Criptografia de chave pública sem certificado

Denise Hideko Goya

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

Programa: Ciência da Computação

Orientador: Prof. Dr. Routo Terada

Durante o desenvolvimento deste trabalho a autora recebeu auxílio financeiro da FAPESP

São Paulo, Dezembro de 2011

Criptografia de chave pública
sem certificado

Esta tese contém as correções e alterações
sugeridas pela Comissão Julgadora durante a defesa
realizada por Denise Hideko Goya em 16/12/2011.
O original encontra-se disponível no Instituto de
Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

- Prof. Dr. Routo Terada (orientador) - IME-USP
- Prof. Dr. Paulo Sérgio Licciardi Messeder Barreto - EP-USP
- Prof. Dr. Julio César López Hernández - UNICAMP
- Prof. Dr. Jeroen Antonius Maria van de Graaf - UnB
- Prof. Dr. Carlos Alberto Maziero - UTFPR

Agradecimentos

A todos que conviveram comigo durante o desenvolvimento desta tese, dedico meus sinceros agradecimentos.

Ao professor Routo Terada, pela orientação, paciência e confiança.

Aos membros da banca de qualificação e da comissão julgadora (professores Ricardo Dahab, Paulo Barreto, Julio López, Jeroen van de Graaf, Carlos Alberto Maziero), pelos questionamentos, correções, sugestões e comentários preciosíssimos.

Aos amigos e membros do Laboratório de Segurança de Dados (LSD) e do Laboratório de Arquitetura e Redes de Computadores (LARC), pelas discussões e companheirismo, em especial aos parceiros que contribuíram com os trabalhos de pesquisa: Cleber Okida, Dionathan Nakamura e Vilc Rufino.

Aos demais amigos, que tornaram mais suave o percurso até o cumprimento desta etapa de minha vida.

Ao professor Kenny Paterson e ao pesquisador Georg Lippold, pelos comentários e observações aos primeiros resultados deste trabalho.

A todos professores e funcionários do Instituto de Matemática e Estatística e da Universidade de São Paulo que colaboraram com minha formação e viabilizaram a produção desta obra.

À FAPESP, pelo auxílio financeiro registrado sob o processo número 2008/06189-0.

E, principalmente, à minha família, pelo apoio, compreensão e afeto. Em particular, aos meus filhos, Aline e Ivan, ao meu esposo Roberto e a todos que tão bem cuidaram deles na minha ausência (vó Maria, professoras e seus colaboradores).

A todos, muito obrigada!

Resumo

A criptografia de chave pública sem certificado (*certificateless*) é uma alternativa ao modelo convencional de criptografia assimétrica, pois a autenticação da chave pública ocorre implicitamente durante a execução dos protocolos, sem a necessidade de gerenciamento e distribuição de certificados digitais. Potencialmente reduz custos computacionais e o nível de segurança alcançado é maior quando comparado ao modelo baseado em identidade.

Nesta tese de doutorado, modelos formais de segurança para acordo de chave com autenticação sem certificado são aprimorados visando dois objetivos paralelos: (1) aumentar o nível de confiança que usuários podem depositar na autoridade geradora de chaves secretas parciais e (2) viabilizar protocolos que sejam eficientes computacionalmente e com propriedades de segurança relevantes, dentre as quais se inclui resistência a ataques de adversários que têm total controle do canal de comunicação e que podem substituir chaves públicas de usuários por valores arbitrários. Para atestar que as melhorias efetuadas são praticáveis e possibilitam que os objetivos sejam alcançados, novos protocolos são propostos para o caso que envolve dois participantes na comunicação. Os protocolos são provados seguros, usando-se técnica de redução de problemas computacionais.

Palavras-chave: Criptografia de chave pública sem certificado, segurança demonstrável, modelos de segurança, acordo de chave com autenticação, emparelhamento bilinear, criptografia sobre curva elíptica, segurança de sistemas de informação.

Abstract

Certificateless public key cryptography is an alternative model to traditional asymmetric key cryptography, because the public key authentication occurs implicitly during a protocol run, with no need of digital certificates management and distribution. It has the potential to reduce computing costs, and it allows a higher security level than the one in the identity-based model.

In this PhD thesis, formal security models for certificateless authenticated key agreement are improved with two independent objectives: (1) to increase the trust level for the partial secret key generating authority on which users rely, and (2) to enable computationally efficient protocols, with significant security properties, such as resistance against attacks from adversaries with full control of the communication channel, and from adversaries who are able to replace users' public keys by any chosen value. In order to demonstrate that these improvements made are feasible and achieve the objectives, new protocols are proposed in the two-party case. These protocols are proved secure by using reduction techniques for provable security.

Keywords: Certificateless public key cryptography, provable security, security models, authenticated key agreement, bilinear pairing, elliptic curve cryptography, information systems security.

Sumário

Lista de Abreviaturas

AKA	Acordo de chave com autenticação (*Authenticated Key Agreement*)
AP	Formulação de chave pública no modelo sem certificado devida a Al-Riyami e Paterson (Al-Riyami e Paterson, 2003)
BDH	Problema Diffie-Hellman bilinear
BSS	Formulação de chave pública no modelo sem certificado devida a Baek, Safavi-Naini e Susilo (Baek *et al.*, 2005)
CBE	Cifragem baseada em certificado (*Certificate-Based Encryption*)
CCA	Ataque de texto cifrado escolhido (*Chosen Ciphertext Attack*)
CL-AKA	Acordo de chave com autenticação sem certificado (*Certificateless Authenticated Key Agreement*)
CL-KEM	Mecanismo de encapsulamento de chave sem certificado (*Certificateless Key-Encapsulation Mechanism*)
CLE	Cifragem sem certificado (*Certificateless Encryption*)
CLS	Assinatura sem certificado (*Certificateless Signature*)
DBDH	Problema de decisão Diffie-Hellman bilinear
DoD	Negação de decifração (*Denial of Decryption*)
FS	Segurança no futuro: propriedade de protocolo de acordo de chave com autenticação (*Forward Secrecy*)
Gap-BDH	Problema Diffie-Hellman bilinear lacunar
IBE	Cifragem baseada em identidade (*Identity-Based Encryption*)
ICP	Infraestrutura de chaves públicas
ID-AKA	Acordo de chave com autenticação baseado em identidade (*Identity-Based Authenticated Key Agreement*)
HMQV	Variante do protocolo MQV devido a Krawczyk (Krawczyk, 2005)
KCI	Tipo de ataque a protocolo de acordo de chave com autenticação por personificação pelo comprometimento de chave secreta (*Key-Compromise Impersonation*)
KEM	Mecanismo de encapsulamento de chave (*Key-Encapsulation Mechanism*)
KGC	Centro de geração de chaves (*Key Generating Centre*)
LBG	Modelo de segurança para CL-AKA devido a Lippold, Boyd e González Nieto (Lippold *et al.*, 2009)
LK	Formulação de chave pública no modelo sem certificado devida a Lai e Kou (Lai e Kou, 2007)

MQV	Protocolo de acordo de chave com autenticação devido a Menezes, Qu e Vanstone (Menezes *et al.*, 1995)
OTS	Método de assinatura em que a chave pode ser usada para assinar uma única mensagem (*One-Time Signature*)
PFS	Segurança no futuro completa: propriedade de protocolo de acordo de chave com autenticação (*Perfect Forward Secrecy*)
PKE	Cifragem sob chave pública (*Public Key Encryption*)
PKI	Infraestrutura de chaves públicas, ou ICP (*Public Key Infrastructure*)
SJ	Modelo de segurança para CL-AKA devido a Swanson e Jao (Swanson e Jao, 2009)
UKS	Tipo de ataque a protocolo de acordo de chave com autenticação pelo compartilhamento desconhecido de chave (*Unknown Key-Share*)
wPFS	Segurança no futuro completa-fraca: propriedade de protocolo de acordo de chave com autenticação (*Weak Perfect Forward Secrecy*)

Lista de Símbolos e Notação

$\mathbb{Z}_q$	Conjunto dos inteiros módulo q
$\mathbb{Z}_q^*$	$\mathbb{Z}_q - \{0\}$
$\xleftarrow{\$}$	sorteio uniforme e atribuição à variável
$\{0,1\}^k$	cadeia binária de comprimento k
$\{0,1\}^*$	cadeia binária de comprimento arbitrário
$=_p$	equivalência polinomial de problemas computacionais
$\leq_p$	redução polinomial do problema computacional à esquerda para o da direita
$\oplus$	operador de ou exclusivo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Os objetivos deste capítulo são contextualizar e descrever motivações e objetivos para a realização desta tese, apresentar as principais contribuições e descrever a estrutura geral do documento.

1.1 Contexto e Motivação

A criptografia de chave pública surgiu com a busca por soluções para dois problemas intrinsecamente relacionados com o modelo de criptografia simétrica: o de distribuir uma chave secreta e o de autenticar alguém, com garantia de irretratabilidade. Ambos problemas foram tratados por Diffie e Hellman (1976), quando nasceu um novo paradigma para a criptografia, a de chave pública. Nesse modelo, também chamado de assimétrico, todo usuário possui um par de chaves, uma pública e outra secreta. A chave pública pode ser divulgada por meio de um canal público e seu valor é matematicamente relacionado com o valor da chave secreta correspondente, que convencionalmente fica sob guarda exclusiva de seu dono. Enquanto uma chave é usada para realizar uma operação criptográfica, a outra é requerida para que tal operação possa ser invertida.

É natural que um novo paradigma traga consigo novos problemas. Com a criptografia de chave pública, não foi diferente. Um problema central nesse modelo é o de legitimação da chave pública. Como garantir que uma chave pública pertence, de fato, a alguém? No modelo convencional, o valor da chave pública não fornece indícios de quem seja seu proprietário. No entanto, os criptossistemas assimétricos funcionam sob a premissa de que a chave pública é certificada, ou seja, é vinculada com seu dono. Uma solução que se tornou prática comum é a de implantação de uma infraestrutura de chaves públicas (ICP, ou PKI – *Public Key Infrastructure*), cuja marca são os certificados digitais de chave pública. Tal solução, entretanto, embute dificuldades, como as resumidas abaixo:

- processos complexos de implantação e manutenção da infraestrutura;
- custos de emissão, distribuição e armazenamento de certificados;
- custos para recuperar e validar certificados;
- dificuldades com revogação de certificados.

Há pelo menos dois caminhos para minimizar essas dificuldades: modificar o modelo de funcionamento da infraestrutura, que não é escopo deste trabalho, e modificar o próprio modelo de

criptografia de chave pública. Esta última abordagem dá origem ao que chamamos de modelos alternativos.

O modelo de criptografia de chave pública baseado em identidade, proposto por Shamir (1984), dispensa a necessidade de certificados digitais e de uma ICP que os gerencie, uma vez que a chave pública de um usuário é sua própria identificação no sistema (ou seja, dados pessoais, como nome, número do CPF, do telefone celular, endereço IP ou de correio eletrônico). Uma característica inerente ao modelo de criptografia de chave pública baseado em identidade é a de custódia de chaves, em que a autoridade do sistema conhece as chaves secretas de todos seus usuários. A custódia de chaves é indesejável em vários tipos de aplicações; em particular, impossibilita a garantia de irretratabilidade, pois o usuário não é o único conhecedor de sua chave secreta.

Al-Riyami e Paterson (2003) introduziram o paradigma de criptografia de chave pública sem certificado, que une propriedades do modelo convencional com o baseado em identidade. Nesse paradigma, também conhecido por *certificateless*, ICP e certificados digitais se fazem desnecessários, pois a identidade de um usuário fica parcialmente relacionada com a chave secreta. E não há custódia de chaves, pois o usuário é o único conhecedor de sua chave criptográfica secreta. Ademais, para uso interno a uma dada instituição (uma empresa privada ou órgão do governo) este paradigma pode ser apropriado, uma vez que a entidade geradora das chaves (chamada *Key Generating Centre*, KGC) pode ser um membro idôneo desta instituição. Entretanto, justamente pelo fato da chave pública não precisar de certificados, há que se levar em consideração que as chaves públicas podem ser arbitrariamente substituídas por um adversário.

Segurança de Protocolos Criptográficos

O desenvolvimento de protocolos criptográficos de chave pública é baseado em modelos de segurança que descrevem ações de um adversário contra um criptossistema e um objetivo de comprometimento de propriedades do protocolo que, em suma, define o que vem a ser segurança formal para aquele tipo de protocolo.

No modelo de chave pública sem certificado, a definição dos modelos de segurança tem se mostrado complexa, dada a necessidade de se lidar com um adversário que altera o valor da chave pública de usuários à sua escolha. Dent (2008) apresenta uma análise dos modelos de segurança para esquemas de cifragem no modelo de chave pública sem certificado e relaciona uma grande diversidade de modelos e adversários com variados graus de poder.

Considerando-se que esquemas de cifragem garantem autenticidade apenas do usuário de destino, eleva-se consideravelmente a complexidade dos modelos de segurança para protocolos que visam garantir a autenticidade mútua dos usuários (autenticação na origem e no destino das mensagens). Uma classe particular de protocolos com autenticidade mútua é a de estabelecimento de chave secreta com autenticação. No trabalho de Swanson e Jao (2009), relata-se que todos os protocolos para acordo de chave secreta com autenticação sob chave pública sem certificado (*Certificateless Authenticated Key Agreement*, ou CL-AKA) eram inseguros até então e se propõe novo modelo de segurança.

Vale notar que primitivas de estabelecimento de chave secreta são peças fundamentais no desenvolvimento de soluções de segurança de sistemas computacionais, pois quase sempre são a base dessas soluções. No entanto, os protocolos CL-AKA conhecidos que foram mostrados seguros são pouco eficientes e seguem modelos de segurança mais complexos que o de Swanson e Jao, como

em (Lippold *et al.*, 2009) e (Yang e Tan, 2011a), ou satisfazem modelos de adversários mais fracos, como em (Zhang *et al.*, 2010) e (Lippold e González Nieto, 2010).

1.2 Objetivos

O objetivo primário de nosso trabalho é o desenvolvimento de protocolos em que autenticidade é um requisito, dentro do modelo de chave pública sem certificado. Algumas questões que buscamos responder são:

(Q.1) Como se caracterizam os modelos alternativos de criptografia de chave pública, quais são suas limitações e como eles se relacionam entre si?

(Q.2) Quais são as propriedades e restrições dos diferentes modelos de segurança para protocolos de acordo de chave secreta com autenticação sob chave pública sem certificado (CL-AKA)? Em que condições cada um é indicado?

(Q.3) Até que ponto os modelos mais fortes para CL-AKA podem ser enfraquecidos para possibilitarem melhor desempenho computacional, preservando-se importantes propriedades de segurança?

(Q.4) Existem ataques de segurança previstos em esquemas de cifragem ou assinatura que não foram estudados em CL-AKA? Em caso positivo, há como evitá-los?

(Q.5) Os protocolos mostrados seguros são viáveis na prática?

As questões acima não tinham respostas completas registradas na literatura e, ao respondê-las ao longo do desenvolvimento desta tese, geramos alguns resultados inéditos.

1.3 Resumo da Metodologia

Apresentamos a seguir, um breve resumo da metodologia empregada no decorrer da tese:

Modelos alternativos de chave pública: quatro modelos alternativos de chave pública são estudados, com o objetivo de se compreender em que condições eles ajudam a reduzir dificuldades conhecidas relacionadas a infraestruturas de chaves públicas (ICPs) convencionais. São analisados os modelos baseado em identidade (*identity-based*) de Shamir (1984), autocertificado (*self-certified*) de Girault (1991), sem certificados (*certificateless*) de Al-Riyami e Paterson (2003) e baseado em certificado (*certificate-based*) de Gentry (2003). A partir da análise conceitual de cada modelo, são expostas vantagens e desvantagens, discussões sobre possíveis aplicações e contextos de uso. Os detalhes são descritos no Capítulo 2.

Segurança demonstrável: denomina-se segurança demonstrável a metodologia para o estabelecimento de modelos de segurança e respectiva prova formal de que um protocolo criptográfico é seguro. Resumidamente, uma prova de segurança formal parte da existência de um problema computacional difícil (para o qual não se conhece solução de tempo polinomial) e mostra que a existência de um adversário com poderes descritos no modelo de segurança e que seja capaz de atingir o objetivo em tempo polinomial implica na existência de um algoritmo de

tempo polinomial para a solução do problema suposto difícil. Goldwasser e Micali (1984) estabelecem as bases para a metodologia de segurança demonstrável. Bellare e Rogaway (1993) propõem o modelo do oráculo aleatório para proporcionar provas mais simples e protocolos computacionalmente mais eficientes. Definições acerca desse assunto são dadas no Capítulo 3. A análise de segurança das propostas apresentadas nesta tese seguem esses métodos.

Acordo de chave secreta com autenticação: Diffie e Hellman (1976) apresentam o primeiro protocolo de acordo de chave em que dois participantes comunicando-se por meio de um canal público efetuam o cálculo de um segredo em comum. O protocolo Diffie-Hellman original tem a desvantagem de ser determinístico e não garantir autenticidade dos participantes, mas é suficientemente simples para ilustrar essa classe de protocolo; na Figura 1.1, há uma representação dele. Bellare e Rogaway (1994) formalizam modelo de segurança para acordo de chave com autenticação e estabelecem o núcleo de todos os trabalhos que o sucedem; refinamentos sucessivos de modelos de segurança para acordo de chave com autenticação surgem na literatura, dentre os quais citamos (Canetti e Krawczyk, 2001), (Krawczyk, 2005) e (LaMacchia *et al.*, 2007) que servem de base para nosso trabalho. Protocolos de acordo de chave com autenticação são mais complexos que o Diffie-Hellman; na Figura 1.2, é ilustrado o HMQV de Krawczyk (2005) que requer uma infraestrutura de chave pública de apoio. Tanto o Diffie-Hellman quanto o HMQV partem da premissa de intratabilidade do problema Diffie-Hellman computacional (CDH), em que é suposto intratável calcular g^{ab} dados g^a e g^b.

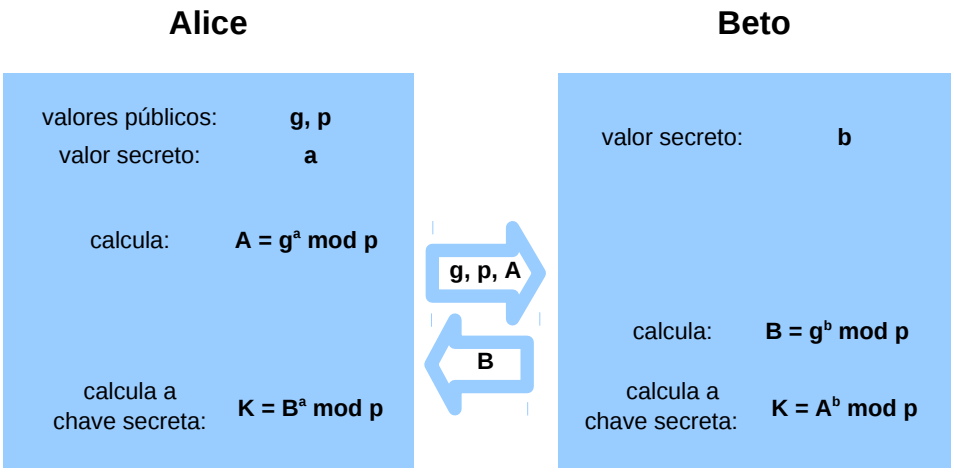


Figura 1.1: *Protocolo de acordo de chave sem autenticação de Diffie-Hellman: seja g um gerador de um grupo finito de ordem prima p; Alice e Beto compartilham o segredo $K = A^b \bmod p = B^a \bmod p$, depois de realizarem a troca de mensagens via canal público e sem vazarem informação a respeito dos valores secretos*

Acordo de chave secreta com autenticação sem certificado: no cenário sem certificado, os modelos de segurança precisam ser ajustados para tratar o corrompimento dos segredos parciais de cada participante e, em particular, para prever ações de um adversário relacionadas com a substituição de chaves públicas. São estudados os modelos de Swanson e Jao (2009), Lippold *et al.* (2009), Zhang *et al.* (2010), Lippold e González Nieto (2010), Yang e Tan (2011a)

(além de outros que mostraram ser pobres ou inconsistentes). A partir da análise desses modelos de segurança, propomos aprimoramentos que são discutidos no Capítulo 4.

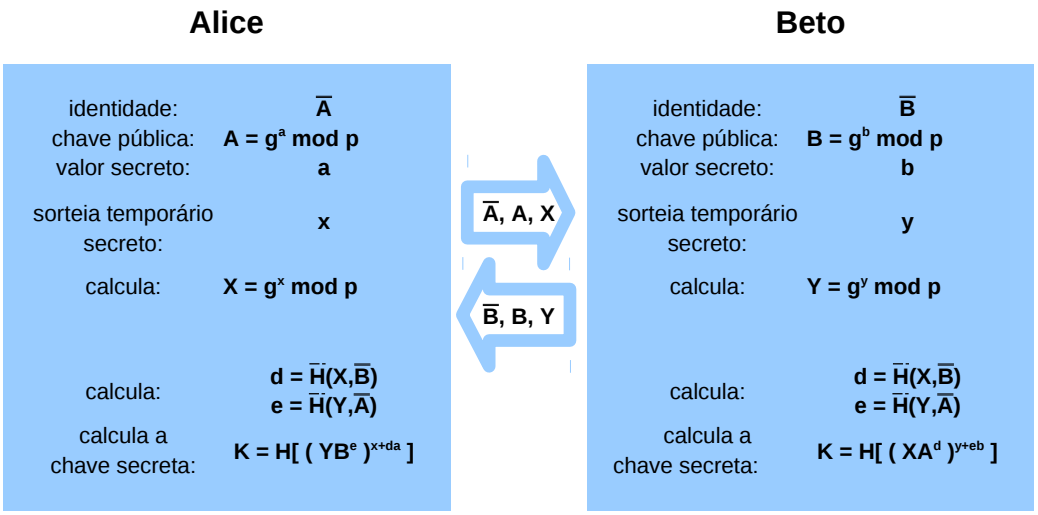


Figura 1.2: *Protocolo HMQV de acordo de chave com autenticação: seja g um gerador de um grupo finito de ordem prima p e sejam H e $\overline{H}$ duas funções de hash; Alice e Beto possuem certificados digitais (que precisam ser verificados por seus pares) e compartilham o segredo K, depois de realizarem os cálculos e a troca de mensagens*

Emparelhamento bilinear: de modo informal, diz-se que um emparelhamento bilinear, ou *bilinear pairing*, é uma função que mapeia dois elementos de grupos albébricos em outro grupo, com a propriedade de bilinearidade. Tem sido bastante explorado em criptografia depois do trabalho de Sakai *et al.* (2000) e é usado também nos protocolos apresentados nesta tese. Informalmente, dados dois elementos aP e bQ de grupos onde o problema CDH (Diffie-Hellman computacional) é suposto difícil, um emparelhamento bilinear e recai sobre outro grupo em que CDH também é difícil e é tal que vale: $e(aP, bQ) = e(abP, Q) = e(P, abQ) = e(P, Q)^{ab}$. Tipicamente, os valores a e b são tornados secretos enquanto aP e bQ são públicos, com base na intratabilidade do problema do logaritmo discreto (isto é, é suposto difícil o cálculo de a a partir de aP). Também é suposto difícil o problema Diffie-Hellman bilinear (BDH) em que dados aP, bP, cP, deve-se calcular $e(P, P)^{abc}$. Esse tópico é retomado no Capítulo 3.

Testes: todos protocolos propostos nesta tese foram codificados em linguagem C, testados e comparados com outros existentes na literatura. O objetivo da realização desses testes foi o de confirmar a viabilidade e o funcionamento, além de avaliar a eficiência frente a outras soluções. Os resultados obtidos são colocados no Capítulo 5.

1.4 Contribuições

Durante a busca por respostas às questões citadas na Seção 1.2, produzimos os itens a seguir, que consideramos serem as principais contribuições de nosso trabalho:

- Levantamento da literatura sobre os modelos alternativos em criptografia de chave pública, que é replicado no Capítulo 2, para responder a questão (Q.1) (Goya *et al.*, 2009a);
- Protocolos de acordo de chave com autenticação sem certificado, mostrados seguros em modelos de segurança variados, objetivando otimização em tempo computacional, tratando as questões (Q.2) e (Q.3) (Goya *et al.*, 2010a), (Goya *et al.*, 2010b), (Goya *et al.*, 2012c); esses protocolos são descritos no Capítulo 5;
- Análise de modelos de segurança para acordo de chave com autenticação sem certificado, extensão deles para elevação do nível de confiança que usuários podem depositar na autoridade geradora de chaves parciais, e proposta de protocolo, para as questões (Q.2) e (Q.4) (Goya *et al.*, 2011); esse assunto é abordado no Capítulo 4;
- Fortalecimento de modelo de segurança para acordo de chave com autenticação sem certificado, visando adversário com poder mais realista e proposta de protocolo mais eficiente, exposto no Capítulo 4, para a questão (Q.3) (Goya *et al.*, 2012c);
- Protocolo de acordo de chave com autenticação sem certificado seguro no modelo fortalecido e com extensão para caso da autoridade mal intencionada, refinando soluções para questões (Q.3) e (Q.4) (Goya *et al.*, 2012b); esses protocolos são apresentados no Capítulo 5;
- Prova de conceito para estabelecimento de chave de sessão entre servidor e dispositivos móveis, no modelo sem certificado, respondendo afirmativamente a questão (Q.5) (Okida *et al.*, 2012), (Goya *et al.*, 2010b); a análise e testes dos protocolos estão no Capítulo 5.

1.5 Organização do Trabalho

O restante desta tese está organizado da seguinte forma:

Capítulo 2: paradigmas de criptografia de chave pública são discutidos. Conceituamos e descrevemos as propriedades de quatro modelos: baseado em identidade (*identity-based*), autocertificado (*self-certified*), sem certificados (*certificateless*) e baseado em certificado (*certificate-based*), com o intuito de esclarecer as diferenças entre cada um e as restrições para suas aplicações;

Capítulo 3: são apresentados fundamentos teóricos que são requisitos para a leitura dos capítulos seguintes. Também são dadas formalizações sobre o modelo sem certificado, que é adotado como principal objeto de estudo;

Capítulo 4: após uma breve exposição de conceitos sobre protocolos de acordo de chave, é definido o tipo com autenticação sem certificado e, na sequência, discutem-se os modelos de segurança para protocolos de acordo de chave com autenticação sem certificado (CL-AKA);

Capítulo 5: alguns protocolos do tipo CL-AKA são revisados e outros novos são propostos, com vistas a otimizar eficiência ou segurança de protocolos predecessores ou com a finalidade de mostrar a viabilidade dos novos modelos de segurança propostos; nesse capítulo se encontram os resultados de testes experimentais realizados;

Capítulo 6: são apresentadas provas de segurança dos protocolos propostos, que mostram que eles satisfazem as propriedades descritas nos respectivos modelos de segurança;

Capítulo 7: é onde as nossas conclusões são traçadas e onde sugerimos alguns trabalhos futuros.

