositar elevada confiança em esquemas criptográficos sem quaisquer, ou informais, garantias de manutenção de suas propriedades. Para tanto, foram propostos métodos formais para avaliar se o esquema em questão de fato garante as propriedades alegadas, sob hipóteses do nível de poder computacional dos adversários ou até mesmo incondicionalmente. Dentre as metodologias baseadas em hipóteses computacionais, a "Segurança Demonstrável", baseada na redução de um problema computacional ao problema de atacar o esquema, tem se tornado o padrão, e cada vez mais exigida em propostas de esquemas criptográficos na atualidade.

Entretanto, as demonstrações de segurança em "Segurança Demonstrável" tem sofrido diversas críticas, dentre elas, a complexidade destas demonstrações, implicando em grande dificuldade de compreensão por parte, principalmente, dos pesquisadores não especiali-

zados em criptografia teórica, e em demonstrações de segurança que não são revisadas adequadamente. Uma outra crítica é a pouca relevância, em muitos casos, do resultado obtido na demonstração (a complexidade da redução, tanto em tempo quanto em probabilidade) para a prática (determinação de valores adequados para os parâmetros do sistema). Felizmente, como apontado no Capítulo 3, já estão sendo propostas novas diretrizes para o uso desta técnica, de modo a proporcionar maior relevância prática à "Segurança Demonstrável", para que os resultados de segurança obtidos a partir desta técnica tragam maior garantia sobre a segurança real obtida.

A confidencialidade, a autenticidade e a irretratabilidade são as principais propriedades de segurança exigidas em sistemas de informação e, consequentemente, também são requisitos comuns dos sistemas criptográficos que compõem tais sistemas. A necessidade de uma destas propriedades normalmente está associada à necessidade da outra, logo, combinar eficientemente em um único esquema criptográfico as operações que provêem estas propriedades, encriptação para confidencialidade e assinatura para autenticidade e irretratabilidade, é particularmente atraente. Aplicando o conceito de cifrassinatura ao paradigma sem certificados, e definindo um modelo de segurança adequado para cifrassinatura neste novo contexto, obtemos o conceito de cifrassinatura sem certificados (CLSC). Revisamos na Seção 5 os esquemas CLSC presentes na literatura, com foco na segurança.

Conforme pode ser visto na Tabela 5.1, todos os esquemas CLSC estritos, isto é, aqueles que seguem o modelo de geração de chaves de Al-Riyami-Paterson (Seção 4.2.2), e que também possuem verificação pública de cifrassinatura foram eventualmente quebrados. Portanto, podemos enunciar o seguinte problema na área de esquemas CLSC, o qual, segundo o nosso conhecimento, está em aberto: obter um esquema CLSC com geração de chaves no modelo Al-Riyami-Paterson e com verificação pública de cifrassinatura que seja demonstrado seguro.

Referências

[Abd11] Michel Abdalla. Comunicação Pessoal, 2011.

[ABR01] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. In David Naccache, editor, *Topics in Cryptology - CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer Berlin / Heidelberg, 2001.

[ACLD08] D F Aranha, R Castro, J López, and R Dahab. Efficient Certificateless Signcryption. In *VIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSEG 2008)*, pages 257–258, 2008.

[ADR02] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the Security of Joint Signature and Encryption. In Lars R Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer, 2002.

[AL99] Carlisle Adams and Steve Lloyd. *Understanding Public-Key Infrastructure: Concepts, Standards, and Deployment Considerations*. Macmillan Publishing, Indianapolis, Indiana, November 1999.

[AMC+07] Man Ho Au, Yi Mu, Jing Chen, Duncan S. Wong, Joseph K. Liu, and Guomin Yang. Malicious KGC attacks in certificateless cryptography. *ASIAN ACM Symposium on Information, Computer and Communications Security*, 2007.

[ARP03] S Al-Riyami and K Paterson. Certificateless public key cryptography. *Advances in Cryptology-ASIACRYPT 2003*, 2003.

[BBP04] Bellare, Boldyreva, and Palacio. An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem. In *EUROCRYPT: Advances in Cryptology: Proceedings of EUROCRYPT*, 2004.