Capítulo 2

Modelos Alternativos de Criptografia de Chave Pública

Os objetivos deste capítulo são descrever variantes do modelo de criptografia assimétrica que visam a redução das dificuldades impostas por uma infraestrutura de chave pública (ICP) convencional. São vistas as conceituações, vantagens e desvantagens dos modelos baseado em identidade (*identity-based*), autocertificado (*self-certified*), sem certificados (*certificateless*) e baseado em certificado (*certificate-based*).

2.1 Criptografia de Chave Pública

Em criptografia de chave pública, todo usuário tem um par de chaves (s, P). Por conveniência, passaremos a nos referir ao valor P como *chave pública*, enquanto o valor s faz o papel de *chave secreta* (privada). Graficamente, esses dois tipos de chave são representados como se vê na Figura 2.1.

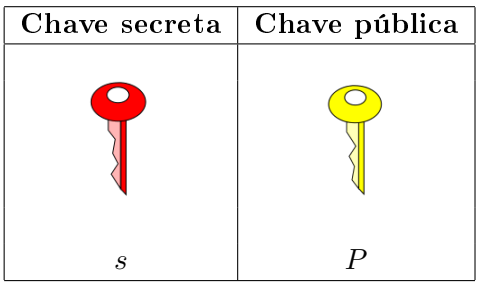


Figura 2.1: *Par de chaves no modelo de criptografia de chave pública*

A chave pública P está matematicamente relacionada com a chave secreta s, de forma que:

- P é calculada a partir de s, mas
- a partir do valor de P, é computacionalmente inviável descobrir s.

Desse modo, podemos escrever genericamente que $P = f(s)$, onde f é uma função injetora. Às vezes, a função f inclui parâmetros do sistema ou outros dados. Por exemplo, na cifragem de ElGamal, $P = g^s$, onde g é um parâmetro do sistema. Eventualmente, f pode parecer complexa

e, em implementações concretas, pode até ser codificada por um algoritmo probabilístico ou s ser calculada a partir de P. Mas, por simplicidade, basta pensar que a chave pública é função da secreta.

Ora, se a chave pública é apenas resultado de um cálculo, como comprovar que esse cálculo está única e exclusivamente associado a alguém que conheça o valor secreto s que o gerou? Em criptossistemas de chave pública, se não houver uma garantia de legitimidade da chave pública, não haverá segurança alguma.

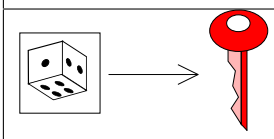
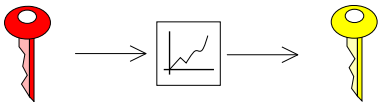
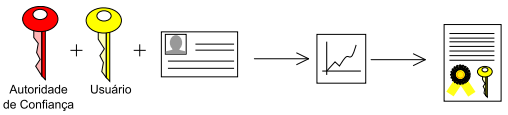
Uma solução para a legitimação de chaves públicas que nos interessa revisar aqui é a baseada em certificados digitais. Para se garantir que (s, P) pertence a um certo usuário com identidade ID, introduzimos uma entidade confiável que chamaremos de Autoridade de Confiança (AC), ou autoridade certificadora. AC tem seu próprio par de chaves (s_{AC}, P_{AC}) e emite certificados digitais que atestam que um certo valor P pertence a uma entidade identificada por ID. Todos que confiarem na autoridade, usarão P certos de que estarão se comunicando com ID.

Um **certificado digital** (no modelo convencional) nada mais é do que um documento que contém, na essência, dados sobre ID, sua chave pública P, dados sobre a autoridade AC, além de uma assinatura digital de AC que assegura a validade dos dados do certificado. Portanto, a assinatura no certificado é uma **garantia** de que a chave P está associada com ID. Podemos interpretar essa assinatura como uma função que tem na entrada a chave pública P, a identidade ID e a chave secreta da autoridade s_{AC}.

Uma **ICP** nada mais é do que um agregado de hardware, software, pessoal especializado e procedimentos, para gerenciar todo o ciclo de vida dos certificados: da sua criação, distribuição até sua renovação ou revogação. Alguns aspectos do funcionamento dessa infraestrutura serão revistos ao longo do texto, conforme se fizer necessário.

Os principais atributos da criptografia de chave pública, no modelo convencional sobre ICP, são sintetizados na Tabela 2.1. Durante a análise dos modelos alternativos, veremos que a variação desses atributos modifica sensivelmente as propriedades do modelo.

Tabela 2.1: *Atributos do modelo convencional de criptografia de chave pública.*

Chave secreta	**Chave pública**	**Garantia**
s	$P = f(s)$	$f(ID, P, s_{AC})$
		
escolhida pelo usuário (ou pela autoridade)	calculada pelo usuário (ou pela autoridade)	calculada pela autoridade, exclusivamente

Para cifrar uma mensagem para o dono de P ou para verificar uma assinatura dele, os usuários usam o valor da chave P, obtido do certificado (ver a Figura 2.2).

Para decifrar uma mensagem destinada ao dono de P ou para que este crie uma assinatura, é necessária a chave secreta s correspondente.

2.1.1 Níveis de Confiança na Autoridade do Sistema

Quando dissemos que a autoridade do sistema deve ser uma entidade de confiança, não entramos no mérito sobre o que significa confiança. No trabalho de Girault (1991), são definidos os três níveis

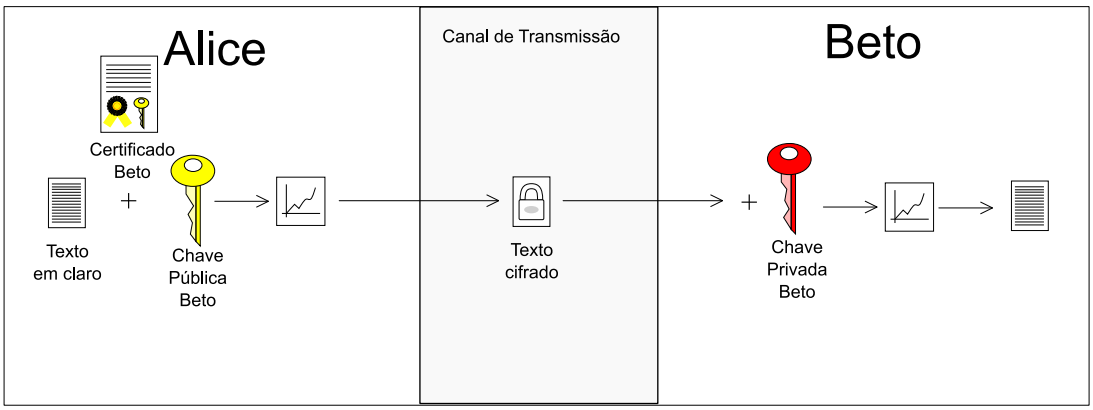


Figura 2.2: *Cifrando no modelo convencional com ICP*

abaixo, que indicam o grau de credibilidade que devemos depositar na autoridade.

Nível 1: a autoridade conhece (ou calcula facilmente) chaves secretas dos usuários; pode personificar qualquer entidade sem ser detectada;

Nível 2: a autoridade desconhece (ou dificilmente calcula) chaves secretas dos usuários; pode personificar qualquer entidade, gerando falsas chaves públicas, sem ser detectada;

Nível 3: a autoridade desconhece (ou dificilmente calcula) chaves secretas dos usuários; pode personificar qualquer entidade, porém é detectada.

No modelo com ICP e certificados de chave pública X.509, a autoridade alcança nível 3, o mais desejável. Os modelos alternativos, muitas vezes caem em níveis mais baixos e, consequentemente, a autoridade tem mais poder e os usuários precisam confiar mais nela.

2.2 Criptografia de Chave Pública Baseada em Identidade

A ideia central da criptografia de chave pública baseada em identidade é muito simples. Se é um problema o fato da chave pública ser um valor numérico sem sentido explícito, por que não calcular a chave secreta a partir da pública, que passa a ser uma cadeia de caracteres com algum significado? Shamir (1984) propôs que a chave pública fosse a própria identidade do usuário, como nome, endereço de email, CPF, número de telefone celular, endereço IP, número serial de dispositivos eletrônicos, etc.

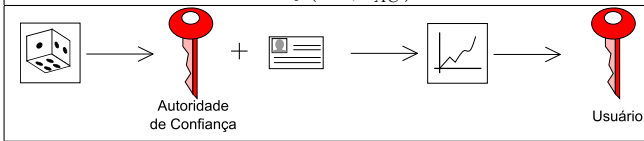
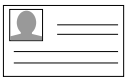
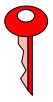
Se a chave pública é predeterminada (igual à identidade), como, então, calcular a chave secreta? A resposta a essa questão vem junto com as primeiras premissas de segurança do modelo: existe uma AC, com as seguintes atribuições principais:

- Criar e manter a guarda segura de uma chave mestra secreta s_{AC};
- Identificar e registrar todos usuários do sistema;
- Calcular as chaves secretas dos usuários;
- Entregar as chaves secretas de forma segura (com sigilo e autenticidade).

Em 1984, Shamir descreveu o modelo e algoritmos para assinatura digital. Foram necessárias quase duas décadas até que fossem descobertos algoritmos de cifragem eficientes e demonstrados seguros, para que o modelo baseado em identidade despertasse um interesse renovado nos pesquisadores e na indústria.

Para fins de comparação, na Tabela 2.2, vê-se que a chave secreta é calculada em função do segredo da autoridade do sistema e da identidade do usuário. Para uma f conveniente, é inviável recuperar a chave mestra a partir dos valores de ID. E somente a autoridade é capaz de gerar as chaves secretas, de modo que a própria chave secreta s é a garantia de que o uso de ID funcionará nas operações criptográficas envolvendo o dono dessa identidade.

Tabela 2.2: *Atributos do modelo de criptografia de chave pública baseada em identidade.*

Chave secreta	**Chave pública**	**Garantia**
$s = f(ID, s_{AC})$	ID	s
		
calculada pela autoridade e compartilhada com o usuário	escolhida pelo usuário ou formatada pela autoridade	

Para cifrar uma mensagem para o dono de ID ou para verificar uma assinatura de ID, os usuários usam a identidade ID mais os parâmetros públicos do sistema, que incluem a **chave pública da autoridade** (ver a Figura 2.3).

Para decifrar uma mensagem destinada a ID ou para que este crie uma assinatura, é necessária a chave secreta de ID.

2.2.1 Vantagens

O modelo baseado em identidade é atraente por apresentar várias vantagens interessantes. A primeira é que a chave pública pode, em grande parte dos casos, ser memorizável por humanos. Muito diferente da chave pública convencional, que costuma ser uma cadeia binária com centenas ou milhares de bits. A identidade pode ser informada pelo próprio usuário aos seus parceiros e não há a obrigatoriedade de manutenção de diretórios de chaves.

Se houver credibilidade na autoridade do sistema e no conteúdo identificador da identidade, a

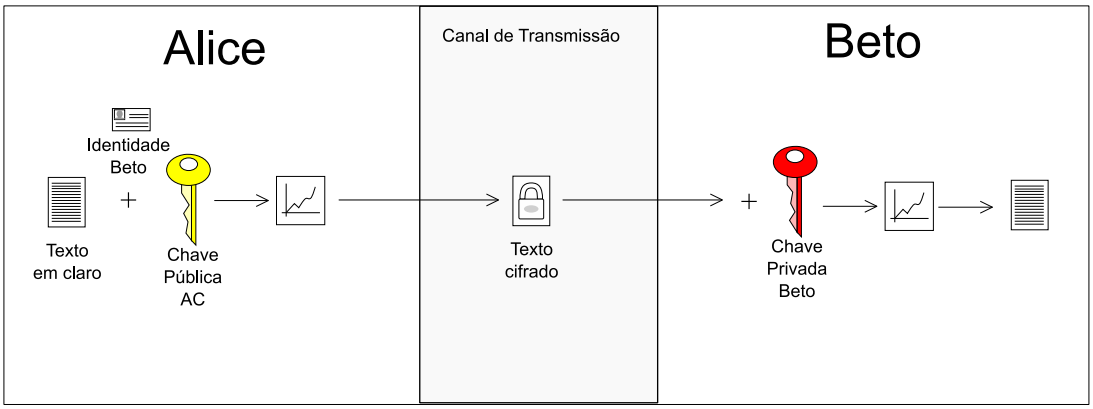


Figura 2.3: *Cifrando no modelo baseado na identidade*

certificação ocorre implicitamente. Tornam-se dispensáveis os certificados digitais e, mais ainda, a infraestrutura que os gerencie. Como consequência, muitos procedimentos existentes sobre uma ICP deixam de existir nos sistemas baseados em identidade.

Para que seja possível visualizar a economia de tempo de processamento, de custos de armazenamento e de transmissões de dados, vamos relembrar, por exemplo, como é, de maneira geral, uma operação de cifragem com ICP. Para Beto cifrar uma mensagem para Alice, antes de tudo, ele deve obter o certificado que fora emitido para a Alice (consultando um diretório público ou a própria Alice). De posse do certificado, Beto precisa verificar o período de validade e a assinatura contida no certificado. A verificação da assinatura é um processo que, às vezes, percorre todo o caminho de certificação das autoridades certificadoras envolvidas na hierarquia, até se chegar à autoridade certificadora raiz. Se até este ponto, nada falhou, Beto pode guardar o certificado da Alice para futuros usos. Entretanto, antes de cada uso, Beto precisa consultar uma autoridade de validação, para verificar se o certificado não foi revogado (quase sempre, um procedimento de consulta a um servidor que esteja on-line). Estando o certificado válido e não revogado, Beto extrai a chave pública da Alice, cifra a mensagem e transmite.

No modelo baseado em identidade, basta que os parâmetros do sistema sejam autênticos (no modelo com ICP também é preciso que os parâmetros do sistema sejam autênticos). Satisfeita essa condição, as operações de Beto para cifrar com base na identidade se resumem a obter o identificador da Alice, cifrar e enviar (considerando-se que revogação de identidade é tratada como explicado adiante).

Uma peculiaridade do modelo baseado em identidade é o fato da chave pública poder ser usada antes do cálculo da chave secreta. Assim, é possível cifrar uma mensagem para quem ainda não se registrou junto à autoridade do sistema nem possui chave secreta de decifração. Por contraste, no modelo com certificados, o usuário deve primeiramente se registrar e obter seu certificado, para então poder receber uma mensagem cifrada sob sua chave pública.

2.2.2 Desvantagens

O preço a pagar por todas essas vantagens é uma série de desvantagens, que, em alguns contextos, podem ser muito críticas.

A primeira desvantagem, que é característica de sistemas baseados em identidade "puros" (isto é, sem modificações que tentam minimizar ou eliminar os efeitos dessa característica) é a **custódia de chaves**. Conforme explicado anteriormente, a autoridade do sistema tem a capacidade de gerar as chaves secretas de todos os usuários sob sua responsabilidade. Isso implica que a autoridade alcança nível 1 de confiança, na definição de Girault (1991). Consequentemente, pode decifrar quaisquer textos cifrados a que tiver acesso (se puder identificar a identidade do destinatário). Também pode assinar em nome de qualquer usuário e não há como garantir irretratabilidade. Portanto, é fundamental que a autoridade do sistema seja confiável o bastante para que ações de bisbilhotagem ou de falsificação como essas sejam controláveis.

A propriedade de custódia de chaves, referenciada por *key escrow* nos textos em inglês, nem sempre é indesejável. Dentro de uma empresa, por exemplo, se todos os documentos e dados sensíveis são cifrados pelo funcionário que o gerou, a diretoria pode ter acesso à decifração em caso de morte ou desligamento do funcionário. Quando houver necessidade de monitoria do conteúdo de emails cifrados, também pode ser justificável a custódia de chaves. No entanto, para a maioria das

aplicações, custódia de chaves é uma desvantagem.

Outro ponto desfavorável ao modelo baseado em identidade é a necessidade de um **canal seguro** para distribuição das chaves secretas. Se a entrega ocorrer num ambiente em rede e remotamente, há que se garantir autenticação mútua e entrega com sigilo. Para podermos comparar, no modelo sobre ICP, a autenticação frequentemente acontece durante o registro do usuário junto à autoridade de registro (muitas vezes presencial), mas não há transmissão nem compartilhamento de segredo.

Do lado do usuário, a guarda da chave secreta deve ser à prova de perdas e roubos, o que sugere o uso de mecanismos adicionais para proteção da chave, como dispositivos de hardware, senha ou desafio/resposta. E sempre que for necessário renovar a chave secreta, é obrigatória a interação do usuário com a autoridade do sistema. Esses aspectos de guarda e renovação parecem ser equivalentemente implementados nos modelos com ICP e baseado em identidade.

O **risco de comprometimento da chave mestra** no sistema baseado em identidade é muito alto, maior que no modelo convencional. Sobre a ICP, o comprometimento da chave secreta da autoridade potencialmente leva à criação de certificados falsos de chaves públicas. Portanto, são afetadas as operações que ocorrerem num momento posterior ao comprometimento da chave mestra. Já no modelo baseado em identidade, quando a chave mestra é roubada ou perdida, operações criptográficas do passado e do futuro estão comprometidas, para todos os usuários sob a autoridade em questão. Qualquer texto cifrado capturado poderá ser decifrado, se o intruso souber as identidades, e independentemente se a cifra foi gerada antes ou depois do vazamento da chave mestra.

Outra preocupação que se deve ter no modelo baseado em identidade é a de lidar com **homônimos**, uma vez que a identidade deve ser única, e a possibilidade de **revogação de identidades**. Caso a chave secreta de um usuário seja comprometida, sua identidade deve ser revogada. Portanto, não é recomendável usar simplesmente o número do CPF ou do telefone celular, por exemplo, como identificador de usuário. Nem sempre é possível cancelá-los. Uma alternativa é concatenar ao identificador principal um período de validade, como em `JoaoSilva-deJan09aDez09`. Caso esse usuário tenha seu segredo comprometido, nova chave secreta poderia ser gerada para uma nova identidade, como `JoaoSilva-deAgo09aDez09`. No entanto, pode não ser trivial garantir que ninguém use a antiga identidade comprometida. Diminuir a granularidade do período, por um lado pode ajudar, por outro, sobrecarregará a autoridade, que terá que renovar as chaves com maior frequência.

2.2.3 Características Adicionais

Como observado por Shamir (1984), o modelo baseado em identidade é ideal para grupos fechados de usuários, como executivos de uma multinacional ou filiais de um banco, uma vez que a sede dessas corporações podem servir como autoridade do sistema, em que todos confiam. Aplicações de pequena escala, em que os custos com implantação e manutenção de uma ICP sejam proibitivos, são candidatas ao uso do modelo baseado em identidade. Quando as desvantagens citadas anteriormente não forem críticas, as características do modelo possibilitam implementações interessantes.

Vale evidenciarmos a flexibilidade de definição da chave pública. A cadeia de caracteres associada com a identidade pode, em princípio, ser qualquer coisa. A concatenação de indicadores de tempo e atributos possibilitam aplicações em serviços com disponibilidade temporal ou sigilo no envio de mensagens com base em papéis.

Alguns exemplos de serviços com disponibilidade temporal: documento confidencial que possa

ser revelado à imprensa ou a um grupo particular, somente a partir de determinada data e hora; lances de um leilão que devem ser mantidos em segredo até o fim das negociações; ou exibição de um filme que deve ser habilitada somente dentro do período de locação contratado.

Vários trabalhos foram desenvolvidos para adaptar o modelo baseado em identidade a hierarquias de autoridades, dentre os quais, o de Gentry e Silverberg (2002) é um dos pioneiros. Uma das vantagens em trabalhar de forma hierárquica é divisão da responsabilidade da autoridade do sistema com autoridades subordinadas. E isso traz consequências importantes. A criação das chaves secretas dos usuários pode ser delegada a níveis inferiores da hierarquia, distribuindo a carga de trabalho da entidade geradora. Além disso, o segredo de cada usuário passa a depender das chaves mestras de mais de uma autoridade e, portanto, diminui o risco do comprometimento de uma chave mestra e há maior controle sobre o uso indevido de uma chave secreta por conta da custódia.

A organização hierárquica de autoridades também viabiliza a cifragem para grupos de usuários. Mas uma generalização desse tema se deu com o trabalho de Abdalla *et al.* (2006), a partir do qual a identidade pode conter caracteres curingas. Num serviço de correio eletrônico cifrado, por exemplo, `*@*.usp.br` atinge todos usuários de todos departamentos da USP; `admin@*.usp.br` diz respeito a todos administradores de sistema dentro daquela universidade.

O modelo baseado em identidade também tem sido alvo de estudos na busca por alternativas ao SSL/TLS, para aplicações na Web, como se pode ver com Crampton *et al.* (2007). Com a eliminação de certificados, simplificam-se o processo de distribuição de chaves públicas e o controle de acesso. De forma semelhante, o modelo tem sido explorado para prover segurança em várias outras áreas de aplicação, como computação em grade e redes de sensores (ver por exemplo Oliveira *et al.* (2011), Lim e Paterson (2005) e Szczechowiak *et al.* (2008)), entre outras aplicações.

2.2.4 Breve Histórico

Quando Shamir descreveu o modelo baseado em identidade, em 1984, apenas assinatura digital ganhou algoritmos concretos. Por muitos anos sem sucesso, pesquisadores trabalharam na busca por algoritmos eficientes e seguros para cifragem e decifração, que compõem o chamado esquema IBE (*Identity Based Encryption*).

A partir de 2001, dois criptossistemas baseados em identidade, mudaram este panorama. Um foi proposto por Cocks (2001), fundamentado em resíduos quadráticos, e outro por Boneh e Franklin (2003), baseado em emparelhamentos bilineares. Este último ganhou notoriedade não apenas pela eficiência, bem como pela formalização completa de IBE e pelas demonstrações de segurança que se tornaram referência para outros protocolos de cifragem sobre emparelhamentos. Vale mencionarmos que o uso de emparelhamentos bilineares em esquemas baseados em identidade havia sido anteriormente estudado por Sakai *et al.* (2000) e que Joux (2000) também já tinha sinalizado que emparelhamentos apresentavam grande atrativo para a criptografia, ao propor o primeiro protocolo de acordo de chaves de três participantes com uma só interação.

Os criptossistemas hierárquicos baseados em identidades surgiram na sequência, conforme detalhados por Gentry e Silverberg (2002) e, posteriormente, por Chatterjee e Sarkar (2007); Yao *et al.* (2004). As implementações com hierarquia de autoridades se justificam não apenas porque reduzem a responsabilidade e a sobrecarga sobre uma autoridade centralizada, mas também porque são uma abordagem para eliminar a característica de custódia de chaves.

Desde então, o modelo recebeu cada vez mais atenção dos pesquisadores, que passaram a acres-

centar melhorias em várias frentes: aumento da velocidade do cálculo de emparelhamento por Fan *et al.* (2008), redução do tamanho da chave com Naccache (2007), e aumento do nível de segurança com demonstrações sem a hipótese de oráculos aleatórios, que é o caso do esquema proposto por Waters (2005). Paralelamente, surgiram inúmeros protocolos e aplicações interessantes; só para citar alguns:

- Assinatura em anel;
- Assinatura curtas;
- Assinatura em grupo;
- Cifrassinatura;
- Acordo de chaves autenticado;
- Acordo de chaves com vários participantes;
- Implementação de disponibilidade condicional.

2.2.5 Avanços importantes

Não é tarefa fácil enumerar todos os avanços relevantes que foram feitos sobre o modelo baseado em identidade, pois muitos trabalhos merecem menção. Entretanto, para finalizarmos esta subseção, pontuamos dois bastante recentes, que nos dão uma noção razoável sobre o atual estágio das pesquisas que tratam dois pontos críticos do modelo baseado em identidade: custódia de chaves e revogação da identidade.

Custódia de chaves é provavelmente uma das mais indesejáveis características no modelo baseado em identidade. Na tentativa de removê-la, Al-Riyami e Paterson (2003) e Gentry (2003) acabaram por criar modelos alternativos, que não são baseados em identidade por possuírem outra chave pública. Entretanto, mantendo-se a identidade como único valor de chave pública, a solução mais comum para eliminar a custódia de chaves gira em torno de uma hierarquia de autoridades e de adaptações sobre os esquemas existentes.

Chow (2009), no entanto, propõe uma nova abordagem para impedir que a autoridade decifre mensagens de seus usuários. Se o texto cifrado embutir uma garantia sobre o anonimato do destinatário, passa a ser proibitivo o custo computacional para que a autoridade tente decifrar as mensagens trafegadas. Chow estendeu o modelo de segurança para IBE de modo a contemplar o anonimato, descreveu um esquema concreto seguro sob esse modelo e propôs uma nova arquitetura para permitir que a emissão da chave secreta seja feita de forma anônima perante a autoridade.

A solução convencional que existe para o problema de revogação de identidade é a concatenação de períodos de validade junto do identificador do usuário, criando identidades como no exemplo `JoaoSilva-Setembro09`. O problema dessa técnica é a alta carga de trabalho sobre a autoridade, na medida em que aumenta a frequência de renovação de chaves. Em termos computacionais, dizemos que a complexidade da renovação é linear no número de usuários e períodos de validade.

No trabalho de Boldyreva *et al.* (2008), a identidade é preservada (sem concatenação) e a complexidade de renovação é logarítmica no número de usuários, o que é muito eficiente quando a quantidade de usuários envolvidos é bastante grande.

2.2.6 Aplicações do Modelo Baseado em Identidade

Appenzeller e Lynn (2002) propuseram um novo protocolo de segurança na camada de rede que permite comunicação autenticada e cifrada entre os nós da rede. Os autores usaram IBE na formulação do protocolo, tornando-o uma **alternativa ao IPSec**. Dentre as vantagens em se usar IBE, podem ser citados o fato de ser desnecessário o processo de *handshaking* inicial e não há troca de certificados para se enviar mensagens cifradas. Neste caso, o remetente simplesmente envia um pacote cifrado com o endereço IP do destinatário.

A importância da **computação em grade** se deve ao aumento expressivo de aplicações que requerem poder computacional e capacidade de armazenamento cada vez maiores. A comunicação segura sobre uma infraestrutura de computação em grade normalmente ocorre sobre uma ICP. Uma forma de aliviar a carga de trabalho, reduzir o tráfego e evitar o armazenamento de certificados digitais é empregar o modelo baseado em identidade, como uma abordagem alternativa em tais ambientes Lim (2006). Nesse trabalho, o autor propõe um protocolo de acordo de chaves com autenticação mútua baseado em identidade e indica como viabilizar serviços de delegação e *single sign-on*.

Redes tolerantes a atrasos e desconexões (DTNs) são redes caracterizadas pela conectividade intermitente e que, por algum motivo, são desconectadas, interrompidas ou apresentam um certo atraso na entrega de pacotes. Podem estar em uma área de grande extensão ou até debaixo da água. Nesses tipos de redes, o modelo baseado na identidade pode contribuir na implantação de confidencialidade, conforme Asokan *et al.* (2007). Segundo os autores, a adoção de IBE em DTNs diminui a carga sobre o servidor e os receptores são menos exigidos com relação à conectividade.

Outra aplicação de interesse em que o modelo baseado em identidade tem participação é a **busca em dados cifrados**. Um caso particular desse é o de busca em dados cifrados com base em palavras-chave (*Public-key Encryption with Keyword Search*, ou PEKS).

Considere sistemas de emails cifrados, em que palavras-chave podem ser pesquisadas por um redirecionador sem que o conteúdo seja revelado. Por exemplo, uma secretária previamente autorizada, pré-organizando emails de seu diretor por palavras-chave como “urgente” ou “projeto-X”. Sem que ela consiga ler o conteúdo dos emails, será capaz de redirecionar as mensagens com tais palavras-chave e priorizá-las. Usuários não autorizados são incapazes de identificar tais palavras-chave dentro dos emails, uma vez que estão cifradas. Naturalmente, essa secretária pode ser substituída por um sistema automatizado de filtragem ou de roteamento, junto do servidor de emails.

Analogamente, *logs* cifrados podem ser pesquisados por um auditor pré-habilitado a localizar ocorrências relacionadas a palavras específicas. No trabalho de Abdalla *et al.* (2008) é apresentado um importante avanço nessa área; uma das contribuições refere-se a uma transformação de um esquema IBE com garantia de anonimato para um esquema seguro de criptografia de chave pública com busca de palavras-chave.

Uma das formas de se implementar autenticação mútua é por meio do uso de **cartões inteligentes**. Scott *et al.* (2006) descrevem a implementação de três emparelhamentos bilineares sobre um *smartcard* de 32 bits. Neste artigo, os autores demonstram que os emparelhamentos podem ser calculados com eficiência equivalente à alcançada pelas primitivas criptográficas clássicas.

2.3 Criptografia de Chave Pública Autocertificada

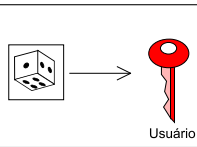
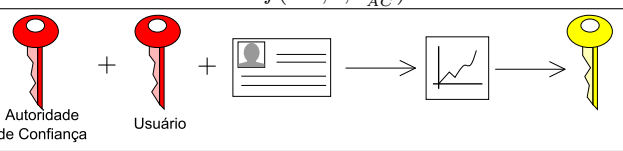
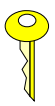
O modelo de criptografia de chave pública autocertificada foi proposto inicialmente por Girault (1991). Aqui, a ideia é que a própria chave pública contenha uma garantia de que seu valor está associado com a identidade. Isso é possível se existir uma autoridade confiável que auxilia na geração da chave pública e se o cálculo dessa chave depender:

- da identidade do usuário;
- do segredo da autoridade;
- e do segredo do usuário.

A criação da chave pública autocertificada sempre ocorre a partir de um protocolo interativo entre o usuário e a autoridade. Analogamente ao modelo convencional de criptografia de chave pública, na proposta de Girault o usuário escolhe sua própria chave secreta e a mantém em sigilo. Durante a interação, tanto o usuário quanto a autoridade precisam comprovar que conhecem um segredo (suas respectivas chaves secretas), sem revelá-lo ao outro. Desse modo, a interação pode embutir um protocolo de identificação, uma assinatura ou uma forma qualquer de conhecimento-zero.

Dependendo da construção do protocolo, a chave pública é calculada pelo usuário ou pela autoridade. Porém, em todos os casos, só o usuário conhece sua chave secreta, o que significa que não há custódia de chaves, um dos pontos fracos do modelo baseado em identidade. Os atributos desse modelo são resumidos na Tabela 2.3.

Tabela 2.3: *Atributos do modelo de criptografia de chave pública autocertificada.*

Chave secreta	**Chave pública**	**Garantia**
s	$P = f(ID, s, s_{AC})$	P
		
escolhida pelo usuário	calculada pela autoridade ou pelo usuário, em um protocolo de prova interativa	

O uso de uma chave pública falsa impede a correta inversão da operação criptográfica. Em outras palavras, se Beto cifrar um texto com uma chave pública falsificada para Alice, ela não será capaz de decifrá-lo. Da mesma forma, se Alice assinar um documento, todos que forem enganados sobre sua verdadeira chave pública não poderão verificar essa assinatura. Isso caracteriza o que chamamos de certificação implícita.

No modelo tradicional com certificados, a validade da chave pública é explicitamente verificada antes de seu uso. Já no modelo autocertificado, isso não ocorre. A legitimidade de uma chave pública autocertificada é confirmada implicitamente quando a inversão da operação criptográfica for bem sucedida. É claro que essa propriedade nem sempre é conveniente. No trabalho de Kim *et al.* (1999), por exemplo, pode ser conferida uma solução em que é possível verificar previamente a chave certificada. Na Figura 2.4, vemos o uso da chave autocertificada num processo de cifragem e decifração.

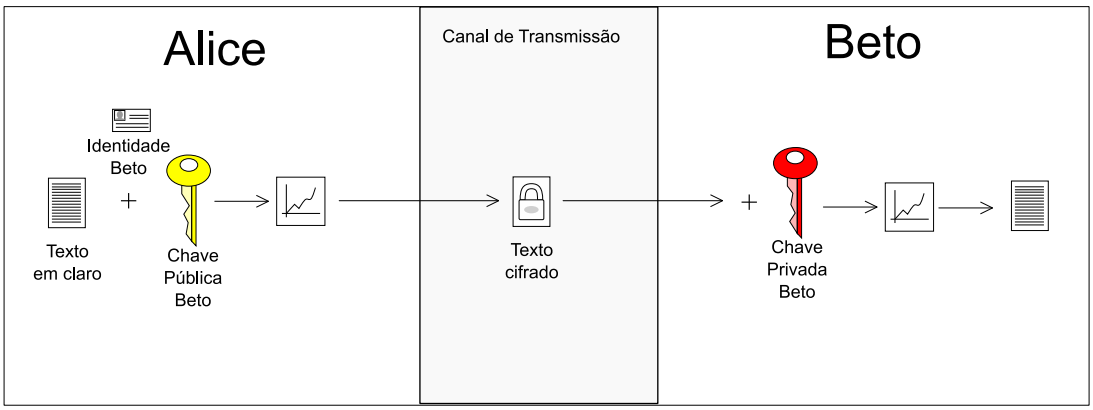


Figura 2.4: *Cifrando no modelo autocertificado*

Em seu trabalho, Girault mostrou duas formas de se gerar chave pública autocertificada e indicou como realizar um acordo de chaves com autenticação mútua. Em outro trabalho independente, Günther (1989), também foi apresentado um protocolo de acordo de chaves com autenticação em que as identidades dos usuários eram usadas nos cálculos da chave negociada. A ideia de explorar a identidade para indiretamente validar a chave pública é retomada em Al-Riyami e Paterson (2003), com outro modelo alternativo, que estudaremos na próxima subseção.

Em síntese, a **vantagem** mais importante da criptografia de chave pública autocertificada é a inexistência de certificados digitais para as chaves públicas (uma vez que a certificação acontece implicitamente) e sem recair em custódia de chaves.

Em contrapartida, surgem **desvantagens**. Relativamente ao modelo baseado em identidade, o de chave pública autocertificada é menos atrativo por requerer um repositório de chaves públicas (ou cada usuário deve informar sua chave pública aos outros).

Entretanto, o ponto mais crítico é a segurança do modelo originalmente definido por Girault (1991). Na época daquela publicação, as formalizações de demonstração de segurança para criptografia de chave pública começavam a ser construídas. Até então, os protocolos não possuíam demonstrações matemáticas de sua segurança; continham simplesmente alguns argumentos intuitivos sobre a dificuldade de sua quebra. Os protocolos de Girault e a maioria dos trabalhos subsequentes não apresentavam modelos formais de segurança e, portanto, não eram demonstrados seguros. Ao contrário, muitos deles foram quebrados em algum momento posterior.

Até mesmo os esquemas de Girault (1991) sofreram abalo. No texto original, o autor alegou ter encontrado uma forma de manter o nível 3 de confiança na autoridade, sem ter que distribuir certificados digitais. Com Saeednia (2003), no entanto, os esquemas de Girault foram rebaixados ao nível 1 (o mesmo do modelo baseado em identidade). Uma correção foi sugerida nesse mesmo artigo, porém o preço para se recuperar a classificação de nível 3 foi um alto custo computacional.

Vários pesquisadores estudaram o problema de construir esquemas baseados na chave pública autocertificada que pudessem ser demonstrados seguros. Os resultados positivos quase sempre surgiram quando foram introduzidas variações sobre a concepção original de Girault.

Essas variações, que ganharam nomes como certificado implícito ou assinatura autocertificada, são mais adequadamente enquadrados dentro de outros modelos. Breves descrições sobre as variantes mais interessantes serão dadas na próxima subseção.

A chave pública autocertificada, como garantia de autenticação do usuário, não evoluiu como modelo independente. Isto é, não se chegou, pelo menos até o momento, a um conjunto de primiti-

vas básicas que garantam confidencialidade, autenticidade e integridade, todas demonstravelmente seguras.

2.3.1 Breve Histórico e Avanços

Com o objetivo de exemplificarmos o conceito de chave pública autocertificada, vamos descrever um dos protocolos de geração de chaves de Girault (1991), um que é baseado na assinatura RSA.

Inicialmente, a autoridade seleciona parâmetros RSA, isto é, escolhe n, produto de dois primos p, q. Calcula seu par de chaves: escolhe e relativamente primo a $(p-1)$ e $(q-1)$, e calcula d como inversa de e no módulo $(p-1)(q-1)$. Escolhe g de ordem maximal no grupo multiplicativo $(\mathbb{Z}/n\mathbb{Z})^*$. Os parâmetros n, e, g são tornados públicos e os demais, mantidos em segredo.

Um usuário com identidade I escolhe sua chave secreta s, calcula $v = g^{-s} \ mod \ n$ e entrega v para a autoridade. Por meio de um protocolo de identificação, o usuário deve provar à autoridade que conhece s, sem revelá-lo.

A autoridade, então, calcula e entrega a chave pública:

$$P = (v - I)^d \ mod \ n$$

Conforme dissemos anteriormente, a chave pública autocertificada é calculada em função da identidade do usuário (I), do segredo da autoridade (d) e do segredo do usuário (s), pois $P = (g^{-s} - I)^d \ mod \ n$.

A autoridade alcançará nível 3 de confiança se demonstrar que gera os parâmetros de forma honesta, conforme detalhado por Saeednia (2003).

Um exemplo de uso da chave autocertificada acima é dado por um protocolo de acordo de chaves com autenticação, proposto por Girault (1991), que, embora seja vulnerável ao ataque do intermediário, ilustra o princípio de funcionamento da chave autocertificada de forma simples. Considere que os atributos de Alice são (I_A, s_A, P_A) e de Beto, (I_B, s_B, P_B). Ambos negociam uma chave secreta calculando:

$$\text{Alice calcula} \quad (P_B^e + I_B)^{s_A}$$

$$\text{Beto calcula} \quad (P_A^e + I_A)^{s_B}$$

Como $d \cdot e \equiv 1 \ \ mod \ (p-1)(q-1)$, pode-se verificar que $P^e + I \equiv g^{-s} \ mod \ n$. Desse modo, os dois cálculos acima são iguais a $g^{s_A s_B} \ mod \ n$. O protocolo é autenticado, pois Alice tem certeza de que conversa com Beto e vice-versa, sem a necessidade de se conferir certificados.

Uma quantidade considerável de publicações na Ásia, durante as duas últimas décadas, tem como tema a chave pública autocertificada. É possível encontrar propostas de cifragem, cifra autenticada, acordo de chaves, assinatura e assinatura em grupo, só para citar algumas. Praticamente todas são desprovidas de demonstrações formais e, portanto, não devemos considerá-las para uso prático.

Talvez boa parte desses trabalhos tenha sido motivada pelo forte potencial de **aplicação** do conceito. Petersen *et al.* (1997) relatam os seguintes usos para chave pública autocertificada, com vantagens sobre o modelo convencional de certificados digitais:

- Delegação do poder de decifrar ou assinar;
- Delegação de direitos;

- Votação eletrônica;
- Dinheiro eletrônico;
- Acordo não-interativo de chaves de sessão, com autenticação.

Na ocasião da publicação de Petersen *et al.* (1997) ainda não eram conhecidas as aplicações de emparelhamentos bilineares, que hoje são base de soluções mais eficientes (e demonstravelmente seguras) para os usos acima. Entretanto, vale citarmos as ideias principais que esses autores relatam para aplicação da chave pública autocertificada, pois essas ideias são de alguma forma retomadas ou reaproveitadas em trabalhos posteriores.

Para os casos de delegação de assinatura ou de decifração, os protocolos de geração de chaves autocertificadas são adaptados para serem executados entre o usuário principal e aquele para quem é delegado algum poder. O "procurador" calcula seu par de chaves por meio do protocolo interativo com o emissor da "procuração". A delegação de direitos ocorre dentro de um contexto de hierarquia de autoridades; cada nó da hierarquia pode ser associado a um privilégio, que é concedido àqueles que forem previamente autorizados, por meio da emissão de chaves autocertificadas.

As aplicações de votação e dinheiro eletrônicos citadas por Petersen *et al.* (1997) se baseiam num protocolo de geração de chave pública autocertificada para um pseudônimo (em vez da identidade). O pseudônimo garante o anonimato e o sigilo do voto ou do uso de um dinheiro emitido, mas, ao mesmo tempo, oferece algum nível de rastreabilidade, por exemplo para assegurar que cada eleitor vote uma única vez, ou para que num caso de extorsão o banco e uma entidade de confiança dentro do sistema possam, em conjunto, rastrear as movimentações fraudulentas.

O acordo de chaves com autenticação de Petersen *et al.* (1997) permite que as chaves públicas autocertificadas deem origem a novos pares de chaves, recalculados pelos próprios usuários sem necessidade de interação com a autoridade. Esses pares de chaves são usados no cálculo de chaves de sessão, em protocolo com autenticação implicitamente verificada. Mais precisamente falando, cada usuário divulga um valor de testemunho (em vez da própria chave pública); a partir de um testemunho, da identidade e dos parâmetros públicos, qualquer usuário pode recalcular a chave pública autocertificada. Quando o valor de testemunho é combinado com um período de tempo ou a um número de sessão, por exemplo, é possível realizar uma negociação não interativa de chaves de sessão.

2.3.2 Variações e Mais Aplicações

Quando tentamos isolar os trabalhos relacionados ao de Girault (1991) que apresentam formalizações conceituais e/ou demonstrações de segurança, encontramos variantes do conceito original que não podem ser enquadrados dentro do modelo de chave pública autocertificada, porém mantêm algum aspecto da autocertificação.

Uma primeira variante é a assinatura autocertificada, que pressupõe a existência de uma ICP convencional. Ou seja, todo usuário tem um par de chaves calculado da forma tradicional e obtém um certificado digital para a chave pública. Esse certificado, entretanto, é distribuído de uma forma diferenciada, para otimizar as operações de verificação de assinatura.

No modelo convencional com ICP, a verificação de uma única assinatura sobre um documento acaba levando a dois procedimentos de verificação: primeiro é necessário verificar a assinatura da

autoridade sobre o certificado; sendo esta válida, é extraída a chave pública do usuário para se conferir a assinatura do documento em questão. O esquema de assinatura proposto em Lee e Kim (2002) evita essa verificação dupla. A partir do valor de assinatura do certificado obtido da autoridade, o usuário calcula um par de chaves para assinatura (pode opcionalmente ser diferente para cada documento assinado). A nova chave pública calculada é autocertificada. A assinatura a ser transmitida consiste dos dados do certificado e de um testemunho, a partir do qual é recalculada a chave pública para verificação da assinatura.

Em Shao (2007), outro esquema de assinatura autocertificada é proposto. O autor estabelece que parte do valor da assinatura do certificado deve ser secreto e obtém uma variante mais próxima do trabalho de Al-Riyami e Paterson (2003), de criptografia de chave pública sem certificado.

Também com o objetivo principal de eliminar a dupla verificação de assinaturas em sistemas baseados na infraestrutura de chaves públicas, existe o que a empresa Certicom chama de certificado implícito. Estruturalmente, em nada difere um certificado implícito de um padrão X.509 para chaves públicas; a infraestrutura necessária para ambos é a mesma. No entanto, o valor de assinatura no certificado é usado para qualquer usuário recalcular a chave pública autocertificada. E essa operação de extração do valor da chave pública tem custo computacional menor que o de uma verificação de assinatura (para validar um certificado).

Um exemplo de geração de certificado implícito é descrito em Brown *et al.* (2002). Todos algoritmos associados a essa tecnologia e que são desenvolvidos pela Certicom estão protegidos por uma série de patentes (a exemplo de *US Patent 20090041238*). A linha de produtos *ZigBee Smart Energy* inclui um dispositivo que realiza as funções de uma autoridade certificadora, emitindo certificados implícitos sobre curvas elípticas ECQV (Qu-Vanstone) para dispositivos com limitação de recursos (de armazenamento, de banda ou computacionais).

Outro uso menos esperado da chave autocertificada também se deu dentro do modelo com certificados. Uma dificuldade existente em sistemas com criptografia de chave pública é a validação da segurança de esquemas demonstravelmente seguros, quando implementados em conjunto com outras operações. Pode acontecer, por exemplo, de um protocolo de assinatura com segurança demonstrável não ter garantia de segurança quando usado dentro de uma ICP. No trabalho de Boldyreva *et al.* (2007), os modelos de segurança para esquemas de cifragem e de assinatura incluem as várias operações que acontecem dentro de uma infraestrutura de chaves públicas. Os autores propuseram dois protocolos demonstrados seguros nesse modelo. Na prática, esses protocolos oferecem maior garantia de segurança em implementações reais, pois o modelo de adversários é mais amplo. E um dos pontos chave desse trabalho é o emprego da certificação implícita.

Na Seção 3.4, descreveremos duas variações sobre o modelo sem certificado que capturam o conceito de chave pública que embute em seu cálculo os valores secretos da autoridade e do usuário. Essas variações, chamadas CL-BSS e CL-LK, possuem especificações formais que são possíveis candidatas para definir modelos de segurança para chave pública autocertificada.

2.4 Criptografia de Chave Pública sem Certificados

O modelo de criptografia de chave pública sem certificados foi apresentado originalmente por Al-Riyami e Paterson (2003). Os autores buscavam uma forma de eliminar a custódia de chaves do modelo baseado em identidade, mais especificamente do protocolo de cifragem de Boneh e Franklin

(2003). Combinando ideias deste último com o modelo de chave pública autocertificada de Girault (1991), chegou-se a um modelo intermediário entre o baseado em identidade e o convencional com certificados.

O modelo resultante deixa de ser baseado em identidade, pois há uma chave pública diferente do identificador do usuário. Porém, por haver uso da identidade, ocorre a certificação implícita, sem que haja necessidade de distribuição e armazenamento de certificados digitais. Por esses motivos, costuma-se dizer que o modelo sem certificados de Al-Riyami e Paterson (2003) reúne o melhor dos dois paradigmas: não há certificados e não há custódia de chaves.

Como nos modelos anteriores, o sem certificados também depende da existência de uma autoridade de confiança AC. O papel dela é, além de identificar e registrar todos os usuários, calcular parte das chaves secretas dos usuários.

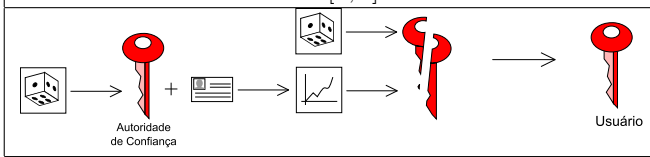
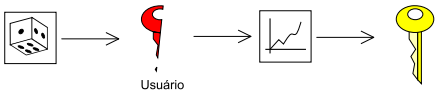
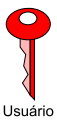
O cálculo das chaves parciais secretas envolve a chave mestra secreta da autoridade, e a identidade do usuário. A entrega dessas chaves parciais deve acontecer de forma segura, com autenticidade e sigilo. Porém, só com o conhecimento da chave secreta parcial, não é possível decifrar nem assinar documentos. Logo não há custódia de chaves.

A chave pública de cada usuário é calculada de modo semelhante ao que ocorre no modelo convencional. Cada usuário escolhe seu segredo e gera a chave pública a partir desse segredo. A chave pública pode ser divulgada pelo próprio usuário ou é colocada em um diretório público.

A chave secreta completa é composta por duas partes: a parcial fornecida pela autoridade e o segredo do usuário. Somente com essas duas partes é possível assinar mensagens ou realizar decifrações. E o usuário é o único que conhece o segredo completo.

Na Tabela 2.4, podemos visualizar os principais parâmetros envolvidos no modelo sem certificados.

Tabela 2.4: *Atributos do modelo de criptografia de chave pública sem certificado.*

Chave secreta	**Chave pública**	**Garantia**
$s = [x, d]$	$[P, ID]$	s
		
x é segredo escolhido pelo usuário $d = f(ID, s_{AC})$ é segredo parcial calculado pela autoridade e compartilhado com o usuário	$P = f(x)$ é calculada pelo usuário	

Para cifrar uma mensagem para A ou para verificar uma assinatura de A, outros usuários precisam da chave pública e da identidade de A. Desse modo, podemos dizer que a identidade é parte da chave pública. Dentre os parâmetros do sistema, a chave pública da autoridade é fundamental nos cálculos e é um dos dados de entrada (ver a Figura 2.5).

Para criar uma assinatura de A é necessária a chave secreta completa de A, mais a identidade de A, a fim de se confirmar a certificação implícita. Para decifrar uma mensagem emitida para A, basta a chave secreta completa de A.

Na **análise de segurança** do modelo sem certificado, os autores modelaram dois tipos de adversários. O primeiro cumpre o papel de um usuário do sistema que tem o poder de substituir a

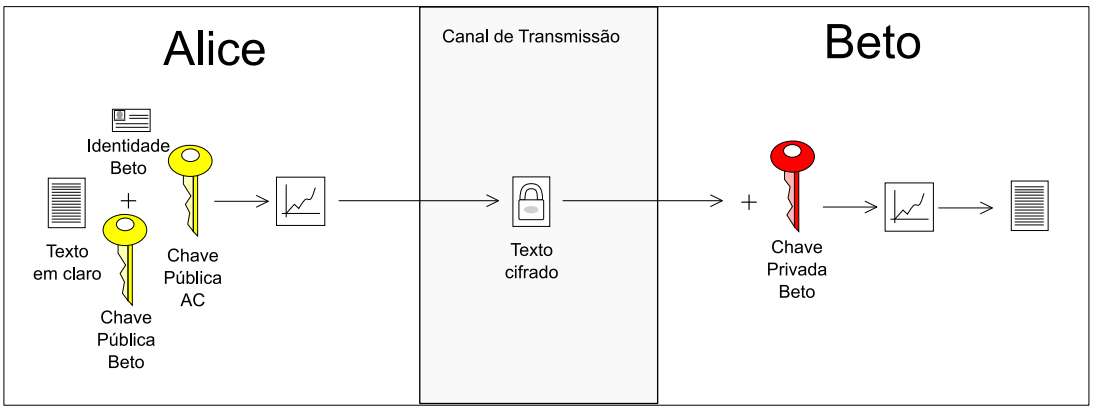


Figura 2.5: *Cifrando no modelo sem certificado*

chave pública de qualquer outro usuário; ele no entanto desconhece a chave mestra da autoridade. Esse tipo de adversário foi definido, pois, como não há certificados para validarem explicitamente as chaves públicas antes de seu uso, não há garantia alguma da veracidade delas.

O segundo tipo de adversário representa a própria autoridade do sistema que, embora honesta, pode tentar bisbilhotar as mensagens cifradas.

Um esquema é considerado seguro no modelo sem certificados quando é acompanhado de uma prova de que adversários, de qualquer dos dois tipos, alcançam probabilidade desprezível de sucesso em diferenciar entre duas cifras de duas mensagens conhecidas.

O esquema de cifragem apresentado em Al-Riyami e Paterson (2003) foi demonstrado seguro sobre essa modelagem de dois adversários. Posteriormente, surgiram propostas de assinatura e outros protocolos variantes, com demonstrações sob os mesmos princípios desse modelo de segurança.

A criptografia de chave pública sem certificados apresenta outras **vantagens**, além da já citada desnecessidade de distribuição de certificados e eliminação da custódia de chaves.

Primeiramente, o risco do comprometimento da chave mestra é menor do que no modelo baseado em identidade. Se houver perda ou violação da chave mestra num sistema sem certificado, serão comprometidas apenas as chaves secretas parciais. Para conseguir decifrar mensagens cifradas anteriormente ao comprometimento da chave mestra, o impostor deverá também obter o segredo do usuário. E para personificar usuários ou decifrar novas mensagens, antes terá que substituir chaves públicas por valores cujo segredo associado seja escolhido pelo adversário.

Outra propriedade, que é exclusiva do modelo sem certificado, é que o processo de renovação da chave pública pode ser totalmente controlado pelo usuário. Isto é, o usuário pode atualizar sua chave pública e até mesmo possuir várias delas, sem ter que se comunicar com a autoridade do sistema. A chave secreta parcial será única e gerada uma só vez para cada identificador.

E, assim como no modelo baseado em identidade, é possível criar a chave pública e usá-la (para cifrar) antes mesmo que o usuário tenha entrado em contato com a autoridade para seu registro e obtenção da chave parcial (que habilitará a decifração).

Uma das **desvantagens** do modelo que ainda não tem uma solução simples é o que foi chamado de ataque *Denial of Decryption* (DoD), em referência aos ataques de negação de serviço (Liu *et al.*, 2007). O DoD se dá quando alguém divulga falsamente uma chave pública ou a substitui em um repositório público, com o objetivo de prejudicar um usuário legítimo do sistema. Este, por sua vez, será incapaz de decifrar mensagens que lhe tenham sido enviadas, se tiverem sido cifradas com a chave falsa. Analogamente, assinaturas válidas deixam de ser verificáveis se for divulgada uma

chave pública ilegítima. E, num primeiro instante, não é possível detectar se o que está errado é a chave ou a assinatura.

Outro preço a se pagar pela vantagem de não haver certificados é uma consequência da autenticação implícita. Quem envia uma mensagem cifrada só terá condições de verificar a autenticidade da chave pública depois que o destinatário conseguir decifrar. Um remetente terá ciência de um eventual ataque DoD somente depois de ter consumido considerável tempo de processamento e de comunicação.

Embora não seja necessário armazenar nem distribuir certificados, o modelo ainda requer um repositório de chaves públicas (ou uma forma de distribuição dessas chaves). Porém, mais crítico do que manter esse repositório, é gerenciar um **canal autêntico e confidencial** para entrega da parcial secreta; no modelo convencional com ICP, o canal deve ser autêntico, mas o certificado é público.

Comparando com o modelo baseado em identidade, aqui esse canal seguro é usado com frequência um pouco menor: basta uma troca segura para cada identificador de usuário. No modelo baseado em identidade, se forem usados períodos de validade no identificador, o canal seguro terá que ser estabelecido a cada renovação.

Da forma como foi proposto originalmente, o modelo sem certificado situa-se no **nível 2 de confiança** na autoridade, pois uma autoridade desonesta pode divulgar chaves públicas falsas. E não é possível comprovar se quem gerou a chave falsa foi a autoridade ou um usuário qualquer.