[BCV09] Daniel R L Brown, Matthew J Campagna, and Scott A Vanstone. Security of ECQV-Certified ECDSA Against Passive Adversaries. Technical report, IACR, 2009.

[BDC+08] P Barreto, AM Deusajute, ES Cruz, GCF Pereira, and RR Silva. Toward efficient certificateless signcryption from (and without) bilinear pairings. In *SBSeg 2008*, pages 115–125, Gramado, 2008.

[BDJR97] Mihir Bellare, Anand Desai, E Jokipii, and Phillip Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *IEEE Foundations of Computer Science*, pages 394–403, 1997.

[Bel98] Mihir Bellare. Practice-Oriented Provable-Security. In *Proceedings of the First International Workshop on Information Security*, ISW '97, pages 221–231, London, UK, 1998. Springer-Verlag.

[BF01] Dan Boneh and Matt Franklin. Identity-Based Encryption from the Weil Pairing. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer Berlin / Heidelberg, 2001.

[BF08] M. Barbosa and P. Farshim. Certificateless Signcryption. 2008.

[BGK05] Johannes Blömer, Jorge Guajardo, and Volker Krummel. Provably Secure Masking of AES. In Helena Handschuh and M Hasan, editors, *Selected Areas in Cryptography*, volume 3357 of *Lecture Notes in Computer Science*, pages 69–83. Springer Berlin / Heidelberg, 2005.

[BGR95] Mihir Bellare, Roch Guérin, and Phillip Rogaway. XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions. In Don Coppersmith, editor, *Advances in Cryptology - Crypto 95*, volume 963 of *Lecture Notes in Computer Science*, pages 15–28. Springer-Verlag, 1995.

[BGV02] D Brown, R Gallant, and S Vanstone. Provably secure implicit certificate schemes. In *Financial Cryptography*, pages 156–165. Springer, Springer, 2002.

[BKR94] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of cipher block chaining. In Yvo G Desmedt, editor, *Advances in Cryptology - CRYPTO 94*, volume 839 of *Lecture Notes in Computer Science*, pages 341–358. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1994.

[BLMQ05] Paulo Barreto, Benoît Libert, Noel McCullagh, and Jean-Jacques Quisquater. Efficient and Provably-Secure Identity-Based Signatures and Signcryption from Bilinear Maps. In Bimal Roy, editor, *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 515–532. Springer Berlin / Heidelberg, 2005.

[Boy03] Xavier Boyen. Multipurpose Identity-Based Signcryption (A Swiss Army Knife for Identity-Based Cryptography). In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 383–399. Springer, 2003.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, 1993.

[BR95] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer Berlin / Heidelberg, 1995.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures-how to sign with RSA and Rabin. In *Proceedings of the 15th annual international conference on Theory and application of cryptographic techniques*, EUROCRYPT'96, pages 399–416, Berlin, Heidelberg, 1996. Springer-Verlag.

[BR97] Mihir Bellare and Phillip Rogaway. Minimizing the use of random oracles in authenticated encryption schemes. In Yongfei Han, Tatsuaki Okamoto, and Sihan Qing, editors, *Information and Communications Security*, volume 1334 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin / Heidelberg, 1997.

[BR04] Mihir Bellare and Phillip Rogaway. The Game-Playing Technique. Cryptology ePrint Archive, Report 2004/331 version 20041219:042446, 2004.

[BR08] Mihir Bellare and Phillip Rogaway. Code-Based Game-Playing Proofs and the Security of Triple Encryption. Cryptology ePrint Archive, Report 2004/331, 2008.

[Bro05a] D Brown. On the provable security of ECDSA. In I Blake, G Seroussi, and N Smart, editors, *Advances in Elliptic Curve Cryptography*. Cambridge University Press, 2005.

[Bro05b] Daniel R L Brown. Generic Groups, Collision Resistance, and ECDSA. *Designs, Codes, and Cryptography*, 35(1):119–152, April 2005.

[BS07] Mihir Bellare and Sarah Shoup. Two-Tier Signatures, Strongly Unforgeable Signatures, and Fiat-Shamir Without Random Oracles. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography - PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, pages 201–216. Springer Berlin / Heidelberg, 2007.