Uma autoridade desonesta pode proceder da seguinte forma: ela escolhe um segredo, calcula a chave pública correspondente e a divulga como se fosse de um usuário A; se a autoridade capturar textos cifrados para A sob essa falsa chave pública, terá condições de decifrá-los, pois conhece o segredo e a chave parcial secreta. Em seguida, para tentar ocultar a fraude, cifra novamente com a chave pública verdadeira e repassa para o destinatário, fingindo ser o remetente original. Raciocínio semelhante vale para falsificação de assinatura por parte da autoridade.

Portanto, os usuários de um sistema no modelo sem certificados precisam confiar que a autoridade do sistema não propaga ativamente chaves públicas falsas.

Conforme discutido por Al-Riyami (2005), é possível elevar o nível de confiança dentro do modelo sem certificado, se o valor da chave pública for incluído no cálculo da chave parcial secreta. Desse modo, é possível detectar uma fraude da autoridade, sob algumas condições, explicadas a seguir.

Vamos primeiro considerar o caso de um esquema de assinatura, projetado para garantir irretratabilidade. No modelo sem certificados, para assegurar que uma assinatura não tenha posteriormente a autoria negada, o identificador de cada usuário deve incluir o valor da chave pública, por exemplo, concatenando ID com P ($ID||P$). E no cálculo da chave parcial secreta, o usuário precisa provar o conhecimento do valor secreto correspondente; ao mesmo tempo, a autoridade também deve demonstrar que conhece a chave mestra. Em outras palavras, a geração da chave parcial deve consistir de um protocolo interativo com provas de posse de segredo. Nessas condições, o esquema de assinatura tem a propriedade de **irretratabilidade** e a autoridade alcança **nível 3 de confiança**.

Para esquemas de cifragem, a análise é um pouco mais trabalhosa. Quando o identificador do usuário embute o valor da chave pública (como em $ID||P$), não há obrigatoriedade de que a chave parcial seja mantida em segredo. Isso é verdade, pois, a parcial sozinha não é a chave secreta completa e, para que a autoridade consiga bisbilhotar mensagens de algum usuário, antes terá que substituir falsamente a chave pública e o identificador, além de gerar nova parcial.

Considere, então, a seguinte fraude: a autoridade substitui a identidade original da Alice $ID||P$ por $ID||P'$; Beto, ao tentar enviar uma mensagem com sigilo para Alice, é enganado sobre os dados dela e cifra usando $ID||P'$; a autoridade captura tal mensagem e a decifra, descobrindo o conteúdo; a autoridade cifra novamente sob o identificador real e submete para Alice como se nada anormal tivesse ocorrido.

Nem sempre essa fraude será detectável. Primeiro, Beto e Alice terão que desconfiar de tal possibilidade e perceberem que a Alice possui dois identificadores distintos: um foi usado por Beto para cifrar, e outro pela Alice para decifrar.

Se a chave parcial for considerada pública, a evidência dessa fraude será a existência das duas chaves parciais, que só a autoridade consegue emitir. No entanto, a autoridade tentará ocultar a chave parcial derivada da chave pública falsa, para não se autoincriminar.

Por outro lado, se a chave parcial for considerada secreta, compartilhada apenas entre a autoridade e o usuário em questão, a evidência criptográfica dessa fraude será a existência de duas cifras para uma mesma mensagem, sob duas chaves públicas distintas de Alice. Entretanto essa evidência pode ser questionada pela autoridade: quem garante que Beto não inventou a chave pública falsa, criou as duas cifras e está tentando levantar suspeitas sobre a autoridade?

Desse modo, nem sempre a fraude será detectável para o caso de cifragem. E o nível de confiança não chega a 3. Por esse raciocínio, consideramos que o trabalho de Yang e Tan (2011c) está inconsistente. Os autores propõem uma formalização para esquemas de cifragem sem certificado e afirmam que nessa nova formalização a autoridade alcança nível de confiança 3. Yang e Tan supõem que a simples existência de duas chaves públicas para uma mesma identidade implica a existência de duas chaves parciais secretas para essas chaves (que só poderiam ter sido geradas pela autoridade). No entanto, se a autoridade de fato está tentando fraudar o sistema e se conseguir convencer Beto a usar uma chave pública falsa de Alice, poderá negar que gerou a chave parcial secreta para essa falsa chave pública, uma vez que não estará disponível a mais ninguém além da suposta geradora e fraudadora. E no caso de Beto e Alice conseguirem comprovar a existência de duas cifras diferentes para uma mesma mensagem e mesmo destinatário (o que seria uma possível evidência de fraude), não há como afirmar que a autoridade foi a responsável, pois um usuário desonesto também poderia tê-lo feito, uma vez que bastam dados públicos para se gerar uma cifra. Ou seja, não há evidência que prove que a autoridade obteve sucesso em decifrar uma mensagem cifrada sob uma falsa chave e, consequentemente, o nível de confiança que se pode depositar na autoridade é menor que 3.

Usos e aplicações do modelo sem certificado são em grande parte relacionados às aplicações do modelo baseado em identidade. Quando uma aplicação deste último tiver como requisito a eliminação da custódia de chaves, o modelo sem certificado pode ser empregado. Isso pode ser afirmado, pois protocolos com base na identidade quase sempre podem ser modificados para o modelo de Al-Riyami e Paterson.

O modelo sem certificado também é indicado para uso em grupos fechados ou parceiros de negócios, em contextos em que a autoridade de confiança pode ser representada por alguém do próprio grupo.

2.4.1 Evolução do Modelo

Vamos pontuar os trabalhos mais significativos que marcaram a evolução do modelo de criptografia de chave pública sem certificados. Citaremos primeiramente os esquemas que contemplam o

sigilo; os de assinatura são descritos posteriormente.

Dentre os protocolos de maior eficiência computacional para cifragem, podemos citar os de Libert e Quisquater (2006) e de Cheng *et al.* (2007). Ambos usam o modelo de oráculos aleatórios para demonstrar a segurança de construções genéricas de CLE e dos esquemas concretos propostos. Embora o primeiro protocolo seja mostrado seguro em um modelo de adversários mais forte, sua prova envolve a hipótese de dificuldade de um problema computacional que não é mais difícil que o problema usado na prova de segurança do segundo.

No contraponto da maior eficiência computacional, encontram-se os esquemas de maior segurança, demonstrados no modelo padrão (sem oráculos aleatórios), e sob o modelo mais forte de adversários. Dentro desse nicho, o trabalho Dent *et al.* (2008) é a melhor referência que se tem até o momento.

Embora a esmagadora maioria das propostas envolva emparelhamentos bilineares, não é obrigatória tal técnica para que seja viável o modelo sem certificados. Sun *et al.* (2007) mostram que há um exemplo de esquema de cifragem sem emparelhamentos. Os autores melhoram uma versão anterior, Baek *et al.* (2005), entretanto preservam uma grande quantidade de exponenciações. Em termos de eficiência, esse esquema é superado por muitos outros que são baseados em emparelhamentos. Só para se ter uma ideia, a versão mais antiga é comparada com esquemas de cifrassinatura em Barreto *et al.* (2008) e perde em várias ocasiões (isto é, mesmo apenas cifrando, o esquema sem emparelhamento é mais lento que alguns esquemas que cifram e assinam simultaneamente).

Ataque de Negação de Decifração

Talvez não seja exagero dizer que o calcanhar de Aquiles do modelo sem certificados é o que se convencionou chamar de ***Denial of Decryption*** (DoD), ou ataque de negação de decifração. É razoável pensar em uma tal forma de ataque, pois as chaves públicas são distribuídas e não certificadas explicitamente. Se o destinatário (dono da verdadeira chave pública) não conseguir decifrar ou obter uma mensagem diferente da original, o ataque é bem sucedido. No contexto de assinatura, usuários não conseguem validar assinaturas legítimas e sequer podem identificar que o erro está na chave pública e não na assinatura. No caso de implementações que envolvam um repositório centralizado para armazenar as chaves públicas, o ataque DoD terá este nome mais que justificado, se o impostor substituir várias (ou todas) chaves.

A solução delineada por Liu *et al.* (2007) para se evitar o DoD envolve o cálculo da chave pública dependentemente da chave parcial secreta. Isso, entretanto, elimina uma das características peculiares do modelo sem certificado, que é a possibilidade de se gerar chaves públicas antes da interação com a autoridade do sistema. A proposta combina CLE e CLS (respectivamente cifragem e assinatura no modelo sem certificado), criando **duas** chaves parciais secretas, uma para CLE e outra para CLS. Portanto, há também duas chaves secretas completas, uma para decifrar e outra para assinar. A chave pública passa a ser um par $< pk, \sigma >$ onde σ é a assinatura do usuário sobre sua própria chave pública pk. Um remetente deve verificar σ antes de cifrar com pk e, assim, DoD é evitado.

Os autores chamaram essa solução de *self-generated certificate PKC*; ela se assemelha a certificados autoassinados em ICPs. Um questionamento que ainda está em aberto é se não é possível outro tipo de solução, pois, na verdade, os autores inseriram uma espécie de certificado (a assinatura, que deve ser verificada e que só pode existir depois de uma interação com a autoridade) em um modelo

que, em princípio, é sem certificados.

Um outro aspecto problemático no modelo sem certificados, ainda que em menor escala que o DoD, é o ataque da **autoridade mal intencionada**. O modelo de segurança definido originalmente por Al-Riyami e Paterson (2003), presume que a autoridade é honesta e segue os protocolos conforme especificados. No trabalho de Au *et al.* (2007b), no entanto, é apresentada a possibilidade de que a autoridade gere parâmetros desonestamente, de modo a conseguir um atalho para decifrar textos, sem o conhecimento dos usuários.

À exceção de Hu *et al.* (2006); Libert e Quisquater (2006), todos os esquemas propostos até o trabalho de Au *et al.* (2007b), são vulneráveis a essa autoridade mal intencionada e mesmo os trabalhos mais recentes também o são em grande parte. O esquema de Dent *et al.* (2008), por exemplo, falha nesse quesito. Dent (2008) provou que um esquema CLE construído sob a técnica de Dodis e Katz (2005) evita esse tipo de ação; uma implementação concreta de esquema de cifragem sob essa técnica pode ser vista no trabalho de Chow *et al.* (2006), cuja proposta tem por objetivo principal o de resolver o problema de **revogação de chaves públicas** no modelo sem certificados. Outro trabalho que trata o problema da autoridade mal intencionada é o de Hwang *et al.* (2008), que apresenta um CLE demonstrado seguro sem oráculos aleatórios.

A propósito do problema de revogação, a solução de Chow *et al.* (2006) requer um mediador confiável, a quem a autoridade entrega as chaves parciais secretas, e que se torna capaz de iniciar o processo de decifração. A cifra parcialmente decifrada é submetida ao destinatário final, que completa a operação com o secreto aleatório. Com esta solução, o mediador deve estar online e disponível sempre que alguém precisar cifrar ou decifrar.

Com relação a **assinaturas** no modelo sem certificados, existem inúmeros trabalhos publicados, muitos porém demonstrados posteriormente inseguros. Um esquema que ainda se mantém seguro é apresentado por Zhang *et al.* (2006), que é melhorado por Hu *et al.* (2007). Este último aperfeiçoa alguns aspectos do modelo de segurança para assinatura, mas regride em outro (o oráculo de assinatura sempre responde corretamente, mesmo que o adversário forneça valor inválido de chave secreta). Todas essas propostas se baseiam no modelo do oráculo aleatório.

Liu *et al.* (2007) citam um esquema de assinatura demonstrado seguro no modelo padrão, no entanto requer parâmetros bastante longos.

Vale citar o trabalho de Zhang e Wang (2008) que, embora tenha usado um modelo mais fraco de adversários, apresentou CLS seguro contra autoridade mal intencionada, não vulnerável ao ataque DoD, que alcança nível 3 de Girault e todas as demonstrações são feitas sob o modelo padrão, sem oráculos aleatórios. Os autores se valem de uma técnica menos usual em CLS: é adotado um protocolo de conhecimento-zero na geração da chave parcial secreta.

2.4.2 Aplicações do Modelo sem Certificado

Um **fluxo criptográfico**, na denominação de Al-Riyami (2005), é uma sequência de operações criptográficas, como cifrar, autenticar e decifrar, que precisa ser executada numa determinada ordem. No modelo baseado em identidade e no sem certificado, é possível emitir e usar a chave pública antes que a chave secreta completa esteja disponível. O exemplo abaixo ilustra uma aplicação que faz uso dessa característica.

Na maneira tradicional, quando a emissão de um documento depende da aprovação de muitos órgãos, o usuário solicita a aprovação de cada órgão e, com essas autorizações em mãos, é feita uma

solicitação ao órgão emissor do documento. Este último verificará a validade de cada autorização para então emitir o documento solicitado.

Com o uso de sistemas sem certificado, é possível ao órgão emissor entregar o documento eletrônico cifrado com a chave pública e identidade do usuário. Cada órgão que deve aprovar o documento entrega ao usuário uma chave parcial privada. Após recolher todas as chaves parciais que compõe a chave secreta completa, o usuário passa a ter acesso ao documento, sem a necessidade de retornar no órgão emissor. Como não há custódia de chaves, o conluio de um ou mais órgãos não permitirá acesso ao documento.

Esse mesmo exemplo poderia ser adotado com software em ambientes sigilosos, onde o acesso ao software só é liberado após conclusão de etapas de autenticação.

As construções genéricas de esquemas sem certificado se valem de uma composição de esquemas seguros de criptografia de chave pública convencional e baseada em identidade. Isso indica que as aplicações baseadas em identidades podem ser convertidas para o modelo sem certificado, ao custo da necessidade de divulgação das chaves públicas (ou da manutenção de um diretório de chaves) e de um canal seguro para distribuição dos segredos parciais. O modelo sem certificado pode vir a ser uma **ponte** entre sistemas baseados em identidade com os que requerem ICP, em particular, a **ICP-Brasil**.

A possibilidade de renovação da chave pública pelo usuário, sem interação com a autoridade, é uma característica exclusiva do modelo de Al-Riyami e Paterson. Isso pode vir a ser vantagem e ser explorado em alguma aplicação específica. Alguns protocolos com base na chave pública autocertificada apresentam tal propriedade, porém sem garantia de segurança.

2.5 Criptografia de Chave Pública Baseada em Certificado

O modelo de criptografia de chave pública baseada em certificado, proposto por Gentry (2003) é uma variação do modelo convencional que mantém a infraestrutura de chaves públicas.

O objetivo inicial do autor era resolver o problema de revogação de chaves públicas e eliminar o tráfego de validação de certificados. Na solução proposta, os certificados são usados apenas pelo próprio usuário dono (e não por todos que precisem se comunicar com segurança com o proprietário do certificado, como é no modelo tradicional).

No modelo de Gentry, a hierarquia de autoridades certificadoras existe da mesma forma que nas ICPs sob padrão X.509. Cada usuário cria seu par de chaves, ou, alternativamente, a autoridade cria e entrega para o usuário, dependendo dos requisitos da aplicação.

O que muda para a autoridade certificadora é que, para cada período i predeterminado, novo certificado deve ser gerado para todos os usuários (e esse período é menor que na ICP comum). Esse certificado funciona como se fosse uma parte adicional à chave secreta do usuário, pois sem certificado válido para um período, não é possível assinar nem decifrar. Portanto, do ponto de vista do usuário, o que muda em relação ao modelo convencional é a forma como é usado o certificado.

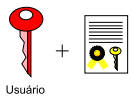
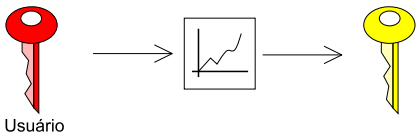
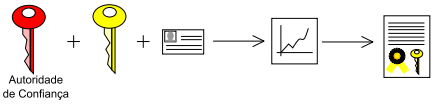
Ao proprietário do certificado, cabe a responsabilidade de obter e armazenar localmente o certificado válido para o período. Ele será usado como se fosse um componente da chave secreta, embora possa ser distribuído em canal público.

Aos usuários terceiros, o certificado não tem serventia alguma, pois para cifrar para A ou para verificar uma assinatura de A, bastam a chave pública e a identidade de A, mais o período i. A

certificação da chave pública ocorre implicitamente.

Na Tabela 2.5, podemos visualizar os principais parâmetros envolvidos no modelo baseado em certificado. E, na Figura 2.6, é ilustrado o processo de cifragem e decifração.

Tabela 2.5: *Atributos do modelo de criptografia de chave pública baseado em certificado.*

Chave secreta	**Chave pública**	**Garantia**
$[s, c]$	$[P = f(s), ID]$	$c = f(ID, P, s_{AC}, i)$
		
s é segredo escolhido pelo usuário (ou pela autoridade)	calculada pelo usuário (ou pela autoridade)	c é o certificado calculado pela autoridade (exclusivamente), para período i

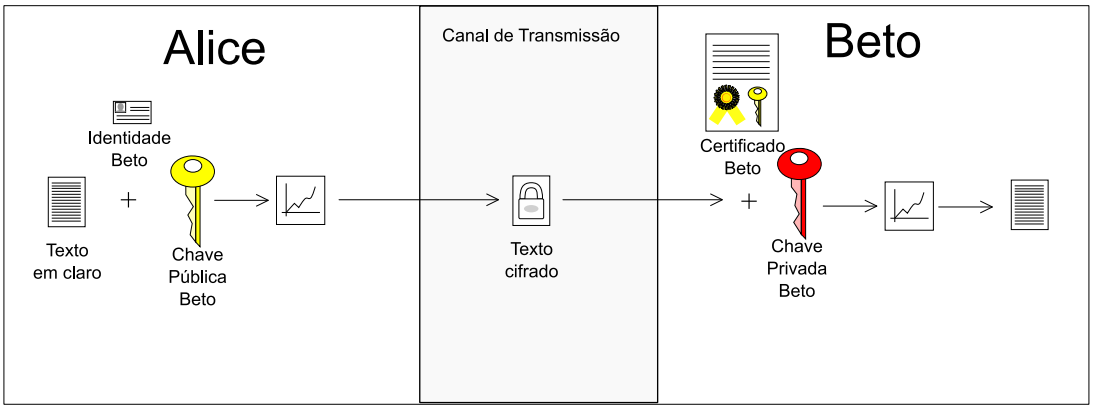


Figura 2.6: *Cifrando no modelo baseado em certificado*

Se compararmos com o modelo convencional de ICP, o modelo de Gentry tem como principal **vantagem** a eliminação do tráfego de validação dos certificados, ou seja, não há consultas para verificação do estado do certificado. O autor estabelece que o certificado emitido para o período i é automaticamente invalidado quando expira o prazo para o qual foi emitido. Uma premissa fundamental é que a chave pública sempre é considerada válida por quem a usa. Portanto, o período deve ser curto o bastante para que o comprometimento de uma chave secreta cause o menor dano possível até o fim daquele período.

Se uma chave secreta é comprometida, o usuário notifica a autoridade e gera novo par de chaves. A autoridade, por sua vez, passa a emitir certificados somente para a nova chave.

O trabalho que um usuário tem antes de cifrar para alguém (ou verificar uma assinatura) é, portanto, o de obter a identidade e a chave pública e de usá-las diretamente. Numa ICP convencional, como já dissemos anteriormente, primeiro é necessário obter o certificado, verificá-lo (prazo de validade e assinatura), validá-lo (consultar uma lista de certificados revogados, por exemplo), para então extrair e usar a chave pública.

Comparando com o modelo baseado em identidade, o baseado em certificado tem a vantagem de eliminar a custódia de chaves, pois a criação do par de chaves acontece exatamente como no sistema tradicional, em que a chave secreta pode ser de conhecimento exclusivo do usuário.

Em relação ao modelo sem certificado, o de Gentry apresenta como vantagens o **nível 3 de con-**

fiança na autoridade e a não necessidade de canal sigiloso para troca de segredos. Em contrapartida, o modelo baseado em certificado ainda requer uma infraestrutura para emissão e distribuição de certificados e pode ser considerado simplesmente uma variação do modelo convencional de criptografia de chave pública sobre ICP.

A principal **desvantagem** do modelo de Gentry é a potencial sobrecarga sobre a autoridade, devido à reemissão de certificados. Isso vai depender de como for definido, no sistema, a frequência com que os certificados devem ser atualizados, combinado com o número de usuários. Para quando for grande o número de usuários, o autor descreve uma técnica auxiliar, que foi denominada *subset covers*, em que os certificados têm sobrevida maior e são "reconfirmados" a cada período de tempo; como consequência, a carga de emissões diminui.

Um ponto bastante crítico associado ao modelo baseado em certificado situa-se nos detalhes de projeto e implementação, pois deles dependem o quão bem será resolvido o problema de revogação de certificados.

Dentro do modelo baseado em certificado, revogação significa: a autoridade para de emitir certificados para a chave pública cuja chave secreta foi comprometida. Na realidade, nem chave pública e nem certificado são revogados.

Suponha que Alice teve seu segredo violado e imediatamente ela entrou em contato com a autoridade para gerar novo par de chaves. Se os demais usuários não forem informados de que a chave pública foi alterada e a continuarem usando, aquele que violou a antiga chave da Alice potencialmente poderá usar o certificado "revogado" até o fim do período de validade dele, seja na tentativa de falsificar assinaturas ou para ler documentos cifrados, capturados de alguma forma.

Desse modo, o ideal é que, antes de usar uma chave pública, todo usuário a obtenha de algum repositório atualizado, ou a pegue diretamente do usuário que a gerou. E ambas possibilidades requerem sistemas online, aproximando-se mais do que costuma ser implementado tradicionalmente, com listas de certificados revogados (CLR) ou verificação online do estado do certificado (OCSP). Outra alternativa é reduzir o período i de emissão de certificados, o que sobrecarrega a autoridade.

Se a decisão de projeto for para que seja mantido um repositório de chaves públicas válidas, consultado por todos usuários antes de uma operação criptográfica, surge nova preocupação no gerenciamento de chaves: a segurança desse repositório deve ser adequada o bastante para que ele nunca seja atualizado falsamente. E esse repositório de chaves públicas substitui o que seria um repositório de certificados digitais, na implementação convencional.

Por outro lado, cada usuário talvez tenha que ter um repositório particular de certificados emitidos para vários períodos. Isso pode ocorrer se i for relativamente pequeno e existirem, por exemplo, certificados diferentes para cada hora ou menos. Suponha um sistema de email baseado em certificado e o seguinte cenário. Alice fica fora da empresa, em visita a algum cliente, passa um período sem consultar o sistema, e ao mesmo tempo recebe grande quantidade de emails cifrados. Ao retornar, Alice terá que obter todos os certificados necessários para ler as mensagens, alguns referentes a um mesmo período, outros não.

Cabe um comentário sobre o tamanho do certificado de Gentry. A rigor, o autor separa as informações do certificado em duas partes. Os dados do usuário, que são essencialmente textuais, fazem parte de seu ID; os usuários podem manter uma cópia local ou, por praticidade, guardar apenas um hash desses dados, pois eles serão usados nos protocolos. O equivalente à assinatura da autoridade sobre os dados do certificado, fica fisicamente separado dos dados. O que o autor chama

de certificado, e que é usado como parte da chave secreta, é apenas esse valor de assinatura. E somente esse valor é atualizado e transmitido constantemente.

Portanto, embora o tráfego para distribuição de certificados possa ser mais contínuo, em cada conexão, o tamanho dos dados é menor. Ainda assim, não é trivial comparar o tráfego neste modelo com o padrão X.509.

O tráfego para distribuição de certificados pode ser maior ou menor no modelo baseado em certificado. Isso depende das características do sistema que o implementar. Digamos que Alice se comunica com sigilo com uma quantidade n de outros usuários do sistema. Nenhum desses n usuários terá que obter certificado da Alice (como ocorre no modelo da ICP tradicional), mas Alice tem que obter seu próprio certificado, um número x de vezes. Durante a fase de projeto de um sistema baseado em certificado, é prudente avaliar como x se relaciona com n e averiguar se o valor de x pode ser um problema ou não.

Gentry (2003) afirma que o modelo baseado em certificado começa a se tornar ineficiente atualizando, a cada hora, mais de 225 milhões de certificados, aproximadamente. Esses dados foram obtidos pelo autor, analisando a capacidade de autoridades certificadoras em sistemas na época.

Para a **análise de segurança** do modelo, similarmente ao modelo sem certificados, são definidos dois tipos de adversários: um representa o usuário comum, porém não certificado; outro adversário representa a própria autoridade, que não conhece o segredo dos usuários e é considerada honesta em não divulgar falsas chaves públicas, mas tenta bisbilhotar mensagens cifradas.

Os esquemas demonstrados seguros no modelo baseado em certificado preveem os dois tipos de usuário. Isto é, um esquema de cifragem é demonstrado seguro se, sob a condição de que o adversário, seja de qual tipo, não conhece o segredo associado a uma chave pública, alvo de um ataque, não tem condições de diferenciar entre dois textos cifrados de duas mensagens distintas. E isso acontece mesmo que o adversário tiver a capacidade de decifrar mensagens para outras chaves públicas. E num esquema de assinatura, apenas com o conhecimento do certificado e da chave pública, os adversários de ambos os tipos não são capazes de gerar uma assinatura válida.

Os esquemas de assinatura mais recentes são demonstrados seguros mesmo que o adversário tenha a capacidade de substituir chaves públicas por valores à sua escolha, porém sob a condição de que não há emissão de certificado válido para a falsa chave.

Para se compreender o **nível de confiança** que a autoridade alcança sob o modelo baseado em certificado, é necessária uma análise semelhante à que foi apresentada no estudo do modelo sem certificado. Aqui, o certificado é uma assinatura da autoridade sobre as informações de identificação do usuário, incluindo a chave pública e um período.

Para que num esquema de assinatura haja garantia de irretratabilidade, o registro da chave pública deve ocorrer sob canal autêntico e com prova de posse de segredo. Se a autoridade divulgar uma chave pública ilegítima, ela terá condições de criar um certificado para essa chave e assinar falsamente. A fraude será detectada se toda chave pública (usada na verificação de assinaturas) for divulgada obrigatoriamente junto do certificado. Dois certificados e duas chaves comprometem a idoneidade da autoridade. Nessas condições, o nível de confiança é 3 e há irretratabilidade de assinatura.

Nos esquemas de cifragem, também será necessária a publicação do certificado junto de cada chave pública. Isso se deve ao fato da autoridade pretender decifrar mensagens, cifradas sob falsas chaves públicas, e reenviar ao usuário final recifradas com a verdadeira chave. A evidência da fraude

se dará com a existência de um certificado emitido para a chave falsa. Como somente a autoridade tem acesso à chave mestra, só ela emite certificados. Se a substituição de chave pública ocorrer em canal autêntico, a autoridade é responsabilizada pela fraude.

Entretanto, gerenciar o armazenamento de todos os certificados, para cada chave pública, passa a ser uma preocupação adicional. E é maior que na ICP tradicional, pois a tendência é que o volume de certificados seja maior.

2.5.1 Breve Histórico

Na ocasião da publicação do trabalho de Gentry (2003), o autor havia definido apenas a cifragem sob o modelo e estudou como seria a construção num ambiente com várias autoridades em hierarquia. Desde então, surgiram novas propostas de esquemas para cifragem e também apareceram modelos para assinatura no paradigma de Gentry.

Na medida em que novos trabalhos foram apresentados, as formalizações do modelo evoluíram e os algoritmos foram aprimorados, seja com relação ao desempenho computacional, seja na melhoria de algum aspecto de segurança. Vamos citar alguns trabalhos de maior interesse.

2.5.2 Cifragem no Modelo Baseado em Certificado

Vamos revisar o esquema cifragem baseado em certificado (CBE) de Gentry (2003), que possui demonstração de segurança contra ataques de texto cifrado escolhido. O esquema é composto de cinco algoritmos, fundamentados em emparelhamento bilinear:

inicializa Dado um parâmetro de segurança k, gera a chave mestra $s \in \mathbb{Z}/q\mathbb{Z}$ e os parâmetros públicos do sistema $\mathsf{params} = \langle q, G, G_T, e, n, P, P_{pub}, H_1, H_2, H_3, H_4 \rangle$, onde G e G_T são grupos de ordem prima q, $e : G \times G \to G_T$ é um emparelhamento bilinear admissível, P é gerador de G, $P_{pub} = sP$ e H_i são funções de hash, tais que:

$H_1 : \{0,1\}^* \to G^*$

$H_2 : G_T \to \{0,1\}^n$

$H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$

$H_4 : \{0,1\}^n \to \{0,1\}^n$

O espaço de mensagens é $\mathcal{M} = \{0,1\}^n$. O espaço de textos cifrados é $\mathcal{C} = G^* \times \{0,1\}^n \times \{0,1\}^n$.

A entidade A escolhe aleatoriamente sua chave secreta $t_A \in \mathbb{Z}_q^*$ e calcula sua chave pública $N_A = t_A P$, como no modelo convencional.

certifica A identificação do usuário A é A_{info}, que combina um identificador $ID_A \in \{0,1\}^*$, com N_A. A autoridade calcula $P_A = H_1(P_{pub}, i, A_{info})$, para um período i. O certificado $Cert_A = sP_A$ é enviado à entidade A, que calcula sua chave de decifração (válida para o período i): $S_A = Cert_A + t_A P'_A$, onde $P'_A = H_1(A_{info})$.

cifra Dados um texto $m \in \mathcal{M}$, um identificador A_{info}, um período de tempo i e params:

Calcular o texto cifrado $\langle rP, \sigma \oplus H_2\left((e(P_{pub}, P_A)e(N_A, P'_A))^r\right), m \oplus H_4(\sigma) \rangle$, onde $r = H_3(m, \sigma)$, para $\sigma \in \{0,1\}^n$, escolhido aleatoriamente, e P_A e P'_A são calculados como em **certifica**.

decifra Dados $C = \langle U, V, W \rangle \in \mathcal{C}$, a chave secreta S_A e params:

Calcular $V \oplus H_2(e(S_A, U)) = \sigma$ e $W \oplus H_4(\sigma) = m$. Com $r = H_3(\sigma, m)$, verificar se $U = rP$. Se for, m é a resposta, senão C é rejeitado.

Gentry adotou a assinatura agregada do esquema baseado em identidade de Boneh *et al.* (2003), para usar como chave de decifração: o valor de assinatura S_A agrega os segredos da autoridade e do usuário, permitindo a correta inversão da operação criptográfica.

A definição formal de cifragem no modelo baseado em certificado foi revista em 2005 por Al-Riyami e Paterson; modificações sutis também foram acrescentadas no modelo de adversários. A partir de então, todos os esquemas de cifragem baseados em certificado seguem as definições e modelo de segurança desse trabalho.

Ainda no trabalho de Al-Riyami e Paterson (2005), é dada uma prova de que, dado um esquema de cifragem sem certificado seguro, é possível construir um esquema seguro baseado em certificado. Contudo, Kang e Park (2005) apontam uma falha na demonstração que invalida o resultado. Os autores lembram que também Yum e Lee (2004) apresentaram uma demonstração de equivalência entre os modelos, posteriormente invalidada, e sugerem que essa sequência de insucessos pode ser um indício de que cada um dos conceitos tenha vantagens próprias, apesar dos vários aspectos em comum.

O trabalho de Dodis e Katz (2005) apresenta uma série de resultados importantes para a criptografia de chave pública. Além de formalizar segurança de cifragem múltipla (encadeada), os autores propõem construções genéricas para composição segura de esquemas de cifragem (PKE, no modelo convencional) e estudam aplicações decorrentes. Uma das aplicações apontadas resulta na primeira construção genérica de um CBE seguro, a partir de um IBE e um PKE seguros. Todas as demonstrações de segurança são feitas no modelo padrão, sem oráculos aleatórios.

Uma descrição simplificada da construção de Dodis e Katz é dada por Galindo *et al.* (2008): para cifrar uma mensagem m para um usuário com identidade ID, para um período i:

- Gerar um par de chaves (vk, sk) de um esquema de assinatura de uso único (OTS, one-time signature);
- Quebrar m em duas partes m_1 e m_2 tais que $m = m_1 \oplus m_2$;
- Cifrar m_1 usando o esquema IBE, com identidade $id||i$ e label vk, gerando C_1;
- Cifrar m_2 usando o esquema PKE, com label vk, gerando C_2;
- Assinar (C_1, C_2) com a chave sk, gerando σ;
- Texto cifrado de saída é $C = (vk, C_1, C_2, \sigma)$.

Na tentativa de concretizar um esquema de cifragem mais eficiente, Galindo *et al.* (2008) adotaram uma estratégia diferente e conseguiram reduzir a cifra para $C = (vk, C_1, \sigma)$, embutindo C_2 em C_1, e a demonstraram segura sem oráculos aleatórios.

Um terceiro esquema CBE demonstrado seguro no modelo padrão foi dado em Liu e Zhou (2008). Este, entretanto, usa como hipótese a dificuldade de um problema computacional pouco conhecido.

Outra construção genérica de CBE a partir de um PKE e de um IBE seguros é descrita em Lu e Li (2008), que usa a transformação de Fujisaki e Okamoto (1999) para alcançar segurança contra ataques de texto cifrado escolhido (mesma estratégia usada por Boneh e Franklin (2003) e Gentry (2003) e tantos outros). As demonstrações acontecem sob oráculos aleatórios.

Posteriormente, Lu *et al.* (2009) apresentam um esquema concreto eficiente, baseado nessa última construção genérica e na geração de chaves de Sakai e Kasahara (2003). Como resultado é obtido um esquema de cifragem mais eficiente que o de original de Gentry (2003), ao custo da hipótese de dificuldade de um problema computacional menos estudado.

2.5.3 Assinatura no Modelo Baseado em Certificado

A primeira formalização de assinatura para o modelo baseado em certificado foi feita por Kang *et al.* (2004). Nesse mesmo artigo, foi proposto um esquema concreto de assinatura, que posteriormente foi demonstrado inseguro a um ataque de substituição de chave pública, por Li *et al.* (2007).

Esses últimos autores fortaleceram o modelo de adversário, fornecendo a ele o poder de substituição de chaves públicas. No jogo com o adversário, o simulador do sistema permite que as chaves públicas sejam substituídas por valores à escolha do adversário, para quaisquer usuários. Se for garantido que o impostor não obtém certificado válido para uma falsa chave, ele não terá condições de forjar assinatura em um esquema demonstrado seguro.

Na prática, o novo modelo de segurança para assinatura baseada em certificado se tornou mais próximo do proposto por Al-Riyami e Paterson (2003), para o modelo sem certificado. Os trabalhos subsequentes relacionados a assinaturas seguem esse modelo de adversário fortalecido. Curiosamente, os mais recentes esquemas de cifragem baseados em certificado ainda não pressupõem substituição de chaves.

Além de melhorar os requisitos de segurança para assinatura, Li *et al.* (2007) apresentaram esquema concreto com emparelhamentos bilineares, com demonstração de segurança no novo modelo, sob oráculo aleatório.

Para completar um leque de possibilidades, Liu *et al.* (2008) apresentaram mais duas assinaturas: uma sem emparelhamentos, com o objetivo de maximizar desempenho computacional; e outra, sem oráculos aleatórios, para fornecer uma solução no mais alto nível de segurança. O primeiro esquema de assinatura, sem emparelhamentos, foi comparado a outros e os autores afirmaram superioridade em desempenho; a demonstração de segurança foi baseada no modelo de oráculos aleatórios. O segundo esquema, demonstrado seguro no modelo padrão, faz uso de emparelhamentos bilineares; é teoricamente mais seguro e perde em desempenho, quando comparado aos demais, apenas por conta da função de hash concretizada por Waters (2005).

Variações sobre assinatura também foram estudadas para o modelo: Au *et al.* (2007a) mostram assinatura em anel; Shao (2008) propõe um esquema de assinatura cifrada verificável; e Liu *et al.* (2009) elaboram esquema de assinatura agregada baseada em certificado, em que n mensagens podem ser assinadas por n participantes, com comprimento fixo, independente de n.

Tanto a assinatura agregada de Liu *et al.* (2009) quanto a assinatura sem emparelhamentos de Liu *et al.* (2008) são sugeridas para uso em ambientes com restrição de recursos computacionais ou de banda, como redes sem fio (de sensores ou dispositivos móveis).

Todos os esquemas citados pressupõem que a autoridade do sistema gera honestamente os

parâmetros do sistema. Isto é, até a elaboração deste texto, nenhum trabalho levou em consideração a ação de uma autoridade que gera os parâmetros públicos de modo a obter vantagens em conseguir, sem ser detectada, falsificar assinaturas ou decifrar mensagens de usuários. Au *et al.* (2007b) fazem tal alerta no contexto do modelo sem certificado, mas também se aplica ao modelo baseado em certificado.

2.5.4 Aplicações do Modelo Baseado em Certificado

Uma aplicação interessante que o modelo de Gentry possibilita é uma construção mais elegante de um sistema de *proxy* em que é possível revogar o poder de decifração de um "procurador", ainda durante o período de validade da chave dada a ele. O modelo formal e a segurança de um esquema concreto foram analisados por Wang *et al.* (2007).

Basicamente, uma aplicação que possa ser realizada em uma ICP convencional, pode ser implementada no modelo de Gentry. A escolha entre um modelo ou outro depende essencialmente da conveniência ou não da característica de renovação de certificados para tratar revogação.

O modelo baseado em certificado requer implementações mais semelhantes às de um sistema convencional de chave pública. São poucas as diferenças existentes entre uma ICP e a infraestrutura de gerenciamento de certificados do modelo de Gentry. Por esse motivo, é mais fácil o reaproveitamento de módulos prontos ou de bibliotecas.

2.6 Comparações Gerais

Conforme já indicamos na seção anterior, alguns pesquisadores exploraram as semelhanças existentes entre os modelos aqui estudados, para tentar construir conversões genéricas de um para o outro. Todas as tentativas foram posteriormente invalidadas, devido a alguma falha nas demonstrações de conversão. Exemplos disso podem ser vistos nos trabalhos de Al-Riyami e Paterson (2005); Yum e Lee (2004), contestados respectivamente por Kang e Park (2005); Libert e Quisquater (2006). No primeiro desses trabalhos, chegou-se a mostrar (falsamente) que cifragem nos modelos baseado em identidade, sem certificado e baseado em certificado (IBE, CLE e CBE) são essencialmente equivalentes entre si, isto é, dado um deles se constrói os outros dois.

Portanto, pelo menos até o momento, cada um dos paradigmas aparentemente possui vida própria, com propriedades, prós e contras próprios.

Abstraindo-se as diferenças relacionadas com os detalhes de gerenciamento de chaves, podemos classificar e ordenar os modelos de chave pública da seguinte forma:

- num extremo, fica o modelo baseado em identidade, com nível de confiança 1;
- no outro extremo, fica o modelo convencional sobre ICP, com nível de confiança 3;
- no meio ficam os demais, híbridos dos dois primeiros, com níveis de 1 a 3.

Os atributos de criptografia de chave pública, discutidos ao longo do texto, encontram-se resumidos na Tabela 2.6. Tais atributos são compostos pelo par de chaves (pública e secreta) e pela garantia de que o par se relaciona com seu dono, identificado por ID. A função f em cada ocorrência na tabela é uma representação genérica de função matemática; possui propriedades específicas em cada caso.

Tabela 2.6: *Atributos dos modelos de criptografia de chave pública.*

Atributos	Modelos				
	Com ICP	**Baseado em Identidade**	**Auto-certificado**	**Sem Certificado**	**Baseado em Certificado**
Chave secreta	s	$s = f(ID, s_{AC})$	s	$s = [x, f(ID, s_{AC})]$	$[s, c]$
Chave pública	$P = f(s)$	ID	$P = f(ID, s, s_{AC})$	$[f(x), ID]$	$[P = f(s), ID]$
Garantia	$f(ID, P, s_{AC})$	s	P	s	$c = f(ID, P, s_{AC}, i)$

Na Tabela 2.7, algumas das propriedades dos modelos são colocadas lado a lado para comparação. Nela, a segunda coluna refere-se ao modelo convencional sobre ICP. Em geral, um "sim" é considerado um ponto positivo; um "não", ponto negativo.

A primeira propriedade, "Dispensa ICP", diz respeito à infraestrutura tradicional de gerenciamento de chaves públicas. Naturalmente, todos os modelos necessitam de algum tipo de infraestrutura, cujos requisitos e características são retratados pelas demais linhas da tabela. A propriedade "Fluxo criptográfico" se refere ao uso da chave pública antes da emissão da chave secreta. Não incluímos na tabela itens sobre revogação de chave e tratamento de homônimos, pois concluímos que nenhum dos modelos possui boa solução para esses problemas.

Tabela 2.7: *Comparação entre os modelos de criptografia de chave pública.*

Propriedades	Modelos				
	Com ICP	**Baseado em Identidade**	**Auto-certificado**	**Sem Certificado**	**Baseado em Certificado**
Dispensa ICP	não	sim	sim	sim	não
Dispensa certificados	não	sim	sim	sim	não
Dispensa diretório de chaves ou certificados	não	sim	não	não	não
Certificação explícita	sim	não	não	não	não
Nível de confiança	3	1	3(*)	2	3
Sem custódia de chaves	sim	não	sim	sim	sim
Chave secreta criada integralmente pelo usuário	sim	não	sim	não	não
Irretratabilidade	sim	não	sim	sim(*)	sim(*)
Risco da chave mestra	alto	altíssimo	alto	alto	alto
Dispensa canal seguro para distribuir chaves	sim	não	sim	não	sim
Renovação de chaves controlada pelo usuário	não	não	não	sim	não
Fluxo criptográfico	não	sim	não	sim	não
Resistente a ataque *DoD*	sim	sim	sim	não	sim
Principais protocolos demonstrados seguros	sim	sim	não	sim	sim

(*) Sob as condições discorridas no texto

A análise dos modelos alternativos leva-nos a concluir que as diferenças básicas entre eles decorrem do fato de ser permitido ao adversário corromper o registro da chave pública (não permitido no modelo convencional nem no baseado em identidade) e de um processo intrínseco de prova de posse de segredo (existente no convencional e baseado em certificado). As propriedades listadas na Tabela 2.7 são fundamentalmente decorrentes desses fatos.

2.7 Síntese e Considerações Finais

Neste capítulo foram discutidas as propriedades de quatro modelos de criptografia de chave pública que são alternativos ao convencional, por minimizarem algumas das dificuldades impostas pela infraestrutura de chaves públicas. A partir da análise conceitual, foram levantados pontos fortes e fracos de cada modelo, e discutidas algumas possíveis aplicações.

Os quadros comparativos apresentam um sumário dos parâmetros que constroem cada modelo e uma síntese das características relevantes, que são meras consequências da variações exercidas sobre esses parâmetros.

É possível melhorar alguns aspectos em cada modelo, eliminando uma característica ruim ou acrescentando uma propriedade atraente. Refinamentos assim tem sido estudados, mas há pontos ainda em aberto, em todos os modelos. Pudemos observar que a composição dos atributos de criptografia de chave pública comprometem a caracterização de cada modelo. Variações sobre as composições atuais podem induzir novas variantes ou aprimoramentos.

No capítulo a seguir, lembraremos alguns fundamentos teóricos e formalizaremos definições relacionadas com o modelo de chave pública sem certificado que serão usados no restante deste trabalho.

Capítulo 3

Fundamentos Teóricos

O objetivo deste capítulo é apresentar princípios teóricos e formalizações que são usados nesta tese. As variantes 1 e 2 do teste do *trapdoor* BDH, além das variantes BDH*, BDH$^+$, Gap-BDH* e Gap-BDH$^+$, todas descritas neste capítulo, foram produzidas por nós para serem usadas como ferramentas nos capítulos seguintes. Descrevemos variações de formulação de chave pública sem certificado e sugerimos uma interpretação dessas variações relativamente aos demais modelos.

3.1 Segurança Demonstrável

A teoria da complexidade computacional pode ser usada como ferramental para se definir segurança de protocolos criptográficos de chave pública. Sob a ótica da segurança baseada em reduções de problemas computacionais (conhecida por segurança demonstrável), quando se diz que um sistema pode ser provado seguro, significa que é possível mostrar um limite mínimo necessário sobre a quantidade de passos computacionais para se quebrar esse sistema. Em linhas gerais, para se dizer que um protocolo é seguro, os seguintes passos devem ser cumpridos:

1. definir um modelo de segurança;
2. escolher um problema computacional suposto difícil;
3. escrever uma redução polinomial;
4. interpretar a redução.

O modelo de segurança é um conjunto de definições formais que podem ser agrupadas da seguinte forma:

- definições sobre o poder do adversário: ações que são permitidas ao adversário realizar sobre o sistema;
- definições dos objetivos do adversário: o que o adversário deve extrair do sistema;
- definição de protocolo seguro: a partir das definições acima, o que significa segurança de um protocolo perante um adversário.

O adversário é modelado como uma máquina de Turing probabilística, que atua com outra máquina chamada simulador do sistema. Para um conjunto pré-estabelecido de dados no criptossistema e para um dado parâmetro de segurança se inicia uma simulação de ataque (que nada mais é do que um algoritmo cuja complexidade é analisada). O simulador usa os passos do adversário para construir um algoritmo que resolve o problema computacional (escolhido no passo 2). Quando esse algoritmo construído possui complexidade polinomial em k, tem-se uma redução polinomial. Assim, se o problema escolhido é suposto difícil (isto é, não se conhece algoritmo de tempo polinomial que o resolva), é razoável afirmar que o criptossistema é seguro no modelo de segurança escolhido. Depois do passo 3 e antes de se adotar o esquema criptográfico na prática, é necessário analisar a redução e calcular os tamanhos de chave.

Trataremos, mais adiante, sobre os problemas computacionais usados nesta tese. São exemplos de referências sobre segurança demonstrável os trabalhos de Pointcheval (2005) e Castro *et al.* (2007).

A seguir, definimos função negligenciável e o modelo de oráculos aleatórios em segurança demonstrável.

3.1.1 Função Negligenciável

Uma função $f : \mathbb{N} \to \mathbb{R}$ é dita *negligenciável* se para todo polinômio p existe um inteiro n_0, tal que $|f(n)| \leq \frac{1}{|p(n)|}$ para todo $n \geq n_0$.

3.1.2 Modelo do Oráculo Aleatório

Informalmente falando, o modelo de oráculo aleatório, desenvolvido por Bellare e Rogaway (1993), supõe a existência de funções determinísticas que produzem resultados que são indistinguíveis de cadeias verdadeiramente aleatórias. Formalmente, um *oráculo aleatório* é um mapeamento $R : \{0,1\}^* \to \{0,1\}^*$, onde cada bit de $R(x)$ é escolhido independentemente e de modo uniforme, para todo x. A distinguibilidade de um único bit gerado pelo oráculo aleatório é uma função de probabilidade negligenciável.

A validade dessa técnica é bastante discutida na literatura, como se vê nas críticas de, por exemplo, Koblitz e Menezes (2004). Entretanto, o modelo do oráculo aleatório produz protocolos mais eficientes que os gerados no modelo padrão de segurança demonstrável, que evita hipóteses ideais e tão fortes quanto a da existência de oráculos aleatórios. Também permite que um problema seja estudado para que uma solução sem oráculos possa ser elaborada posteriormente. Empregamos o modelo de oráculo aleatório na segurança demonstrável dos protocolos apresentados no Capítulo 5, como ponto de partida para alcançarmos protocolos viáveis na prática, em especial em dispositivos móveis.

3.2 Emparelhamento Bilinear

Grande parte dos protocolos aqui discutidos fazem uso de emparelhamentos bilineares. Vamos, portanto, defini-los.

Sejam G e G_T grupos cíclicos de ordem prima q. Considere G um grupo aditivo e G_T um grupo multiplicativo.

Um mapeamento $e : G \times G \to G_T$ é dito ser um *emparelhamento bilinear admissível* se satisfaz as seguintes propriedades (Boneh e Franklin, 2003):

1. Bilinearidade: $e(aP, bQ) = e(P, Q)^{ab}$, para todo $P, Q \in G$ e $a, b \in \mathbb{Z}_q$;
2. Não-degeneração: o mapeamento não leva todos os pares de $G \times G$ para a identidade de G_T. Como os grupos possuem ordem prima, é equivalente dizer: para um gerador P de G, $e(P, P) \neq 1_{G_T}$, isto é, $e(P, P)$ é um gerador de G_T;
3. Eficiência computacional: existe um algoritmo de complexidade de tempo polinomial que calcula $e(P, Q)$, para todo $P, Q \in G$.

Emparelhamentos assim definidos são ditos *emparelhamentos simétricos* e, em geral, só estão disponíveis sobre curvas supersingulares. Essa definição pode ser generalizada para o que chamamos de *emparelhamentos assimétricos*, com $e : G_1 \times G_2 \to G_T$, em que G_1, G_2 e G_T são grupos de mesma ordem prima e $G_1 \neq G_2$.

3.3 Problemas Computacionais

Considere um emparelhamento bilinear admissível $e : G \times G \to G_T$, conforme acima. Sejam $P \in G$ um gerador de G e valores aleatórios $a, b, c \in \mathbb{Z}_q$. Os seguintes problemas computacionais são supostos difíceis:

BDH. Problema Diffie-Hellman Bilinear: dados $\langle P, aP, bP, cP \rangle$, calcular $e(P, P)^{abc}$.

DBDH. Problema de Decisão Diffie-Hellman Bilinear: dados $\langle aP, bP, cP, T \rangle$, decidir se $e(P, P)^{abc} \stackrel{?}{=} T$.

Gap-BDH. Problema Diffie-Hellman Bilinear Lacunar: dados $\langle P, aP, bP, cP \rangle$, calcular $e(P, P)^{abc}$, com a ajuda de um oráculo de decisão DBDH.

O problema BDH foi definido por Boneh e Franklin (2003); a família de problemas lacunares (Gap) foi apresentada por Okamoto e Pointcheval (2001). Um protocolo que tenha sido provado seguro sob a hipótese de dificuldade do problema BDH é, do ponto de vista da teoria da complexidade, pelo menos tão seguro quanto outro que tenha sido provado seguro sob Gap-BDH, no mesmo modelo de segurança. Por essa razão, é preferível que um protocolo se apoie na suposição de dificuldade do BDH e evite o uso do Gap-BDH, sempre que possível.

3.3.1 Problema Strong Twin Bilinear Diffie-Hellman

O problema *Strong Twin Bilinear Diffie-Hellman* foi definido por Cash, Kiltz e Shoup (Cash *et al.*, 2009) e foi mostrado equivalente ao problema Diffie-Hellman. Os autores desenvolveram uma técnica que pode ser embutida na construção de protocolos para se evitar hipóteses baseadas em Gap. A ideia central da técnica pode ser resumida com o teorema do teste do *trapdoor*, para o qual uma variante sobre BDH é descrita abaixo.

Teorema 3.1 (Teste do Trapdoor BDH – Cash *et al.* (2009)) *Sejam G e G_T grupos de ordem prima q, com $P \in G$ um gerador de G. Seja $e : G \times G \to G_T$ um emparelhamento bilinear admissível. Suponha que B_1, y, z são variáveis aleatórias mutuamente independentes, onde B_1 é tomada*

de G, e y, z são uniformemente distribuídas sobre $\mathbb{Z}_q$. Defina a variável aleatória $B_2 := yP - zB_1$ e suponha que X, Y são variáveis aleatórias tomadas de G. Suponha, ainda, que T_1, T_2 são variáveis aleatórias tomadas de G_T e cada uma delas é definida como uma função de (B_1, B_2). Então valem as afirmações:

(i) B_2 é uniformemente distribuída sobre G;

(ii) B_1 e B_2 são independentes;

(iii) Se $B_1 = b_1 P$ e $B_2 = b_2 P$, então a probabilidade que o valor verdade de

$$T_1{}^z T_2 = e(X, Y)^y \tag{3.1}$$

não corresponde com o valor verdade de

$$T_1 = e(X, Y)^{b_1} \quad \wedge \quad T_2 = e(X, Y)^{b_2} \tag{3.2}$$

é no máximo $1/q$. Ademais, se (3.2) vale, então (3.1) certamente vale.

Para as provas de segurança apresentadas no Capítulo 6, são necessárias duas variantes do teste do *trapdoor*, que passamos a denominar como Variantes 1 e 2, respectivamente descritas nos Teoremas 3.2 e 3.3 abaixo.

Teorema 3.2 (Teste do Trapdoor BDH – Variante 1) *Sejam G e G_T grupos de ordem prima q, com $P \in G$ um gerador de G. Seja $e : G \times G \to G_T$ um emparelhamento bilinear admissível. Suponha que B_1, C_1, y_1, y_2, z são variáveis aleatórias mutuamente independentes, onde B_1, C_1 são tomadas de G, e y_1, y_2, z são uniformemente distribuídas sobre $\mathbb{Z}_q$. Defina as variáveis aleatórias $B_2 := y_1 P - zB_1$ e $C_2 := y_2 P - zC_1$. Suponha que A é uma variável aleatória tomada de G. Suponha, ainda, que T_1, T_2 são variáveis aleatórias tomadas de G_T e cada uma delas é definida como uma função de (A, B_1, C_1) e (A, B_2, C_2). Então valem as afirmações:*

(i) B_2, C_2 são uniformemente distribuídos sobre G;

(ii) $e(B_1, C_1)$ e $e(B_2, C_2)$ são independentes;

(iii) A probabilidade que o valor verdade de

$$\frac{T_2}{{T_1}^{z^2}} = \frac{e(A, P)^{y_1 y_2}}{[e(A, C_1)^{y_1} \cdot e(A, B_1)^{y_2}]^z} \tag{3.3}$$

não corresponde com o valor verdade de

$$T_1 = e(A, P)^{b_1 c_1} \quad \wedge \quad T_2 = e(A, P)^{b_2 c_2} \tag{3.4}$$

é no máximo $1/q$. Ademais, se (3.4) vale, então (3.3) certamente vale.

Demonstração

A prova é similar à encontrada no trabalho de Cash *et al.* (2009).