[BSNS05] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Certificateless Public Key Encryption Without Pairing. In Jianying Zhou, Javier Lopez, Robert Deng, and Feng Bao, editors, *Information Security*, volume 3650 of *Lecture Notes in Computer Science*, pages 134–148. Springer Berlin / Heidelberg, 2005.

[BSZ02] Joonsang Baek, Ron Steinfeld, and Yuliang Zheng. Formal Proofs for the Security of Signcryption. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002, Paris, France, February 12-14, 2002, Proceedings*, volume 2274 of *Lecture Notes in Computer Science*, pages 80–98. Springer, 2002.

[BV98] D Boneh and R Venkatesan. Breaking {RSA} May Not Be Equivalent to Factoring. In *Advances in Cryptology – {EUROCRYPT} '98*, pages 59–71, 1998.

[CC05] Zhaohui Cheng and Richard Comley. Efficient Certificateless Public Key Encryption. Cryptology ePrint Archive, Report 2005/012, 2005.

[CDD07] Rafael Dantas Castro, Ricardo Dahab, and Augusto Jun Devegili. Introdução à Segurança Demonstrável. In *Anais do VII Simpósio Brasileiro em SegurancÌ§a da Informação e de Sistemas Computacionais*, pages 103–152, Campinas, 2007.

[Cer04] Certicom. Code and Cipher Vol. 2, No. 2 - Explaining Implicit Certificates. Technical report, Certicom Corp., 2004.

[CGH04] Canetti, Goldreich, and Halevi. The Random Oracle Methodology, Revisited. *JACM: Journal of the ACM*, 51, 2004.

[CJ01] Christophe Clavier and Marc Joye. Universal Exponentiation Algorithm - A First Step towards Provable SPA-Resistance. In Çetin Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 300–308. Springer Berlin / Heidelberg, 2001.

[CKS08] David Cash, Eike Kiltz, and Victor Shoup. The Twin Diffie-Hellman Problem and Applications. In Nigel Smart, editor, *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 127–145. Springer Berlin / Heidelberg, 2008.

[CML05] Chen and Malone-Lee. Improved Identity-Based Signcryption. In *PKC: International Workshop on Practice and Theory in Public Key Cryptography*. LNCS, 2005.

[Cor00] JS Coron. On the exact security of full domain hash. *Advances in Cryptology - CRYPTO 2000*, 2000.

[Cor02] JS Coron. Optimal security proofs for PSS and other signature schemes. *Advances in Cryptology - EUROCRYPT 2002*, 2002.

[Dan08] Rafael Dantas. *Assinaturas de Chave Publica sem Certificados*. Dissertação de mestrado, Universidade Estadual de Campinas, 2008.

[DDN91] D Dolev, C Dwork, and M Naor. Non-malleable cryptography. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, 1991.

[Den08] AW Dent. A survey of certificateless encryption schemes and security models. *International Journal of Information Security*, 2008.

[DH76] W Diffie and M E Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(5):644–654, 1976.

[DZ10] Alexander W Dent and Yuliang Zheng. *Practical signcryption*. Springer-Verlag New York Inc, 2010.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In Michael Wiener, editor, *Advances in*

Cryptology - CRYPTO 99, volume 1666 of *Lecture Notes in Computer Science*, page 79. Springer Berlin / Heidelberg, 1999.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, 1987. Springer-Verlag.

[Gen03] C Gentry. Certificate-based encryption and the certificate revocation problem. *Lecture Notes in Computer Science*, 2003.

[Gir91] Marc Girault. Self-certified public keys. In *Proceedings of the 10th annual international conference on Theory and application of cryptographic techniques*, EUROCRYPT'91, pages 490–497, Berlin, Heidelberg, 1991. Springer-Verlag.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (In)security of the Fiat-Shamir Paradigm. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:102, 2003.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

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

[Gol04] Oded Goldreich. *Foundations of Cryptography: Volume 2: Basic Applications*. Cambridge University Press, pub-CAMBRIDGE:adr, 2004.

[GPS08] Steven D Galbraith, Kenneth G Paterson, and Nigel P Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[Gut02] P Gutmann. PKI: it's not dead, just resting. *Computer*, 2002.

[HFPS02] Russell Housley, Warwick Ford, Tim Polk, and David Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. Internet RFC 3280, April 2002.