Observe que $y_1 = zb_1 + b_2$ e $y_2 = zc_1 + c_2$. É imediato verificar que B_2 e C_2 estão uniformemente distribuídas sobre G. Então (i) segue. Uma vez que B_1, C_1, y_1, y_2, z são mutuamente independentes, também o são: (B_1 e B_2), (C_1 e C_2), (B_2 e C_2). Por e ser emparelhamento bilinear admissível, tem-se que $e(B_1, C_1)$ e $e(B_2, C_2)$ são independentes.

Para provar (iii), condicionamos sobre valores fixos de $e(B_1, C_1)$ e $e(B_2, C_2)$. No espaço de probabilidade condicional resultante, z é uniformemente distribuída sobre $\mathbb{Z}_q$, enquanto $b_1c_1, b_2c_2, e(A, P), T_1$ e T_2 são fixos.

Se (3.4) vale, então substituindo as duas equações de (3.4) em (3.3), é possível ver que (3.3) certamente vale.

Por outro lado, se (3.4) não vale, mostramos a seguir que (3.3) vale com probabilidade de no máximo $1/q$. Observe que $y_1y_2 = z^2b_1c_1 + zb_1c_2 + zb_2c_1 + b_2c_2$. De (3.3), temos:

$$\begin{aligned} \frac{T_2}{{T_1}^{z^2}} &= \frac{e(A,P)^{z^2b_1c_1+zb_1c_2+zb_2c_1+b_2c_2}}{e(A,P)^{zc_1(zb_1+b_2)} \cdot e(A,P)^{zb_1(zc_1+c_2)}} \\ &= \frac{e(A,P)^{b_2c_2}}{(e(A,P)^{b_1c_1})^{z^2}} \end{aligned}$$

E isso é o mesmo que

$$\left(\frac{T_1}{e(A,P)^{b_1c_1}}\right)^{z^2} = \frac{T_2}{e(A,P)^{b_2c_2}} \tag{3.5}$$

Se $T_1 = e(A, P)^{b_1c_1}$ e $T_2 \neq e(A, P)^{b_2c_2}$, então (3.5) certamente não vale.

Se $T_1 \neq e(A, P)^{b_1c_1}$ e $T_2 = e(A, P)^{b_2c_2}$, então $\left(\frac{T_1}{e(A,P)^{b_1c_1}}\right)^{z}$ é um elemento aleatório de G_T (uma vez que z é uniformemente distribuída sobre $\mathbb{Z}_q$). O lado esquerdo de (3.5) é um elemento aleatório, enquanto o lado direito é um elemento fixo de G_T. Portanto (3.5) vale com probabilidade máxima de $1/q$. □

Teorema 3.3 (Teste do Trapdoor BDH – Variante 2) *Sejam G e G_T grupos de ordem prima q, com $P \in G$ um gerador de G. Seja $e : G \times G \to G_T$ um emparelhamento bilinear admissível. Suponha que B_1, C_1, y_1, y_2, z são variáveis aleatórias mutuamente independentes, onde B_1, C_1 são tomadas de G, e y_1, y_2, z são uniformemente distribuídas sobre $\mathbb{Z}_q$. Defina as variáveis aleatórias $B_2 := y_1P - zB_1$ e $C_2 := y_2P - zC_1$. Suponha que A é uma variável aleatória tomada de G. Suponha que $(X, Y) = (xP, yP) \in G^2$ e $x, y \in \mathbb{Z}_q^*$. Suponha, ainda, que T_1, T_2 são variáveis aleatórias tomadas de G_T e cada uma delas é definida como uma função de (A, X, Y, B_1, C_1) e (A, X, Y, B_2, C_2). Se $B_1 = b_1P$, $B_2 = b_2P$, $C_1 = c_1P$ e $C_2 = c_2P$, então valem as afirmações:*

(i) $xB_2 + yC_2$ é uniformemente distribuída sobre G_T;

(ii) $xB_1 + yC_1$ e $xB_2 + yC_2$ são independentes;

(iii) A probabilidade que o valor verdade de

$$T_1{}^zT_2 = e(A, X)^{y_1} \cdot e(A, Y)^{y_2} \tag{3.6}$$

não corresponde com o valor verdade de

$$T_1 = e(A, X)^{b_1} \cdot e(A, Y)^{c_1} \quad \wedge \quad T_2 = e(A, X)^{b_2} \cdot e(A, Y)^{c_2} \tag{3.7}$$

é no máximo $1/q$. Ademais, se (3.7) vale, então (3.6) certamente vale.

Demonstração

A prova é similar à do Teorema 3.2. Observe que $y_1 = zb_1 + b_2$ e $y_2 = zc_1 + c_2$. É imediato verificar que B_2 e C_2 estão uniformemente distribuídas sobre G. Então (i) segue. Uma vez que B_1, C_1, y_1, y_2, z são mutuamente independentes, também o são: (B_1 e B_2), (C_1 e C_2), (B_2 e C_2). Como x e y são inteiros diferentes de zero, (ii) segue.

Para provar (iii), fixamos os valores de $xB_1 + yC_1$ e $xB_2 + yC_2$. No espaço de probabilidade condicional resultante, z é uniformemente distribuída sobre $\mathbb{Z}_q$, enquanto $(xb_1 + yc_1), (xb_2 + yc_2), e(A, P), T_1$ e T_2 são fixos.

Se (3.7) vale, então substituindo as duas equações de (3.7) em (3.6), é possível ver que (3.6) certamente vale:

$$\begin{aligned} T_1{}^z T_2 &= [e(A,X)^{b_1} \cdot e(A,Y)^{c_1}]^z \cdot e(A,X)^{b_2} \cdot e(A,Y)^{c_2} \\ &= e(A,X)^{zb_1+b_2} \cdot e(A,Y)^{zc_1+c_2} \\ &= e(A,X)^{y_1} \cdot e(A,Y)^{y_2} \end{aligned}$$

Por outro lado, se (3.7) não vale, mostramos a seguir que (3.6) vale com probabilidade de no máximo $1/q$. De (3.6), temos:

$$\begin{aligned} T_1{}^z T_2 &= e(A,X)^{y_1} \cdot e(A,Y)^{y_2} \\ &= e(A,X)^{zb_1+b_2} \cdot e(A,Y)^{zc_1+c_2} \\ &= e(A,X)^{b_2} \cdot e(A,Y)^{c_2} \cdot [e(A,X)^{b_1} \cdot e(A,Y)^{c_1}]^z \end{aligned}$$

E isso é o mesmo que

$$\left(\frac{T_1}{e(A,X)^{b_1} \cdot e(A,Y)^{c_1}}\right)^z = \frac{e(A,X)^{b_2} \cdot e(A,Y)^{c_2}}{T_2} \tag{3.8}$$

Se $T_1 = e(A,X)^{b_1} \cdot e(A,Y)^{c_1}$ e $T_2 \neq e(A,X)^{b_2} \cdot e(A,Y)^{c_2}$, então (3.8) certamente não vale.

Se $T_1 \neq e(A,X)^{b_1} \cdot e(A,Y)^{c_1}$ e $T_2 = e(A,X)^{b_2} \cdot e(A,Y)^{c_2}$, o lado esquerdo de (3.8) é um elemento aleatório de G_T (uma vez que z é uniformemente distribuída sobre $\mathbb{Z}_q$), enquanto o lado direito é um elemento fixo de G_T. Portanto (3.8) vale com probabilidade máxima de $1/q$. □

3.3.2 Variantes dos Problemas BDH e Gap-BDH

No transcorrer do Capítulo 6, faremos uso de variantes dos problemas BDH e Gap-BDH. Essas variantes foram por nós definidas, para tornar mais claras algumas passagens das provas de segurança. A seguir, definimos as variantes e mostramos que elas são equivalentes aos problemas originais.

Seja $e : G \times G \to G_T$ um emparelhamento bilinear admissível, com G e G_T grupos de ordem prima q, P gerador de G e valores aleatórios $a, b, c \in \mathbb{Z}_q$ e $r, s \in \mathbb{Z}_q^*$. Defina os seguintes problemas computacionais:

BDH*: dados $\langle aP, bP, cP, rP, sP\rangle$, calcular $e(sP, abP) \cdot e(rP, acP)$.

BDH$^+$: dados $\langle aP, bP, cP, rP, sP\rangle$, calcular $e(sP + cP, raP + abP)$.

Gap-BDH*: dados $\langle aP, bP, cP, rP, sP\rangle$, calcular $e(sP, abP)\cdot e(rP, acP)$, com ajuda de um oráculo de decisão-BDH*, que decide se $e(vP, xyP)\cdot e(uP, xzP) \stackrel{?}{=} T$, para dados $(xP, yP, zP, uP, vP, T) \in G^5 \times G_T$.

Gap-BDH$^+$: dados $\langle aP, bP, cP, rP, sP\rangle$, calcular $e(sP+cP, raP+abP)$, com ajuda de um oráculo de decisão-BDH$^+$.

Os problemas BDH* e BDH$^+$ são equivalentes ao BDH:

BDH $=_p$ BDH*. É imediato que BDH* $\leq_p$ BDH. Para ver que BDH $\leq_p$ BDH*, suponha que existe um algoritmo $\mathcal{A}$ que resolve o BDH*; vamos construir um algoritmo $\mathcal{S}$ que resolve o BDH. $\mathcal{S}$ recebe como entrada (xP, yP, zP), escolhe $u, v \in \mathbb{Z}_q^*$, submete (xP, yP, uP, vP, zP) para $\mathcal{A}$ e recebe como resposta $T = e(zP, xyP)\cdot e(vP, xuP)$. $\mathcal{S}$ devolve $T\cdot e(vP, xP)^{-u}$ como solução.

BDH $=_p$ BDH$^+$. Para ver que BDH $\leq_p$ BDH$^+$, suponha que existe um algoritmo $\mathcal{A}$ que resolve o BDH$^+$; vamos construir um algoritmo $\mathcal{S}$ que resolve o BDH. $\mathcal{S}$ recebe como entrada (xP, yP, zP), escolhe $u, v \in \mathbb{Z}_q^*$, submete (xP, yP, zP, uP, vP) para $\mathcal{A}$ e recebe como resposta $T = e(vP + zP, uxP + xyP) = e(vP, xyP)\cdot e(zP, uxP)\cdot e(zP, xyP)\cdot e(vP, uxP)$. $\mathcal{S}$ devolve $T\cdot e(xP, yP)^{-v}\cdot e(zP, xP)^{-u}\cdot e(vP, xP)^{-u}$ como solução. BDH$^+$ $\leq_p$ BDH segue de forma análoga.

Dessas equivalências, segue que Gap-BDH* e Gap-BDH$^+$ são equivalentes a Gap-BDH.

3.4 Chave Pública sem Certificado e Formalizações

O objetivo desta seção é reunir definições formais relacionadas ao modelo de criptografia de chave pública sem certificado.

A definição original de Al-Riyami e Paterson (2003) para esquema de cifragem sem certificado (CLE, ou *Certificateless Encryption*) inclui um conjunto de sete algoritmos, dos quais nos interessa rever aqueles que estabelecem os parâmetros do sistema e chaves de usuários. São eles:

- Setup: para geração dos parâmetros dos sistema;
- ExtractPartialPrivateKey: para calcular a chave secreta parcial de um usuário;
- SetSecretValue: para gerar o valor secreto de usuário;
- SetPublicKey: para calcular a chave pública a partir do valor secreto de usuário;
- SetPrivateKey: para gerar a chave secreta completa de usuário, a partir da chave secreta parcial e do valor secreto.

Posteriormente, vários autores, dentre eles Cheng e Comley (2005), propuseram uma simplificação e reuniram os algoritmos SetSecretValue, SetPublicKey e SetPrivateKey em um único, chamado SetUserKeys. As etapas de geração de parâmetros e chaves de usuário ficam resumidas a três: Setup, ExtractPartialPrivateKey e SetUserKeys. Essa simplificação implica funcionalidade equivalente à original. No entanto, do ponto de vista dos modelos de segurança, algum impacto pode ser percebido.

Dent (2010) observa três implementações diferentes dos algoritmos acima, em que a sequência de uso dos valores de chave influencia nas propriedades dos protocolos sem certificado. O autor define a seguinte nomeação para essas três implementações:

- formulação AP: refere-se à definição de Al-Riyami e Paterson (2005);
- formulação BSS: refere-se à proposta de Baek, Safavi-Naini e Susilo (Baek *et al.*, 2005);
- formulação LK: refere-se à variante de Lai e Kou (2007).

Passemos a descrever cada uma delas.

3.4.1 Formulação AP

Na formulação AP (Al-Riyami e Paterson, 2003), um usuário pode gerar sua chave pública a qualquer momento, pois independe da chave parcial secreta informada pelo KGC. Uma consequência disso é que a chave pública pode ser criada e usada antes mesmo da chave secreta completa existir.

Vamos formalizar o esquema que envolve apenas três algoritmos, ou seja a versão simplificada de Al-Riyami e Paterson (2003) para geração de chaves e parâmetros:

- Setup(1^k): Este algoritmo é executado pelo KGC; recebe como entrada um parâmetro de segurança de comprimento k e gera como saída os parâmetros públicos do sistema $params$ (que incluem a chave mestra pública mpk) e a chave mestra secreta msk. A chave mestra secreta é mantida em sigilo e é usada pelo KGC para calcular as chaves secretas parciais.
- ExtractPartialSecretKey($params, msk, ID$): Este algoritmo é executado pelo KGC; recebe como entrada $params$, a chave mestra secreta msk e a identidade de um usuário $ID \in \{0,1\}^*$; gera como saída uma chave parcial secreta d_{ID} que é entregue confidencialmente ao dono de ID.
- SetUserKeys($params, ID$): Este algoritmo é executado pelo dono de ID; recebe como entrada $params$ e uma identidade ID; devolve como saída um valor secreto x_{ID} e a chave pública pk_{ID}. O valor dessa chave pública é divulgado (sem autenticação por meio de certificado digital), enquanto o usuário deve manter em sigilo o valor x_{ID}.

3.4.2 Formulação BSS

Na formulação BSS (Baek, Safavi-Naini e Susilo), a chave pública só pode ser gerada depois do usuário receber a chave parcial secreta do KGC. E a chave parcial secreta é obtida com apenas *uma* mensagem segura transmitida pelo KGC. Os algoritmos para estabelecimento dos parâmetros e chaves são:

- Setup(1^k): Este algoritmo é idêntico ao caso AP.
- ExtractPartialSecretKey($params, msk, ID$): Este algoritmo é idêntico ao caso AP.
- SetUserKeys($params, ID, d_{ID}$): Este algoritmo difere do caso AP, pois recebe adicionalmente como dado de entrada o valor da chave parcial secreta. É executado pelo dono de ID; devolve como saída a chave secreta (completa) sk_{ID} e a chave pública pk_{ID}. O valor dessa chave

pública é divulgado (sem autenticação), enquanto o usuário deve manter em sigilo sk_{ID}. Note que não existe valor secreto (x_{ID}) como no caso AP.

A formulação BSS foi desenvolvida com o propósito de viabilizar um protocolo CLE sem emparelhamento e acabou por introduzir inadvertidamente proteção contra ataque de *Denial of Service* (Seção 2.4.1, pág. 27). O trabalho de Baek *et al.* (2005) contém uma falha da demonstração de segurança, mas não se sabe se o protocolo é inseguro. Sun *et al.* (2007) apresentam uma versão segura para cifragem sem emparelhamento e sem certificado, porém menos eficiente.

3.4.3 Formulação LK

Na formulação LK (Lai e Kou, 2007), a chave pública só pode ser gerada depois do usuário completar um protocolo de comunicação com o KGC, em que ambos (usuário e autoridade) enviam mensagens entre si. Trata-se de uma generalização do caso BSS para mais de uma mensagem trocada. Foi um predecessor do trabalho de Liu *et al.* (2007), para tratar o ataque *Denial of Decryption*. Os algoritmos para estabelecimento dos parâmetros e chaves são os seguintes:

- Setup(1^k): Este algoritmo é idêntico aos casos AP e BSS.
- KGCKeyGen($params, msk, ID$) e UserKeyGen($params, ID$) são algoritmos interativos. O KGC executa KGCKeyGen usando a chave mestra secreta e ID como dados de entrada. O usuário (responsável por ID) executa UserKeyGen usando a chave mestra pública e ID. Se o protocolo se encerra com sucesso, o algoritmo UserKeyGen produz como saída um par (pk_{ID}, sk_{ID}); o algoritmo KGCKeyGen não fornece dado de saída. O valor pk_{ID} é divulgado, enquanto o usuário deve manter em sigilo sk_{ID}. Note que aqui também não existe valor secreto (x_{ID}).

3.4.4 Formulação de Chave e Modelos de Segurança

Para se modelar adversários contra esquemas sobre cada uma das formulações de chave, é preciso levar em consideração que os valores que tiverem que ser armazenados ou transmitidos em algum momento podem ser corrompidos.

Na formulação AP, um adversário poderoso deve, em princípio, poder ter acesso a:

1. valor de secreto x_{ID} (que gera a chave pública)
2. chave secreta parcial d_{ID} (calculada pelo KGC)
3. ambos componentes

Nas formulações BSS e LK, um adversário deve poder ter acesso a:

1. chave secreta parcial d_{ID} (calculada pelo KGC)
2. chave secreta sk_{ID}

Conforme observado por Dent (2010), um CLE na formulação LK pode ser mostrado equivalente a um esquema de cifragem convencional que requer infraestrutura de chave pública. Portanto, são de interesse prático apenas as formulações AP e BSS. Esta última produz protocolos relativamente mais complexos, porém sem emparelhamentos.

3.4.5 Relações entre as Formulações

A Figura 3.1 mostra uma relação entre as variações de formulação da chave pública sem certificado, relativamente aos outros modelos de chave pública. No nível 1 de confiança na autoridade do sistema (descrito na Seção 2.1.1, pág. 10), situa-se o modelo baseado em identidade (sigla ID na Figura 3.1). No nível 2, encontram-se as variantes sem certificado nas formulações de Al-Riyami e Paterson (2003) e de Baek *et al.* (2005), respectivamente CL-AP e CL-BSS. A formulação sem certificado CL-LK de Lai e Kou (2007) alcança nível 3 e é mostrada funcionalmente equivalente aos que seguem o convencional com ICP. Também atinge o nível 3 de confiança a proposta de Gentry (2003) para chave pública baseada em certificado CB.

A chave pública autocertificada de Girault (1991), que não possui especificação formal de modelo de segurança, e as variações que a sucederam aproximam-se das formulações CL-BSS e CL-LK, que alcançam níveis 2 e 3, respectivamente. Portanto, duas possíveis formalizações para modelo de segurança de chave pública autocertificada seriam as encontradas nos trabalhos de Baek *et al.* (2005) e Lai e Kou (2007), uma vez que nesses dois o cálculo da chave pública embute segredos da autoridade e do usuário.

Os três modelos no nível 3 – CL-LK, CB e ICP – pressupõem um procedimento de prova de posse de segredo no momento do registro do usuário e de sua chave pública, mas apenas esse último junto com ID impõem a premissa de que o processo de registro da chave pública não pode ser corrompido.

3.5 Síntese

Neste capítulo relacionamos alguns fundamentos teóricos que serão usados no restante deste documento. Também descrevemos as variações de chave pública sem certificado em torno do trabalho original de Al-Riyami e Paterson (2003) e contextualizamos essas variantes relativamente aos demais modelos alternativos de chave pública. No capítulo a seguir, trataremos de acordo de chave secreta com autenticação e seus modelos de segurança.

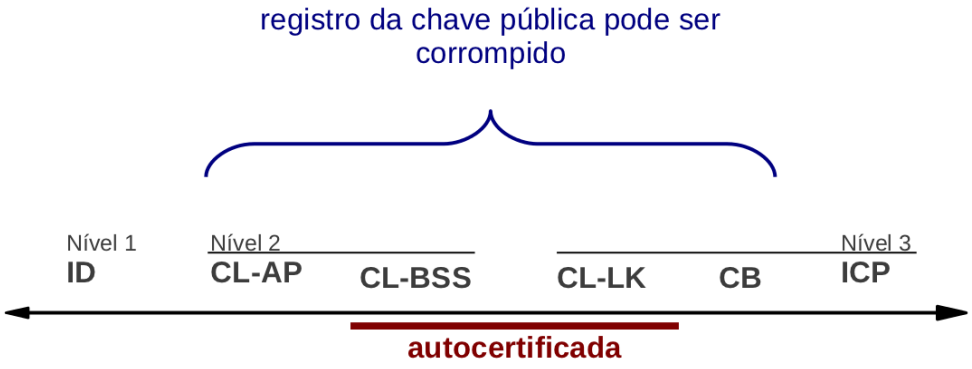


Figura 3.1: *Relações entre os modelos alternativos de chave pública*

Capítulo 4

Acordo de Chave Secreta com Autenticação sem Certificado

Os objetivos deste capítulo são descrever acordo de chave secreta com autenticação e apresentar modelos de segurança para o caso sem certificado, com o envolvimento de dois participantes na comunicação. São contribuições originais desta tese as extensões de segurança encontradas nos modelos SJ^+, Mal-LBG e Mal-SJ^+.

4.1 Protocolos de Acordo de Chave Secreta com Autenticação

Protocolo de acordo de chaves é uma ferramenta primordial para o desenvolvimento de soluções de segurança que requeiram sigilo das comunicações. Com protocolos desse tipo, é possível que participantes em uma comunicação combinem chaves secretas, usando canais públicos. Exemplo clássico deste tipo de protocolo é o de Diffie e Hellman (1976). Protocolos de acordo de chaves com autenticação (AKA, ou *Authenticated Key Agreement*) fazem o mesmo, porém com garantia de autenticidade dos participantes.

Vamos brevemente contrapor as diferenças entre protocolos do tipo AKA, implementados no modelo sobre ICP, baseado em identidade e sem certificado. Não incluímos aqui o caso de protocolos de transporte de chave secreta sobre canal público, que são um caso particular de protocolos de estabelecimento de chave secreta, conforme classificação de Menezes *et al.* (1996).

4.1.1 AKA no Modelo Convencional sobre ICP

Existe uma quantidade relativamente grande de protocolos de acordo de chaves sob o modelo tradicional com suporte de uma ICP, dos quais podemos citar o MQV (Menezes *et al.*, 1995), que recebeu sucessivos aprimoramentos e posteriormente foi padronizado (IEEE, 2000) e chegar à versão HMQV (Krawczyk, 2005). Uma compilação abrangente de protocolos dessa classe é realizada por Boyd e Mathuria (2003). Para uma comparação grosseira de tempo de processamento do MQV (ou HMQV) com um protocolo de chave pública sem certificado seria necessário acrescentar ao MQV, os procedimentos para verificação da assinatura do certificado (eventualmente, obter o certificado antes e verificar as assinaturas no caminho de certificação até a raiz e conferir se o certificado foi revogado). A assinatura e sua verificação tipicamente são realizadas com o RSA ou DSA.

4.1.2 AKA Baseado em Identidade

Protocolos de acordo de chave baseado em identidade (ID-AKA) dispensam o uso de certificados digitais de chave pública, pressupondo-se a existência de uma autoridade de confiança que possa estabelecer um canal seguro com seus usuários, para distribuir chaves secretas. Chen, Cheng e Smart apresentam uma análise de quase vinte protocolos de acordo de chave baseado em identidade (Chen *et al.*, 2007), a partir da qual é possível identificar duas situações de risco no uso de protocolos nesse modelo:

1. quando a autoridade do sistema age de forma mal intencionada e faz uso de seu conhecimento privilegiado das chaves secretas de todos os usuários;

2. quando a chave mestra secreta do sistema é comprometida.

Nesses dois casos, o vazamento do segredo temporário de uma sessão é suficiente para a recuperação da chave de sessão por quem detiver o conhecimento da chave mestra secreta. Fonte de vazamento de segredos temporários pode ser, por exemplo, funções geradoras de números pseudo-aleatórios fracas ou armazenamento de forma insegura de valores aleatórios previamente calculados (este último caso é prática corrente especialmente em aplicações que envolvem dispositivos de menor capacidade de processamento).

4.1.3 AKA sobre Chave Pública sem Certificado: CL-AKA

Uma motivação para o emprego de protocolos de acordo de chave com autenticação sem certificado (CL-AKA, ou *Certificateless Authenticated Key Agreement*) é que as consequências dos riscos acima são atenuadas, pois cada usuário possui três componentes secretos que precisam ser corrompidos (um a mais que no caso baseado em identidade):

1. o valor secreto que gerou a chave pública (de conhecimento exclusivo do usuário);

2. a chave parcial secreta (calculada pelo KGC e compartilhada com o usuário);

3. o segredo temporário de sessão (um número pseudo-aleatório, sorteado para o cálculo da chave de sessão).

Um KGC mal intencionado atuando sobre um protocolo de acordo de chaves sem certificado, ou um adversário que tenha acesso à chave mestra secreta, precisa corromper outros dois segredos adicionais: o valor secreto gerado pelo usuário e o segredo temporário. É razoável esperar que um protocolo CL-AKA se mantenha seguro enquanto um participante na sessão de comunicação não for integralmente corrompido. Isto é, se ao menos um desses três componentes secretos for mantido em sigilo, a segurança do protocolo não deve ser comprometida.

Diversos protocolos CL-AKA podem ser encontrados na literatura. Um estudo realizado por Swanson (2008) mostra que todos os protocolos propostos até então são inseguros e, por esse motivo, a autora define um modelo de segurança adequado ao caso sem certificados (Swanson e Jao, 2009).

Lippold, Boyd e González Nieto aprimoram o modelo de Swanson e Jao e apresentam o primeiro protocolo CL-AKA demonstrado seguro sob um modelo forte de segurança (Lippold *et al.*, 2009). Esse modelo de segurança, que passaremos a referenciar por LBG, satisfaz a noção de que apenas dois componentes secretos comprometidos não são suficientes para um atacante descobrir uma chave

de sessão. Em particular, o KGC que conhece as chaves parciais secretas de todos usuários, mesmo se tiver acesso aos segredos temporários, ainda terá que comprometer o valor secreto que gerou a chave pública do usuário para ser capaz de recuperar uma chave de sessão.

Tanto o modelo de Swanson e Jao (2009) quanto o de Lippold *et al.* (2009) tratam o caso que envolve dois participantes, mutuamente autenticados, com duas passagens de mensagens entre eles.

Nas seções a seguir, vamos definir protocolo CL-AKA com dois participantes, seus algoritmos, as propriedades de segurança almejadas e descrever os modelos relacionados.

4.2 Protocolo CL-AKA

Um protocolo de acordo de chave com autenticação sem certificado (CL-AKA) para dois participantes possui as seguintes fases:

- fase de inicialização do sistema
- fase de geração de chaves de usuário
- fase de sessão de acordo de chave

A fase de inicialização do sistema é definida essencialmente pelo algoritmo Setup, abaixo, enquanto a fase de geração de chaves de usuário é estabelecida pelos algoritmos ExtractPartialSecretKey e SetUserKeys:

- Setup(1^k): Este algoritmo recebe como entrada um parâmetro de segurança de comprimento k e gera como saída os parâmetros públicos do sistema $params$ (que incluem a chave mestra pública mpk) e a chave mestra secreta msk.
- ExtractPartialSecretKey($params, msk, ID$): Este algoritmo recebe como entrada $params$, a chave mestra secreta msk e a identidade de um usuário $ID \in \{0,1\}^*$; gera como saída uma chave parcial secreta d_{ID}.
- SetUserKeys($params, ID$): Este algoritmo recebe como entrada $params$ e uma identidade ID; devolve como saída um valor secreto x_{ID} e a chave pública pk_{ID}.