[HK07] Shai Halevi and Hugo Krawczyk. Security under Key-Dependent Inputs. Technical report, Cryptology ePrint Archive: Report 2007/315, 2007.

[HK09] Dennis Hofheinz and Eike Kiltz. Practical Chosen Ciphertext Secure Encryption from Factoring. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 313–332. Springer Berlin / Heidelberg, 2009.

[HSMZ05] X Huang, W Susilo, Y Mu, and F Zhang. On the security of certificateless signature schemes from Asiacrypt 2003. *Cryptology and Network Security*, 2005.

[HWL10] Ji Huifang, Han Wenbao, and Zhao Long. Certificateless generalized signcryption. Technical report, Cryptology ePrint Archive: Report 2010/204, 2010.

[HWZD06] Bessie Hu, Duncan Wong, Zhenfeng Zhang, and Xiaotie Deng. Key Replacement Attack Against a Generic Construction of Certificateless Signature. In Lynn Batten and Reihaneh Safavi-Naini, editors, *Information Security and Privacy*, volume 4058 of *Lecture Notes in Computer Science*, pages 235–246. Springer Berlin / Heidelberg, 2006.

[ISO08] International Organization for Standardization. ISO/IEC WD 29150, IT Security Techniques - Signcryption. 2008.

[JN01] Antoine Joux and Kim Nguyen. Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. Technical report, Cryptology ePrint Archive: Report 2001/003, 2001.

[Jut01] Charanjit S Jutla. Encryption Modes with Almost Free Message Integrity. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*, pages 529–544. Springer, 2001.

[JWZ10] Zhengping Jin, Qiaoyan Wen, and Hua Zhang. A supplement to Liu et al.'s certificateless signcryption scheme in the standard model. Cryptology ePrint Archive, Report 2010/252, 2010.

[KL08] J Katz and Y Lindell. *Introduction to modern cryptography*. Chapman & Hall/CRC, 2008.

[KM06] Neal Koblitz and Alfred Menezes. Another Look at Provable Security II. Cryptology ePrint Archive, Report 2006/229, 2006.

[KM07] Neal Koblitz and Alfred J Menezes. Another Look at Provable Security. *Journal of Cryptology*, 20(1):3–37, 2007.

[KPH04] Bo Gyeong Kang, Je Hong Park, and Sang Geun Hahn. A Certificate-Based Signature Scheme. In Tatsuaki Okamoto, editor, *Topics in Cryptology - CT-RSA 2004*, volume 2964 of *Lecture Notes in Computer Science*, page 1991. Springer Berlin / Heidelberg, 2004.

[Kra05] Hugo Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566. Springer Berlin / Heidelberg, 2005.

[KW03] Katz and Wang. Efficiency Improvements for Signature Schemes with Tight Security Reductions. In *SIGSAC: 10th ACM Conference on Computer and Communications Security*. ACM SIGSAC, 2003.

[LAS07] Joseph K Liu, Man Ho Au, and Willy Susilo. Self-Generated-Certificate Public Key Cryptography and certificateless signature/encryption scheme in the standard model: extended abstract. In *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, ASIACCS '07, pages 273–283, New York, NY, USA, 2007. ACM.

[LBG09] Georg Lippold, Colin Boyd, and Juan Gonzalez Nieto. Strongly Secure Certificateless Key Agreement. In *Proceedings of the 3rd International Conference Palo Alto on Pairing-Based Cryptography*, Pairing '09, pages 206–230, Berlin, Heidelberg, 2009. Springer-Verlag.

[LBN10] Georg Lippold, Colin Boyd, and Juan Manuel González Nieto. Efficient certificateless KEM in the standard model. In *Proceedings of the 12th international conference on Information security and cryptology*, ICISC'09, pages 34–46, Berlin, Heidelberg, 2010. Springer-Verlag.

[LBSZ08] Joseph Liu, Joonsang Baek, Willy Susilo, and Jianying Zhou. Certificate-Based Signature Schemes without Pairings or Random Oracles. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, *Information Security*, volume 5222 of *Lecture Notes in Computer Science*, pages 285–297. Springer Berlin / Heidelberg, 2008.

[LHLL10] P. Li, M. He, X. Li, and W. Liu. Efficient and provably secure certificateless signcryption from bilinear pairings. *Journal of Computational Information Systems*, 6(11):3643–3650, 2010.

[LHM+07] Jiguo Li, Xinyi Huang, Yi Mu, Willy Susilo, and Qianhong Wu. Certificate-Based Signature: Security Model and Efficient Construction. In Javier Lopez,

Pierangela Samarati, and Josep Ferrer, editors, *Public Key Infrastructure*, volume 4582 of *Lecture Notes in Computer Science*, pages 110–125. Springer Berlin / Heidelberg, 2007.

[LHZM10] Z. Liu, Y. Hu, X. Zhang, and H. Ma. Certificateless signcryption scheme in the standard model. *Information Sciences*, 180(3):452–464, 2010.

[LK02] Byoungcheon Lee and Kwangjo Kim. Self-certified Signatures. In Alfred Menezes and Palash Sarkar, editors, *Progress in Cryptology - INDOCRYPT 2002*, volume 2551 of *Lecture Notes in Computer Science*, pages 199–214. Springer Berlin / Heidelberg, 2002.

[LLM07] B LaMacchia, K Lauter, and A Mityagin. Stronger security of authenticated key exchange. *Lecture Notes in Computer Science*, 2007.

[LQ04] Benoit Libert and Jean-Jacques Quisquater. The Exact Security of an Identity Based Signature and its Applications. Cryptology ePrint Archive, Report 2004/102, 2004.

[LQ06] Benoît Libert and Jean-Jacques Quisquater. On Constructing Certificateless Cryptosystems from Identity Based Encryption. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography - PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 474–490. Springer Berlin / Heidelberg, 2006.

[LST09] Fagen Li, Masaaki Shirase, and Tsuyoshi Takagi. Certificateless Hybrid Signcryption. 2009.

[MB04] Noel McCullagh and Paulo S L M Barreto. Efficient and Forward-Secure Identity-Based Signcryption. Cryptology ePrint Archive, Report 2004/117, 2004.

[MR99] Mihir Bellare Michel Abdalla and Phillip Rogaway. DHAES: An Encryption Scheme Based on the Diffie-Hellman Problem. Cryptology ePrint Archive, Report 1999/007, 1999.

[MRS88] Silvio Micali, Charles Rackoff, and Bob Sloan. The notion of security for probabilistic cryptosystems. *SIAM J. Comput.*, 17(2):412–426, April 1988.

[MT07] Tarjei K Mandt and Chik How Tan. Certificateless authenticated two-party key agreement protocols. In *Proceedings of the 11th Asian computing science conference on Advances in computer science: secure software and related issues*, ASIAN'06, pages 37–44, Berlin, Heidelberg, 2007. Springer-Verlag.

[MvOV96] Alfred J Menezes, Paul C van Oorschot, and Scott A Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[MW99] Ueli M. Maurer and Stefan Wolf. The Relationship Between Breaking the Diffie–Hellman Protocol and Computing Discrete Logarithms. *SIAM Journal on Computing*, 28(5):1689, 1999.

[MZZ10] Songqin Miao, Futai Zhang, and Lei Zhang. *Cryptanalysis of a Certificateless Multi-receiver Signcryption Scheme*. IEEE, November 2010.

[OO98] Kazuo Ohta and Tatsuaki Okamoto. On Concrete Security Treatment of Signatures Derived from Identification. In *Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, pages 354–369, London, UK, 1998. Springer-Verlag.

[OP01] Tatsuaki Okamoto and David Pointcheval. The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In Kwangjo Kim, editor, *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 104–118. Springer Berlin / Heidelberg, 2001.

[Poi05] David Pointcheval. Provable Security for Public Key Schemes. In Dario Catalano, Ronald Cramer, Giovanni Crescenzo, Ivan Darmgå rd, David Pointcheval, and Tsuyoshi Takagi, editors, *Contemporary Cryptology*, Advanced Courses in Mathematics - CRM Barcelona, pages 133–190. Birkhäuser Basel, 2005.

[PS96] D Pointcheval and J Stern. Security proofs for signature schemes. *Advances in Cryptology - EUROCRYPT'96*, 1996.