Uma sessão de acordo de chave ocorre após dois usuários com identidades A e B já possuírem suas chaves definidas. Suponha que o participante A inicia a comunicação: ele executa um procedimento *Inicia*. Em atendimento ao chamado de A, o participante B executa o procedimento *Responde*. Como estamos considerando apenas o caso de protocolos com duas passagens de mensagens (*two-pass* ou *one-round*), cada um dos participantes envia e recebe uma mensagem de dados para o estabelecimento da chave compartilhada. Após a etapa de troca de mensagens, é possível calcular a chave secreta compartilhada. O conjunto desses passos caracteriza uma sessão de acordo de chave e é sumarizado na Figura 4.1.

4.3 CL-AKA e Propriedades de Segurança

Segundo Krawczyk (2005) e LaMacchia *et al.* (2007), as propriedades de segurança mais importantes e requeridas nos protocolos de acordo de chaves com autenticação são as relacionadas a seguir:

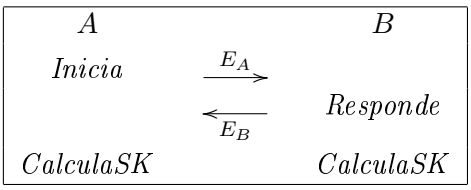


Figura 4.1: *Sessão de acordo de chave entre dois participantes, com duas passagens de mensagens*

- Resistência a ataques de personificação básicos: um adversário não deve ser capaz de personificar um usuário se não conhecer sua chave secreta.
- Resistência a ataques de compartilhamento desconhecido de chave (UKS, ou *Unknown Key-Share*): deve ser inviável convencer um participante A de que ele está compartilhando uma chave com B, quando na realidade está compartilhando com outro usuário C (honestamente registrado no sistema), enquanto C pensa (corretamente) estar compartilhando com A.
- Segurança de chave conhecida: cada execução do protocolo deve produzir uma chave de sessão única. O protocolo deve permanecer seguro mesmo que um adversário descubra algumas chaves de sessão já negociadas.
- Resistência a ataques de personificação pelo comprometimento de chave secreta (KCI, ou *Key-Compromise Impersonation*): se a chave secreta de longa duração de um usuário A é comprometida, um atacante não deve ser capaz de personificar um usuário B perante A.
- Segurança no futuro completa (PFS, ou *Perfect Forward Secrecy*): deve ser inviável para um atacante recuperar uma chave de sessão mesmo que, no futuro, venha a corromper as chaves secretas de longa duração de *todos* os participantes envolvidos naquela sessão.
- Segurança no futuro completa-fraca (wPFS, ou *Weak Perfect Forward Secrecy*): propriedade PFS com a condição de que o adversário não esteja ativamente envolvido na escolha dos segredos temporários para o cálculo da chave de sessão. Krawczyk (2005) demonstrou que wPFS é o melhor que se pode alcançar com relação à segurança no futuro para protocolos com apenas duas passagens de mensagens e sob modelos de segurança que preveem o vazamento de segredos temporários. É o caso de todos os protocolos aqui discutidos.
- Resistência ao vazamento de segredos temporários: o vazamento de um valor secreto temporário não deve comprometer a segurança de sessões que não o tenham usado.

No caso especial em que não são necessários certificados digitais para as chaves públicas, duas propriedades adicionais são desejáveis:

- Segurança no futuro perante o KGC (*KGC Forward Secrecy*): o KGC deve ser incapaz de recuperar chaves de sessão, mesmo que monitore o tráfego durante o estabelecimento das chaves e que, portanto, tenha acesso a todos os dados públicos.
- Resistência ao vazamento de segredos temporários para o KGC: nos protocolos CL-AKA, o KGC ainda que seja capaz de aprender os valores secretos temporários de qualquer sessão (e fazendo uso de seu conhecimento de todas as chaves secretas parciais) deve ser incapaz de recuperar chaves de sessão.

Os modelos de segurança que descreveremos a seguir contemplam essas propriedades.

4.4 Modelos de Segurança para CL-AKA

Nesta seção, classificamos os modelos de segurança para protocolos CL-AKA em função do poder que o adversário tem sob eles. Usamos os adjetivos *forte, fraco* e *moderado*, numa tentativa de qualificar intuitivamente o conjunto de poderes que são dados ao adversário. Nesta tese, um adversário dito *forte* possui os poderes de um adversário dito *moderado*, que, por sua vez, abrange os de um *fraco*. Também usamos a expressão *modelo forte* para nos referirmos a um modelo de segurança cujo adversário seja qualificado como forte. Neste trabalho, um *modelo forte* tem a característica de ser uma extensão de um *modelo fraco*, no sentido de que um protocolo que seja seguro no modelo mais forte também o será no modelo mais fraco.

Nas subseções a seguir, descrevemos os seguintes modelos:

- LBG: modelo de Lippold *et al.* (2009) contra adversário forte
- SJ: modelo de Swanson e Jao (2009) contra adversário moderado
- LG: modelo de Lippold e González Nieto (2010) contra adversário fraco

Posteriormente, na Seção 4.6, propomos as seguintes extensões:

- SJ^+: melhorias no modelo SJ contra adversário moderado
- Mal-LBG: extensão sobre LBG contra autoridade mal intencionada
- Mal-SJ^+: extensão sobre SJ^+ contra autoridade mal intencionada

Esses modelos podem ser relacionados entre si, como se vê representado na Figura 4.2. Pode-se afirmar que o modelo LG é mais fraco de todos, enquanto o modelo Mal-LBG é o mais forte. O modelo Mal-SJ^+ não é comparável a LBG, mas é mais fraco que Mal-LBG.

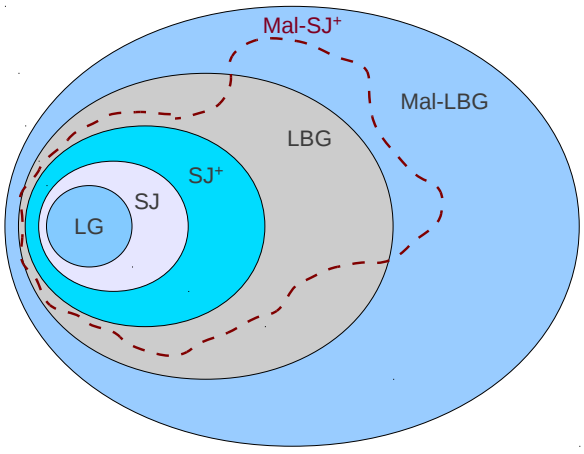


Figura 4.2: *Hierarquia dos modelos de segurança para CL-AKA*

4.4.1 LBG: Modelo de Segurança para CL-AKA contra Adversário Forte

Vamos iniciar a descrição dos modelos de segurança para protocolos de acordo de chave com autenticação sem certificado, tomando como base o trabalho de Lippold *et al.* (2009), tratado aqui por LBG. Nele, a um adversário é permitido:

- substituir a chave pública de um dado usuário *ou* revelar o valor secreto correspondente à chave pública;
- revelar a chave secreta parcial de determinados usuários *ou* revelar a chave mestra secreta do KGC;
- revelar o segredo temporário de uma dada sessão *ou* escolhê-lo ativamente;
- revelar a chave secreta de uma dada sessão;
- interagir de forma adaptativa com o protocolo, iniciando sessões ou registrando novos usuários arbitrariamente.

Formalmente, esse modelo garante a segurança de uma chave de sessão com base na incapacidade de um adversário diferenciar uma chave legítima de um valor escolhido uniformemente ao acaso do espaço de chaves. Esse princípio de indistinguibilidade de chave foi introduzido por Bellare e Rogaway (1994) e foi sucessivamente aprimorado até se chegar às variantes conhecidas como CK01 e eCK, respectivamente propostas por Canetti e Krawczyk (2001) e LaMacchia *et al.* (2007).

Passemos a descrever usuários, sessões e ações do adversário mais formalmente, para então definir um protocolo CL-AKA seguro.

Usuários e Sessões

O modelo de segurança para CL-AKA inclui um conjunto finito de usuários que podem tomar parte em uma comunicação: $\mathcal{U} = \{U_1, \ldots, U_{q_1}\}$, onde q_1 é polinomialmente relacionado com um dado parâmetro de segurança k. Uma sessão entre dois participantes distintos é modelada por uma máquina de Turing probabilística, que é denominada oráculo. Denotamos a $s-$ésima instância do usuário U_i com um parceiro U_j pelo oráculo $\Pi_{i,j}^s$; nesse caso dizemos que U_i é dono da sessão. Um adversário $\mathcal{A}$ também é modelado como uma máquina de Turing probabilística e tem o poder de criar usuários, interagir com as sessões e iniciá-las como bem entender.

Todas as comunicações entre os participantes são controladas pelo adversário. Isto é, os oráculos não se comunicam entre si diretamente; se comunicam apenas com o adversário, que, por sua vez, pode bisbilhotar, atrasar, repetir, modificar, apagar e inserir mensagens no canal. Essas interações ocorrem por meio de consultas a outros oráculos, descritos logo abaixo.

Uma sessão $\Pi_{i,j}^s$ sempre é iniciada em um estado *indefinido* e assume o estado *aceito* quando se encerra com sucesso calculando uma chave de sessão $\mathrm{sk}_{i,j}^s$. Quando uma sessão termina sem ter calculado uma chave, ela assume o estado *rejeitado*.

Oráculos que o Adversário Pode Consultar

No modelo LBG, o adversário pode consultar os seguintes oráculos:

RequestPublicKey(i)**:** é entregue a chave pública do usuário U_i.

EstablishParty(i, pk)**:** a consulta a este oráculo permite ao adversário registrar um usuário totalmente corrompido, sob controle integral do adversário. A chave pública pk informada não necessariamente é uma chave válida e o adversário pode desconhecer o valor secreto que a gerou. Um usuário sobre o qual o adversário não realiza essa consulta é dito *honesto*.

Send$(\Pi^t_{i,j}, x)$**:** se a sessão $\Pi^t_{i,j}$ não existe, então é criada com o papel de remetente e dono como sendo o participante i (se $x = \lambda$) ou como receptor com dono j (se $x \neq \lambda$). Caso os participantes não tenham sido iniciados anteriormente, eles são estabelecidos. O protocolo é executado com a mensagem x e o estado da sessão é atualizado. Nesta tese é considerado apenas o caso em que $i \neq j$.

RevealMasterKey: a chave mestra do sistema é entregue ao adversário.

RevealPartial(i)**:** é entregue a chave parcial secreta do usuário U_i.

RevealSecretValue(i)**:** se a chave pública do usuário U_i tiver sido substituída, a resposta é $\perp$; caso contrário é entregue o valor secreto que gerou a chave pública.

ReplacePublicKey(i, pk)**:** a chave pública do usuário U_i é substituída por pk, que será usada a partir de então por todo participante em toda sessão.

RevealEphemeral$(\Pi^t_{i,j})$**:** é entregue o valor temporário da sessão $(\Pi^t_{i,j})$, cujo dono é U_i.

StrongRevealSessionKey$(\Pi^t_{i,j})$**:** se o estado da sessão não for *aceito*, é devolvido $\perp$; caso contrário o adversário recebe o valor correto da chave de sessão e, nesse caso, diz-se que a sessão foi *revelada*.

Teste$(\Pi^t_{i,j})$**:** se a sessão não for *fresh* (ver definição abaixo), é devolvido $\perp$; caso contrário, um bit $b \in \{0, 1\}$ é escolhido ao acaso; se $b = 0$, é entregue o valor correto da chave de sessão, caso contrário, o adversário recebe um valor aleatório escolhido de dentro do espaço de chaves.

O simulador do sistema é outra máquina de Turing probabilística que trata as consultas aos oráculos realizadas pelo adversário. Simulador e adversário interagem entre si no jogo definido a seguir.

Definição 4.1 *[Jogo-AKA] O jogo entre o adversário e o simulador, denominado Jogo-AKA, é definido em fases:*

- *primeira fase: nela o adversário pode emitir tantas consultas quanto desejar, a qualquer dos oráculos acima descritos, com exceção da consulta de Teste, e em qualquer ordem; o simulador deve responder de acordo com a definição de cada oráculo dada anteriormente;*
- *fase de teste: a primeira fase é encerrada com o adversário escolhendo uma sessão, sobre a qual é realizada a (única) consulta ao oráculo de Teste;*
- *segunda fase: o adversário realiza mais consultas, da mesma forma que na primeira fase;*
- *fase de palpite: o adversário deve emitir um palpite $\hat{b} = 0$, se julgar que a resposta recebida na consulta de Teste for a verdadeira chave da sessão consultada, ou $\hat{b} = 1$ em caso contrário.*

É dito que o adversário ganha o jogo acima se adivinhar corretamente o bit sorteado pelo simulador, isto é, se $b = \hat{b}$. O jogo é abortado sempre que o simulador entregar uma resposta igual a $\perp$, ou sempre que o adversário puder detectar inconsistência em alguma resposta recebida (isto é, quando puder diferenciar entre a simulação e uma execução real do protocolo).

As consultas de corrupção (revelação de algum componente secreto) são agrupadas da seguinte forma:

1. as consultas aos oráculos RevealMasterKey e RevealPartial visam o comprometimento da parte baseada na identidade;
2. as consultas a RevealSecretValue, ReplacePublicKey e EstablishParty desafiam a segurança da chave pública no protocolo;
3. as consultas a RevealEphemeral tentam comprometer uma sessão em particular.

Define-se uma sessão como sendo *totalmente corrompida* se o adversário tiver consultado ao menos um oráculo em cada uma das três categorias acima. Um usuário é dito *totalmente corrompido* se o adversário tiver realizado consultas nas classes 1 e 2 acima. Diz-se que duas sessões possuem *matching* se as mensagens de entrada em uma sessão são iguais às de saída da outra e vice-versa.

Definição 4.2 *[Sessão fresh] Uma sessão $(\Pi^t_{i,j})$ é definida como sendo "fresh" se:*

1. *$(\Pi^t_{i,j})$ tem estado igual a "aceito";*
2. *$(\Pi^t_{i,j})$ não foi revelada;*
3. *nem $(\Pi^t_{i,j})$ e nenhum de seus participantes foi totalmente corrompido;*
4. *nenhuma sessão $(\Pi^u_{j,i})$ que tenha sido revelada possui "matching" com $(\Pi^t_{i,j})$.*

É necessário que o adversário realize uma única consulta ao oráculo de Teste, sobre uma sessão necessariamente *fresh*. Na segunda fase do jogo, o adversário pode realizar quaisquer consultas desde que preserve a característica de *fresh* da sessão consultada em Teste.

A vantagem do adversário $\mathcal{A}$ em vencer o *Jogo-AKA* é definida como:

$$Adv^{\mathcal{A}}(k) = \left| Pr[b = \hat{b}] - \frac{1}{2} \right|$$

Definição 4.3 (Segurança no modelo LBG) *Um protocolo de acordo de chave com autenticação sem certificado é dito seguro no modelo LBG se todo adversário tem vantagem negligenciável em vencer o jogo Jogo-AKA.*

4.4.2 SJ: Modelo de Segurança para CL-AKA contra Adversário Moderado

O modelo de Swanson e Jao (2009) é considerado mais fraco que o de Lippold *et al.* (2009), devido a duas diferenças principais.

A primeira diferença está no tratamento do oráculo que revela para o adversário a chave de uma dada sessão. No trabalho de Swanson e Jao, o usuário continua a usar seu próprio par original de

chaves pública e secreta no cálculo da chave de sessão, mesmo que o adversário tenha substituído a chave pública; nesse caso, os demais usuários calculam a chave de sessão usando a chave pública substituída. No trabalho de Lippold et al., se o adversário substituir uma chave pública, até mesmo o usuário dono passa a usar a nova chave pública escolhida pelo adversário. Esta diferença é equivalente à existente nos modelos *Strong* e *Weak* para cifragem no modelo sem certificado (Dent, 2008) e os protocolos que são seguros no modelo *Strong* também o são no *Weak*. Um adversário contra um esquema CLE, com acesso ao oráculo StrongDecrypt, obtém o texto em claro de uma cifra mesmo que o usuário tenha tido a chave pública substituída (e mesmo que o adversário desconheça o valor secreto que gerou a nova chave pública). De forma análoga, um adversário contra um protocolo CL-AKA sob o modelo LBG que tem acesso ao oráculo StrongRevealSessionKey sempre receberá o verdadeiro valor da chave de sessão. No modelo de Swanson e Jao (2009), o oráculo RevealSessionKey está disponível, em vez do StrongRevealSessionKey. O oráculo RevealSessionKey do modelo SJ pode retornar uma resposta incorreta se a chave pública de um dos participantes tiver sido substituída, pois o usuário dono continuará a usar seu par original público/privado nos cálculos. Nós compartilhamos a opinião de Dent (2008), a respeito do uso dos oráculos Strong, e acreditamos que um protocolo provado seguro no modelo forte não pode ser considerado significativamente melhor que outro provado no modelo moderado.

A segunda diferença entre os modelos SJ e LBG é que o adversário que conhece a chave mestra secreta não pode substituir chaves públicas. Com base na definição de Al-Riyami e Paterson (2003), Swanson e Jao (2009) estabeleceram que um atacante interno, que conhece a chave mestra secreta (também chamado de adversário Tipo II), não pode substituir chaves públicas. Em LBG, ao contrário, o atacante interno pode ainda realizar tal substituição sobre os participantes da sessão de Teste, desde que não comprometa o segredo temporário de sessão. Conforme observado por Yang e Tan (2011a), restrição como essa imposta em SJ pode conflitar com a tentativa de se garantir a propriedade de *Forward Secrecy*. Lembrando que essa noção de segurança implica que um adversário não é capaz de calcular a chave de sessão mesmo que venha a corromper integralmente um dos usuários participantes da sessão, impedir que o KGC substitua chaves públicas significa desconsiderar que essa autoridade possa vir a corromper os participantes no futuro.

Essa segunda diferença entre SJ e LBG pode ser eliminada, alterando-se algumas definições em SJ, sem que seja causado dano às demais definições de Swanson e Jao (2009). Também julgamos necessário outro ajuste no modelo SJ, no qual a definição de usuários "honestos" cria limitações para o adversário e, portanto, diminui a segurança. Swanson e Jao (2009) incluíram na definição de sessão *fresh* a restrição de que seus participantes devem ser "honestos", no sentido de que não foram totalmente corrompidos. Isso também conflita com a propriedade de segurança no futuro, que se deseja garantir. O correto é que uma sessão *fresh* tenha ambos os participantes registrados no sistema de forma honesta, isto é, o usuário criou seu próprio par de chave pública/valor secreto.

As modificações que realizamos no modelo de SJ dão origem a uma extensão, que chamaremos de SJ^+ e que será formalizada na Seção 4.6.1.

4.4.3 LG: Modelo de Segurança contra Adversário Fraco

Lippold e González Nieto (2010) apresentaram uma versão mais fraca de modelo para mostrar a segurança no modelo padrão de uma construção geral de CL-AKA. Nesse modelo mais fraco, que denominamos LG, o adversário tem restrições para corromper sessões ou usuários. Por exemplo,

um dos dois participantes da sessão de Teste não pode ser totalmente corrompido (em LBG, isso é possível se o segredo temporário não for revelado). O adversário também não pode revelar chaves de sessão e nem seus temporários, para os casos em que um de seus participantes tenha a chave pública substituída. E sessões que tenham *matching* com a sessão de Teste não podem ter seu estado interno revelado. Com essas limitações, o modelo fica mais fraco que o de Swanson e Jao (2009), mesmo com o oráculo StrongRevealSessionKey disponível.

Outros Modelos Fracos

Outro modelo de segurança que sabemos ter sido desenvolvido para protocolos CL-AKA foi apresentado por Zhang *et al.* (2010). No entanto, trata-se de um modelo mais fraco que restringe o poder do adversário, como, por exemplo, impedir que um atacante externo revele a chave parcial de um dos participantes da sessão de Teste e não permitir que um adversário interno substitua a chave pública de um dos participantes do Teste. Com essas limitações, protocolos que são demonstrados seguros sob o modelo de Zhang et al. são eficientes, porém na prática não lidam bem com ataques do tipo KCI. Ademais, esse modelo não prevê vazamento de temporários de sessão e nem permite ao adversário revelar o estado interno de sessão, como previsto no modelo de Canetti e Krawczyk (2001).

Cabe ressaltar que modelos de segurança usados em artigos mais antigos, anteriores ao levantamento de Swanson (2008), são inadequados e inseguros para o caso sem certificado.

4.5 Discussão

Ao analisarmos as diferenças que existem entre os modelos de segurança estudados, podemos resumir as seguintes conclusões:

- a diferença prática entre os modelos LBG e SJ é que no primeiro (modelo forte), o adversário pode interferir no registro de novos usuários, enquanto no modelo mais fraco SJ isso não é permitido;
- modelos mais fracos que o moderado não parecem ser úteis na prática, uma vez que ao se restringir o adversário de realizar determinadas ações, na realidade, os pontos fracos do protocolo ficam evidentes, para serem atacados na prática.

4.6 Extensões de Segurança

Nesta seção, descrevemos extensões para os modelos de segurança citados anteriormente. O termo extensão tem o sentido de que as propriedades de segurança do modelo original são preservadas e ampliadas. A primeira extensão se refere a melhorias aplicadas ao modelo de Swanson e Jao (2009). A segunda trata o caso em que o atacante é uma autoridade desonesta, que frauda na geração de parâmetros dos sistema e, posteriormente, age ativamente para comprometer a segurança das chaves de sessão.

4.6.1 SJ$^+$: Modelo de Segurança para CL-AKA contra Adversário Moderado

Os ajustes que realizamos sobre o modelo SJ aumentam o nível de segurança e dão origem ao modelo SJ$^+$, cujo adversário tem poder ainda em nível moderado (relativamente a LBG contra adversário de poder forte).

A primeira modificação é a desabilitação do acesso ao oráculo EstablishParty. No início do jogo, todos os participantes são previamente criados. Alternativamente poderíamos manter o oráculo EstablishParty, mas seria necessário impedir que o oráculo de revelação de chave de sessão pudesse ser consultado sobre sessões em que um de seus participantes fossem desonestamente gerados no sistema.

O oráculo de revelação da chave de uma dada sessão é alterado para:

RevealSessionKey($\Pi_{i,j}^t$)**:** se o estado da sessão não for *aceito*, é devolvido $\perp$; caso contrário o adversário recebe o valor correto da chave de sessão e, nesse caso, diz-se que a sessão foi *revelada*. A chave de sessão é calculada usando-se o valor secreto original de U_i, mesmo que o adversário tenha substituído a chave pública. Em contraste, a chave pública de U_j usada no cálculo da chave de sessão é sempre a última informada pelo adversário.

Substituímos a definição de sessão *fresh* fornecida originalmente por Swanson e Jao (2009) pela Definição 4.2 (pág. 56). E propomos a seguinte definição:

Definição 4.4 (Segurança no modelo SJ$^+$) *Um protocolo de acordo de chave com autenticação sem certificado é dito seguro no modelo SJ$^+$ se todo adversário tem vantagem negligenciável em vencer o jogo Jogo-AKA.*

4.6.2 Extensão contra Autoridade Mal Intencionada

Descrevemos nas seções a seguir as modificações necessárias nos modelos de segurança para prevenir ataques do KGC mal intencionado. Tomamos como ponto de partida o modelo de Lippold *et al.* (2009), mas alterações similares podem ser feitas sobre o modelo de Swanson e Jao (2009).

4.6.3 Mal-LBG: Extensão contra Autoridade Mal Intencionada sobre LBG

O modelo de Lippold *et al.* (2009) especifica dois tipos de adversário, um atacante *externo* e outro *interno*. Nada mudamos nas definições relacionadas ao atacante externo (Tipo I), que é aquele que desconhece a chave mestra secreta, mas pode revelar chaves parciais de entidades à sua escolha, substituir chaves públicas ou revelar segredos temporários de sessão. O atacante externo joga com o simulador o mesmo *Jogo-AKA*, da definição 4.1 (pág. 55).

O atacante interno (Tipo II) conhece a chave mestra secreta e modela o KGC ou um adversário que corrompeu o principal segredo do sistema. Para capturar um comportamento indesejável do KGC em gerar parâmetros de forma desonesta, permitimos que o adversário interno escolha arbitrariamente todos os parâmetros do sistema e os entregue ao simulador do sistema. A chave mestra secreta fica em poder exclusivo do adversário, mas, a rigor, ela pode até nem existir, pois o adversário pode ter criado os parâmetros públicos a partir de uma relação entre outros dados e desconhecer o valor secreto que gera corretamente a chave mestra pública. Por esse motivo, desabilitamos para o adversário interno a consulta aos oráculos que revelam a chave mestra ou a chave parcial de um dado usuário (ou seja, retiramos RevealMasterKey e RevealPartial).

O oráculo StrongRevealSessionKey é alterado para StrongPPK-RevealSessionKey, para que o adversário informe o valor da chave parcial secreta do dono da sessão. Esse valor informado pode ser ilegítimo, porém, se o adversário obtiver vantagem não negligenciável no jogo contra o simulador, é porque é capaz de calcular componentes secretos aos quais não teve acesso; essa vantagem é aproveitada pelo simulador para resolver um problema computacional suposto difícil. O oráculo StrongPPK-RevealSessionKey fornece respostas compatíveis com a chave parcial informada, mesmo que as chaves públicas dos participantes da sessão tenham sido substituídas.

De maneira análoga, é preciso que se modifique o comportamento do oráculo que cria novos usuários (desonestos, isto é, sob total controle do adversário) Em outras palavras, adicionamos o oráculo PPK-EstablishParty, para o qual o adversário informa o valor da chave secreta parcial, além da identidade e chave pública do novo usuário.

O jogo entre o adversário mal intencionado e simulador será chamado de *Jogo-AKA-Mal*. As definições formais para esse caso seguem as empregadas por Dent (2008), que tomou por base o trabalho de Au *et al.* (2007b).

O adversário interno mal intencionado é modelado como uma tripla de algoritmos $\mathcal{M} = (\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2)$ que atua no *Jogo-AKA-Mal* nas seguintes fases:

1. Na primeira fase, o adversário executa $\mathcal{M}_0$ para gerar os parâmetros públicos do sistema. $\mathcal{M}_0$ não pode realizar consultas aos oráculos fornecidos ao adversário (com exceção de eventuais oráculos aleatórios que simulam funções de hash, a depender de cada protocolo).

2. Na segunda fase, o adversário executa $\mathcal{M}_1$ e tem acesso aos oráculos RequestPublicKey, Send, RevealEphemeral, RevealSecretValue, ReplacePublicKey, StrongPPK-RevealSessionKey (e eventuais oráculos aleatórios relacionados com cada protocolo).