[PS00] D Pointcheval and J Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 2000.

[RDP08] Matthieu Rivain, Emmanuelle Dottax, and Emmanuel Prouff. Block Ciphers Implementations Provably Secure Against Second Order Side Channel Analysis. In Kaisa Nyberg, editor, *Fast Software Encryption*, volume 5086 of *Lecture Notes in Computer Science*, pages 127–143. Springer Berlin / Heidelberg, 2008.

[Sae03] S Saeednia. A note on Girault's self-certified model. *Information Processing Letters*, 86(6):323–327, 2003.

[Sch91] C P Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[Sha49] Claude Elwood Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

[Sha85] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53. Springer-Verlag New York, Inc., 1985.

[Sha07] Zuhua Shao. Self-certified signature scheme from pairings. *Journal of Systems and Software*, 80(3):388–395, March 2007.

[Sho06] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Technical report, Cryptology ePrint Archive: Report 2004/332, 2006.

[Sma10] Nigel Smart. *Introduction to Cryptography*. 3rd edition, 2010.

[Str08] R Struik. SEC 4: Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV). Technical report, Certicom Corp., 2008.

[SVR09] S. Sharmila Deva Selvi, S. Sree Vivek, and C. Pandu Rangan. A note on the Certificateless Multi-receiver Signcryption Scheme. 2009.

[SVR10a] S Selvi, S Vivek, and C Rangan. Certificateless KEM and Hybrid Signcryption Schemes Revisited. In Jin Kwak, Robert Deng, Yoojae Won, and Guilin Wang, editors, *Information Security, Practice and Experience*, volume 6047 of *Lecture Notes in Computer Science*, pages 294–307. Springer Berlin / Heidelberg, 2010.

[SVR10b] S.Sharmila Deva Selvi, S.Sree Vivek, and C.Pandu Rangan. Cryptanalysis of Certificateless Signcryption Schemes and an Efficient Construction Without Pairing. Technical report, Cryptology ePrint Archive, Report 2009/298, 2010.

[SVR10c] S.Sharmila Deva Selvi, S.Sree Vivek, and C.Pandu Rangan. Security Weaknesses in Two Certificateless Signcryption Schemes. Technical report, Cryptology ePrint Archive: Report 2010/092, 2010.

[SVR11] S Selvi, S Vivek, and C Rangan. Cryptanalysis of Certificateless Signcryption Schemes and an Efficient Construction without Pairing. In Feng Bao, Moti Yung, Dongdai Lin, and Jiwu Jing, editors, *Information Security and Cryptology*, volume 6151 of *Lecture Notes in Computer Science*, pages 75–92. Springer Berlin / Heidelberg, 2011.

[SVSR08] S Selvi, S Vivek, Deepanshu Shukla, and Pandu Rangan Chandrasekaran. Efficient and Provably Secure Certificateless Multi-receiver Signcryption. In Joonsang Baek, Feng Bao, Kefei Chen, and Xuejia Lai, editors, *Provable Security*, volume 5324 of *Lecture Notes in Computer Science*, pages 52–67. Springer Berlin / Heidelberg, 2008.

[Swa08] CM Swanson. *Security in key agreement: two-party certificateless schemes*. PhD thesis, Waterloo, 2008.

[SZ00] Steinfeld and Zheng. A Signcryption Scheme Based on Integer Factorization. In *ISW: International Workshop on Information Security, LNCS*, 2000.

[Ver26] G S Vernam. Cipher printing telegraph systems for secret wire and radio telegraphic communications. *American Institute of Electrical Engineers, Transactions of the*, 45:295–301, 1926.

[WC08] Chenhuang Wu and Zhixiong Chen. A New Efficient Certificateless Signcryption Scheme. In *Information Science and Engineering, 2008. ISISE '08. International Symposium on*, volume 1, pages 661–664, 2008.

[WSI03] Watanabe, Shikata, and Imai. Equivalence between Semantic Security and Indistinguishability against Chosen Ciphertext Attacks. In *PKC: International Workshop on Practice and Theory in Public Key Cryptography*. LNCS, 2003.

[WYD+11] Jian Weng, Guoxiang Yao, Robert H Deng, Min-Rong Chen, and Xiangxue Li. Cryptanalysis of a certificateless signcryption scheme in the standard model. *Information Sciences*, 181(3):661–667, 2011.

[WZM07] Ling-ling Wang, Guo-yin Zhang, and Chun-guang Ma. Verifiable certificateless ring signcryption scheme based on bilinear pairings. *Journal of Computer Applications*, 2007.

[XZ10a] W. Xie and Z. Zhang. Certificateless Signcryption without Pairing. Technical report, Cryptology ePrint Archive: Report 2009/578, 2010.

[XZ10b] W. Xie and Z. Zhang. Efficient and provably secure certificateless signcryption from bilinear maps. In *Wireless Communications, Networking and Information Security (WCNIS), 2010 IEEE International Conference on*, pages 558–562. IEEE, 2010.

[Yao82] Andrew C Yao. Theory and application of trapdoor functions. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:80–91, 1982.

[ZG09] Jianhong Zhang and Qin Geng. Cryptoanalysis of Two Signcryption Schemes. *Information Assurance and Security, International Symposium on*, 2:65–68, 2009.

[Zhe97a] Y Zheng. Digital signcryption or how to achieve cost (signature & encryption)¡¡ cost (signature)+ cost (encryption). *Advances in Cryptology - CRYPTO'97*, 1997.

[Zhe97b] Y Zheng. Digital signcryption or how to achieve cost (signature & encryption)¡¡ cost (signature)+ cost (encryption). *Full Version. Available from http://www.sis.uncc.edu/˜yzheng/papers/*, 1997.

[ZWXF06] Z Zhang, D Wong, J Xu, and D Feng. Certificateless public-key signature: Security model and efficient construction. *Applied Cryptography and Network ...*, 2006.

Glossário

AKACP Authenticated Key Agreement With Key Confirmation. 69

AKAP Authenticated Key Agreement Protocol. 69

CA Certification Authority. 1

CB-PKC Certificate-based Public Key Cryptography. 15

CBE Certificate-based Encryption. 15

CBS Certificate-based Signature. 15

CL-PKC Certificateless Public Key Cryptography. 12

CLE Certificateless Encryption. 15

CLS Certificateless Signature. 15

CMA Chosen Message Attack. 31

CRL Certificate Revocation List. 16

EUF Existential Unforgeability. 31

EUF-CMA Existential Unforgeability Against Chosen Message Attacks. 32

Gap Lacunar. 48

IB-PKC Identity-based Public Key Cryptography. 5

IBC Identity-based Cryptography. 5

IBE Identity-based Encryption. 5

IBS Identity-based Signature. 5

IBSC Identity-Based Signcryption. 85

IND-CCA Indistinguishability Against Adaptively Chosen Ciphertext Attacks. 29

IND-CCA2 Indistinguishability Against Full Adaptively Chosen Ciphertext Attacks. 29

IND-CPA Indistinguishability Against Adaptively Chosen Plaintext Attacks. 29

insiders Atacantes internos. Atacantes que são usuários do sistema. 88

KACP Key Agreement Protocol with Key Confirmation. 69

KAP Key Agreement Protocol. 68

KCA Known-Ciphertext Attacks ou Ciphertext-only Attacks. 29

KGC Key Generation Center. 6

KPA Known-Plaintext Attacks. 29

LL-keys Long-lived keys. 68

MAC Message Authentication Code. 33

OCSP Open Certificate Status Protocol. 16

padding Enchimento. 42

PIN Personal identification number. É uma senha numérica comumente usada na autenticação do cliente para o banco. 69

PKE Public Key Encryption. Encriptação no modelo PKI. 80

PKS Public Key Signature. Assinatura no modelo PKI. 80

PKSC Public Key Signcryption. Cifrassinatura de Chave Pública Explicitamente Certificada. 82, 85

PPT Probabilistic Polynomial-Time. 26

ROM Random Oracle Model. 34

SC-PKC Self-Certified Public Keys. 8

sEUF Strong Existential Unforgeability. 32

SKDP Session Key Distribution Protocol. 68

TA Trusted Authority. 6

tripartite Trilateral. De três partes. 68

TTP Trusted Third Party. 33

two-party Bilateral. De duas partes. 68