3. Ao fim da segunda fase, $\mathcal{M}_1$ exibe a escolha da sessão sobre a qual faz a consulta de Teste. O simulador escolhe um bit $b \in \{0, 1\}$ para decidir se responde a verdadeira chave da sessão de Teste ou se retorna um valor aleatório.

4. Por fim, o atacante executa $\mathcal{M}_2$, durante o qual é habilitado a acessar os mesmos oráculos da segunda fase. O jogo se encerra com $\mathcal{M}_2$ gerando um palpite $\hat{b}$ para b.

A definição de sessão *fresh* é a mesma da Definição 4.2 (pág. 56). Para segurança de protocolo, definimos:

Definição 4.5 (Segurança no modelo Mal-LBG) *Um protocolo de acordo de chave com autenticação sem certificado é dito seguro no modelo Mal-LBG se:*

1. *é seguro no modelo LBG*

2. *e todo adversário no modelo Mal-LBG tem vantagem negligenciável em vencer o jogo Jogo-AKA-Mal.*

4.6.4 Mal-SJ$^+$: Extensão contra Autoridade Mal Intencionada sobre SJ$^+$

As modificações citadas na seção anterior são aplicáveis ao modelo SJ$^+$, somando-se as observações feitas na Seção 4.6.1. Os oráculos EstablishParty e PPK-EstablishParty são desabilitados para o adversário Mal-SJ$^+$, da mesma forma que o são RevealMasterKey e RevealPartial.

O oráculo StrongPPK-RevealSessionKey é substituído por um equivalente PPK-RevealSessionKey, em que o adversário é obrigado a informar um valor para chave parcial secreta para o dono da sessão consultada. Esse oráculo responde um valor de chave de sessão com base na chave parcial informada, usando o par original de chave pública/valor secreto do dono da sessão, porém usa a chave pública substituída do parceiro, quando for o caso. Os oráculos RevealSessionKey e StrongPPK-RevealSessionKey são desabilitados.

Para a definição de sessão *fresh*, repete-se a mesma usada em Mal-LBG; similarmente define-se protocolo seguro como abaixo:

Definição 4.6 (Segurança no modelo Mal-SJ$^+$) *Um protocolo de acordo de chave com autenticação sem certificado é dito seguro no modelo Mal-SJ$^+$ se:*

1. *é seguro no modelo SJ$^+$*
2. *e todo adversário no modelo Mal-SJ$^+$ tem vantagem negligenciável em vencer o jogo Jogo-AKA-Mal.*

4.7 Síntese e Conclusões Parciais

Neste capítulo, descrevemos acordo de chave secreta com autenticação, apresentamos modelos de segurança para o caso sem certificado e realizamos extensões sobre eles com o objetivo de aumentar a segurança ou melhor formalizar definições anteriores.

No próximo capítulo, propomos novos protocolos que serão mostrados seguros, mais adiante no Capítulo 6, nos modelos de segurança aqui discutidos.

Capítulo 5

Protocolos de Acordo de Chave com Autenticação sem Certificado

O objetivo deste capítulo é apresentar protocolos de acordo de chave para dois participantes mutuamente autenticados sem certificado, que são demonstrados seguros sob diferentes modelos de segurança e problemas computacionais. Aqui descrevemos os protocolos e expomos uma análise comparativa; posteriormente no Capítulo 6, apresentamos as provas formais de segurança. São contribuições originais desta tese os protocolos GOT1, GOT2, GNT1, GNT2, GNT3 e GNT4, que são introduzidos neste capítulo.

5.1 Considerações Iniciais

Classificamos os protocolos segundo o modelo de segurança e problema computacional sobre o qual os protocolos são mostrados seguros. Na Seção 5.5, são apresentados os protocolos que são seguros contra adversário moderado; em seguida, na Seção 5.6, os que são seguros contra adversário forte; na Seção 5.7, os seguros contra autoridade mal intencionada e, por fim, na Seção 5.8, os seguros contra adversário fraco.

Dentro de cada bloco, optamos por relacionar primeiro os protocolos que são baseados na dificuldade do problema computacional Gap-BDH (Seção 3.3, pág. 41), por serem menos complexos que os que virão na sequência, que são seguros sob o problema BDH. A Tabela 5.1 apresenta essa classificação dos protocolos.

Tabela 5.1: *Classificação dos protocolos propostos*

Protocolo	Modelo de segurança	Problema computacional	Poder do adversário
GNT2	SJ+	Gap-BDH	moderado
GNT4		BDH	
GOT1-Gap	LBG	Gap-BDH	forte
GOT1-BDH		BDH	
GNT3		BDH	
GNT2	Mal-SJ+	Gap-BDH	autoridade mal intencionada
GNT1	Mal-LBG	Gap-BDH	
GOT2	LG	Gap-BDH	fraco

Os protocolos aqui descritos são baseados no cálculo de emparelhamentos bilineares, citados na Seção 3.2 (pág. 40). Por questão de clareza, introduzimos os protocolos fazendo uso de emparelhamento simétrico; mais adiante na Seção 5.10, mostramos as adaptações necessárias para que os protocolos façam uso de emparelhamento assimétrico.

Na próxima seção, transcrevemos duas versões para as fases de inicialização do sistema e de geração de chaves de usuário, por serem compartilhadas entre vários dos protocolos aqui discutidos.

5.2 Parâmetros e Chaves na Formulação AP

A maioria dos protocolos citados neste capítulo seguem a formulação AP para chave pública sem certificado (ver Seção 3.4.1, pág. 46), em que a chave pública pode ser usada antes da chave secreta completa estar definida Al-Riyami e Paterson (2003). Identificamos duas versões de parâmetros e chaves que aqui chamamos por básicos e estendidos. Passamos a descrever ambas versões.

5.2.1 Parâmetros Básicos

Os parâmetros básicos na formulação AP de chave pública sem certificado são usados principalmente na construção de protocolos CL-AKA que têm segurança fundamentada no problema computacional Gap-BDH.

Inicialização do Sistema

Para um dado parâmetro de segurança k, são selecionados os grupos aditivo G e multiplicativo G_T, ambos de ordem prima q. Seja $P \in G$ um gerador de G e seja $e : G \times G \to G_T$ um emparelhamento bilinear admissível.

A autoridade do sistema, KGC, seleciona $s \xleftarrow{\$} \mathbb{Z}_q^*$ para ser a chave mestra secreta e calcula sua chave mestra pública como sendo sP. Também seleciona duas funções criptográficas de hash: $H : \{0,1\}^* \to \{0,1\}^k$ e $H_1 : \{0,1\}^* \to G$.

Por fim, a autoridade publica os parâmetros públicos $\texttt{params} = \langle q, G, G_T, e, k, P, sP, H, H_1 \rangle$.

Na Tabela 5.2, sumarizamos esses procedimentos.

Geração de Chaves de Usuário

Cada usuário U dentro do sistema:

- possui um identificador único $ID_U \in \{0,1\}^*$ e o valor público $Q_U = H_1(ID_U) \in G$
- seleciona um valor secreto $x_U \xleftarrow{\$} \mathbb{Z}_q^*$
- calcula sua chave pública $x_U P \in G$
- recebe sua chave secreta parcial, $d_U = sQ_U \in G$, calculada e entregue pela autoridade do sistema de forma segura.

Esses parâmetros e chaves de usuário são sumarizados na Tabela 5.3.

Tabela 5.2: *Inicialização do sistema*

Parâmetro de segurança	k
Grupos algébricos	G, de ordem prima q e gerador P G_T, de ordem prima q
Emparelhamento bilinear	$e : G \times G \to G_T$
Par de chaves mestra	$s \xleftarrow{\$} \mathbb{Z}_q^*$ (msk, chave mestra secreta) $sP \in G$ (mpk, chave mestra pública)
Funções de hash	$H : \{0,1\}^* \to \{0,1\}^k$ $H_1 : \{0,1\}^* \to G$
Parâmetros públicos	$\langle q, G, G_T, e, k, P, sP, H, H_1 \rangle$

Tabela 5.3: *Chaves de usuário*

Valores públicos	$ID_U \in \{0,1\}^*$ $Q_U = H_1(ID_U)$	identidade do usuário
	$x_U P \in G$	chave pública, gerada pelo usuário
Valores secretos	$d_U = sQ_U \in G$	chave secreta parcial, gerada por KGC
	$x_U \xleftarrow{\$} \mathbb{Z}_q^*$	valor secreto, escolhido pelo usuário

5.2.2 Parâmetros Estendidos

Os parâmetros que qualificamos como estendidos são usados essencialmente pelos protocolos CL-AKA cuja segurança é baseada no problema computacional BDH. A principal diferença entre os parâmetros básicos e estendidos é a adição da função de hash H_2 e as consequentes alterações que se fazem necessárias por conta desse acréscimo. As etapas de inicialização e de geração de chaves de usuário são descritas a seguir e sumarizadas respectivamente nas tabelas 5.4 e 5.5.

Inicialização do Sistema

Para um dado parâmetro de segurança k, são selecionados os grupos aditivo G e multiplicativo G_T, ambos de ordem prima q. Seja $P \in G$ um gerador de G e seja $e : G \times G \to G_T$ um emparelhamento bilinear admissível.

A autoridade do sistema seleciona $s \xleftarrow{\$} \mathbb{Z}_q^*$ para ser a chave mestra secreta e calcula sua chave mestra pública como sendo sP. Também seleciona três funções criptográficas de hash: $H : \{0,1\}^* \to \{0,1\}^k$, $H_1 : \{0,1\}^* \to G$ e $H_2 : G \to G$.

Por fim, o KGC publica os parâmetros públicos $\texttt{params} = \langle q, G, G_T, e, k, P, sP, H, H_1, H_2 \rangle$.

Tabela 5.4: *Inicialização do sistema com parâmetros estendidos*

Parâmetro de segurança	k
Grupos algébricos	G, de ordem prima q e gerador P G_T, de ordem prima q
Emparelhamento bilinear	$e : G \times G \to G_T$
Par de chaves mestra	$s \xleftarrow{\$} \mathbb{Z}_q^*$ (msk, chave mestra secreta) $sP \in G$ (mpk, chave mestra pública)
Funções de hash	$H : \{0,1\}^* \to \{0,1\}^k$ $H_1 : \{0,1\}^* \to G$ $H_2 : \{0,1\}^* \to G$
Parâmetros públicos	$\langle q, G, G_T, e, k, P, sP, H, H_1, H_2 \rangle$

Tabela 5.5: *Chaves de usuário com parâmetros estendidos*

Valores públicos	$ID_U \in \{0,1\}^*$ $Q_{U_1} = H_1(ID_U)$ $Q_{U_2} = H_2(ID_U)$	identidade do usuário
	$x_U P \in G$	chave pública, gerada pelo usuário
Valores secretos	$d_U = (d_{U_1}, d_{U_2}) \in G^2$ $d_{U_1} = sQ_{U_1}$ $d_{U_2} = sQ_{U_2}$	chave secreta parcial, gerada por KGC
	$x_U \xleftarrow{\$} \mathbb{Z}_q^*$	valor secreto, escolhido pelo usuário

Geração de Chaves de Usuário

Cada usuário U dentro do sistema:

- possui um identificador único $ID_U \in \{0,1\}^*$ e valores públicos $(Q_{U_1}, Q_{U_2}) \in G^2$, onde $Q_{U_1} = H_1(ID_U)$ e $Q_{U_2} = H_2(ID_U)$
- seleciona um valor secreto $x_U \xleftarrow{\$} \mathbb{Z}_q^*$
- calcula sua chave pública $x_U P \in G$
- recebe sua chave secreta parcial, calculada e entregue pela autoridade do sistema de forma segura, $d_U = (d_{U_1}, d_{U_2}) \in G^2$, onde $d_{U_1} = sQ_{U_1}$ e $d_{U_2} = sQ_{U_2}$.

5.3 Sessão de Acordo de Chave e Troca de Mensagens

Considere que dois usuários, digamos A e B, queiram estabelecer uma chave secreta em comum. Suponha que o participante A inicia a comunicação ao executar o procedimento *Inicia*. Em atendimento ao chamado de A, o participante B executa o procedimento *Responde* para então calcular a chave secreta compartilhada, conforme ilustrado na Figura 4.1 (pág. 52). Para a maioria dos protocolos descritos adiante, as instruções de *Inicia* e *Responde* se repetem. Por esse motivo vamos descrevê-las na sequência e sintetizá-las na Figura 5.1.

O participante A inicia uma sessão, sorteia um valor secreto temporário $r_A \xleftarrow{\$} \mathbb{Z}_q^*$, calcula $r_A P$, nomeia o par $(r_A P, x_A P)$ como E_A e o envia para B. De modo análogo, B escolhe $r_B \xleftarrow{\$} \mathbb{Z}_q^*$, calcula $r_B P$ e envia $E_B = (r_B P, x_B P)$ para A.

Observamos que a inclusão do valor da chave pública na mensagem trocada entre os participantes de uma sessão não é obrigatória. Isto é, o valor $x_A P$ pode ser excluído de E_A se na implementação do protocolo for garantida a entrega da chave pública por outro meio (um diretório de chaves públicas, por exemplo). Na prática, a remoção dos valores de chave pública trafegando entre os usuários a cada sessão otimiza o uso do canal de comunicação. No entanto, acreditamos que, dessa forma, o adversário tem condições reais mais plausíveis de substituir chaves públicas, situação essa que impediria dois usuários de chegarem ao final de uma sessão com o mesmo valor de chave secreta. Um transtorno como tal seria comparável ao ataque DoD (ver pág. 24) sobre esquemas de cifragem sem certificado, pois o dano causado pelo adversário seria o de impedir que usuários legítimos consigam estabelecer chaves de sessão. Aparentemente, é mais fácil para um adversário alterar um diretório público que modificar ativamente o tráfego entre dois usuários. Por essa razão, incluímos os valores de chave pública na descrição da troca de mensagens entre os participantes da sessão.

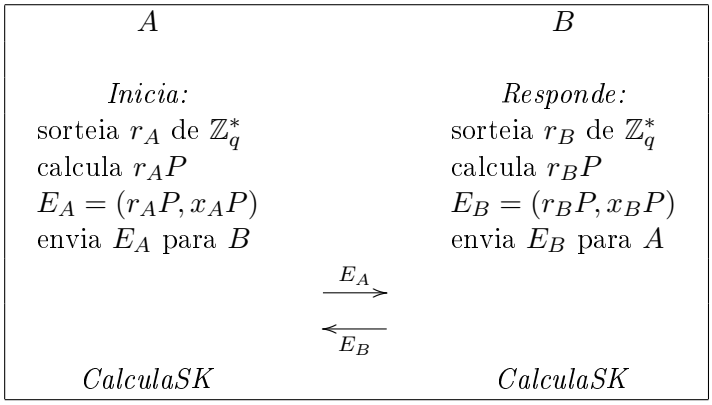


Figura 5.1: *Início de sessão de acordo de chave e troca de mensagens*

5.4 Construção Segura

Uma das contribuições de Lippold *et al.* (2009) foi mostrar que a composição natural de dois protocolos de acordo de chaves, um no modelo baseado em identidade (ID-AKA) e outro no modelo convencional de chave pública (PK-AKA), não leva a um CL-AKA seguro (sob os modelos desses autores e de Swanson (2008)). Por esse motivo, todos seus predecessores falharam na construção de um acordo sem certificado.

Para ser possível uma composição segura entre um ID-AKA e um PK-AKA, há que se considerar

o caso em que um adversário corrompe o segredo temporário de um dos participantes (digamos, r_A) e, além disso, corrompe também um dos dois componentes da chave secreta permanente: a chave parcial secreta ou o valor secreto (d_A ou x_A). Por esse motivo, é necessário combinar ambos os valores d_A e x_A (em uma variável que precisa ser dado de entrada na função de hash para cálculo da chave de sessão).

Essa técnica de construção é empregada nos protocolos descritos nas próximas seções.

5.5 Protocolos Seguros contra Adversário Moderado

Os protocolos apresentados nesta seção são mostrados seguros contra adversário de poder moderado, isto é, que tem acesso a um oráculo fraco de revelação de chave de sessão, conforme descrito na Seção 4.6.1 (pág. 59).

Salientamos que, até onde sabemos, os dois protocolos expostos nesta seção são os únicos mostrados seguros no modelo de adversário moderado. Acreditamos que o modelo de adversário moderado é mais útil para aplicações reais, pois, embora seja mais fraco que o modelo forte, confere ao adversário poder mais realista.

5.5.1 GNT2 – Seguro no Modelo SJ$^+$ sob Gap-BDH

O protocolo GNT2 foi projetado para ter segurança demonstrável no modelo de Swanson e Jao (2009) melhorado, sob a hipótese de dificuldade do problema Gap-BDH e oráculos aleatórios.

As fases de inicialização do sistema e de geração de chaves de usuário deste protocolo são relacionadas com os parâmetros básicos, apresentados respectivamente nas tabelas 5.2 e 5.3 (pág. 65). O início das sessões e a fase de troca de mensagens são os mesmos encontrados na Figura 5.1 (pág. 67).

Para realizar o cálculo da chave secreta compartilhada, o participante A, assim que receber E_B de B, verifica a pertinência a G^2 e calcula

$K = e(r_B P + Q_B, r_A s P + d_A)$

$M = e(x_B P + Q_B, x_A s P + d_A)$

$N = e(Q_B, d_A)$

Então A calcula $Z = (x_A(x_B P), x_A(r_B P), r_A(r_B P), r_A(x_B P))$ e a chave secreta como sendo $SK = H(A, B, E_A, E_B, Z, K, M, N)$.

O participante B, de forma análoga, ao receber E_A de A, verifica a pertinência a G^2 e realiza cálculos similares, substituindo os valores de A pelos de B e vice-versa.

O resumo do protocolo e o cálculo de chave por A e B encontram-se na Figura 5.2.

Justificativas para a Construção e Propriedades de Segurança

Neste protocolo, combinamos os valores temporários de sessão com as chaves secretas parciais de forma análoga à encontrada no protocolo originalmente proposto por Shim (2003), que foi ligeiramente modificado por Chen *et al.* (2007) para torná-lo um protocolo seguro de acordo de chave baseado em identidade. Tal procedimento é observável na variável K do cálculo da chave secreta compartilhada.

Para atender à regra de construção segura de CL-AKA apontada por Lippold *et al.* (2009) e citada na Seção 5.4, combinamos os valores d_A e x_A, na variável M. A variável N no cálculo da chave

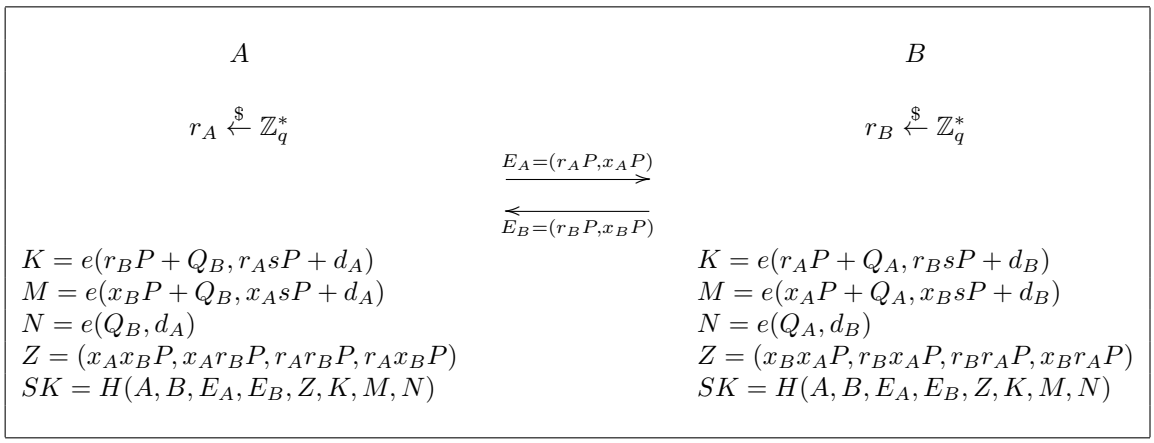


Figura 5.2: *Protocolo GNT2, seguro no modelo SJ⁺ sob Gap-BDH*

de sessão embute o acordo de chave não interativo de Sakai e Kasahara (2003); ela foi incluída para tornar possível a demonstração de segurança do protocolo (mais especificamente, para viabilizar o tratamento do caso 9, em que o adversário substitui as chaves públicas de ambos participantes da sessão de Teste e corrompe os segredos temporários de sessão dos dois envolvidos nessa sessão, conforme veremos na Seção 6.4).

A quádrupla Z é inserida para se garantir resistência a ataques KCI, discutidos na Seção 4.3 (pág. 51), e a mistura dos temporários secretos r_Ar_BP é importante para prover *Weak Perfect Forward Secrecy* e segurança no futuro perante o KGC.

A propriedade de segurança de chave conhecida é alcançada por causa da natureza aleatória dos valores temporários sorteados a cada sessão. Resistência a ataques de personificação básicos é obtida com a inclusão de K, M e N na entrada da função de hash aplicada ao final do cálculo da chave secreta compartilhada. Uma vez que essas três variáveis guardam os três componentes secretos de cada participante na sessão, seriam necessários o corrompimento dos três segredos ou da função de hash para um adversário ser bem sucedido em ataques de personificação básicos. Ataques do tipo UKS são tratados com a inclusão das identidades de ambos participantes na entrada da mesma função de hash.

As variáveis M e N contribuem para a resistência ao vazamento de segredos temporários de sessão, pois envolvem segredos independentes dos valores aleatórios sorteados a cada sessão. Por fim, a combinação x_Ax_BP e a variável M auxiliam na resistência ao vazamento de segredos temporários para o KGC.

Essas técnicas de projeto de protocolos de acordo de chave com autenticação são em grande parte sintetizadas por Boyd e Mathuria (2003).

Este protocolo atende a extensão de segurança contra autoridade mal intencionada, conforme comentaremos na Seção 5.7.1.

5.5.2 GNT4 – Seguro no Modelo SJ⁺ sob BDH

O protocolo GNT4 foi concebido para ser seguro no modelo de Swanson e Jao (2009) melhorado, sob a hipótese de dificuldade do problema BDH e oráculos aleatórios.

As fases de inicialização do sistema e de geração de chaves de usuário são relacionadas com os parâmetros estendidos, apresentados respectivamente nas tabelas 5.4 e 5.5. O início das sessões e a

fase de troca de mensagens são os mesmos encontrados na Figura 5.1.

Para realizar o cálculo da chave secreta compartilhada, o participante A, assim que receber E_B de B, verifica a pertinência a G^2 e calcula

$K = e(r_B P + Q_{B_1}, r_A s P + d_{A_1})$
$L = e(r_B P + Q_{B_2}, r_A s P + d_{A_2})$
$M = e(x_B P + Q_{B_1}, x_A s P + d_{A_1})$
$N = e(x_B P + Q_{B_2}, x_A s P + d_{A_2})$

Então A calcula $Z = (x_A(x_B P), x_A(r_B P), r_A(r_B P), r_A(x_B P))$ e a chave secreta como sendo $SK = H(A, B, E_A, E_B, Z, K, L, M, N)$.

O participante B, de forma análoga, ao receber E_A de A, verifica a pertinência a G^2 e realiza cálculos similares, substituindo os valores de A pelos de B e vice-versa.

O resumo do protocolo e o cálculo de chave por A e B encontram-se na Figura 5.3.

A		B
$r_A \xleftarrow{\$} \mathbb{Z}_q^*$		$r_B \xleftarrow{\$} \mathbb{Z}_q^*$
	$\xrightarrow{E_A=(r_A P, x_A P)}$	
	$\xleftarrow{E_B=(r_B P, x_B P)}$	
$K = e(r_B P + Q_{B_1}, r_A s P + d_{A_1})$		$K = e(r_A P + Q_{A_1}, r_B s P + d_{B_1})$
$L = e(r_B P + Q_{B_2}, r_A s P + d_{A_2})$		$L = e(r_A P + Q_{A_2}, r_B s P + d_{B_2})$
$M = e(x_B P + Q_{B_1}, x_A s P + d_{A_1})$		$M = e(x_A P + Q_{A_1}, x_B s P + d_{B_1})$
$N = e(x_B P + Q_{B_2}, x_A s P + d_{A_2})$		$N = e(x_A P + Q_{A_2}, x_B s P + d_{B_2})$
$Z = (x_A x_B P, x_A r_B P, r_A r_B P, r_A x_B P)$		$Z = (x_B x_A P, r_B x_A P, r_B r_A P, x_B r_A P)$
$SK = H(A, B, E_A, E_B, Z, K, L, M, N)$		$SK = H(A, B, E_A, E_B, Z, K, L, M, N)$

Figura 5.3: *Protocolo GNT4, seguro no modelo SJ^+ sob BDH*

Comentários

No protocolo GNT4, combinamos os valores temporários de sessão com as chaves secretas parciais de forma análoga à encontrada no protocolo de Huang e Cao (2009), que por sua vez fundamentaram seu acordo de chave baseado em identidade na solução apresentada por Chen *et al.* (2007).

As variáveis K e L são duais, para aplicação da técnica do teste do *trapdoor* proposta por Cash *et al.* (2009). Da mesma forma, M e N também são duais, o que permite que o protocolo seja mostrado seguro sob o problema BDH. Demais técnicas usadas para construção deste protocolo são as mesmas discutidas na Seção 5.5.1 (pág. 68).

5.6 Protocolos Seguros contra Adversário Forte

Lippold, Boyd e González Nieto propuseram um protocolo que foi o primeiro mostrado seguro contra adversário forte (Lippold *et al.*, 2009). O protocolo possui duas versões e uma variação que são revisadas a seguir, pois nossas propostas que são seguras contra adversário forte são fortemente relacionadas com elas.

5.6.1 Protocolos de Lippold *et al.* (2009)

Lippold *et al.* (2009) citam uma variante mais eficiente de seu protocolo, sem no entanto explicitá-la no artigo original. Os autores mencionam que a variante é segura sob a hipótese de dificuldade do problema Gap-BDH, mas não apresentam a prova formal. Afirmamos que a variante citada é de fato segura e apresentamos a prova formal no Capítulo 6.

LBG2-Gap – Gap-BDH

As fases de inicialização do sistema e de geração de chaves de usuário são relacionadas com os parâmetros básicos, apresentados respectivamente nas tabelas 5.2 e 5.3 (pág. 65). O início das sessões e a fase de troca de mensagens são os mesmos encontrados na Figura 5.1 (pág. 67). A etapa de cálculo da chave de sessão e resumo do protocolo encontram-se na Figura 5.4.

A		B
$r_A \xleftarrow{\$} \mathbb{Z}_q^*$		$r_B \xleftarrow{\$} \mathbb{Z}_q^*$
	$\xrightarrow{E_A=(r_AP,x_AP)}$	
	$\xleftarrow{E_B=(r_BP,x_BP)}$	
$K = e(Q_B, sP)^{r_A} \cdot e(d_A, r_BP)$		$K = e(Q_A, sP)^{r_B} \cdot e(d_B, r_AP)$
$L = e(Q_B, sP)^{x_A} \cdot e(d_A, x_BP)$		$L = e(Q_A, sP)^{x_B} \cdot e(d_B, x_AP)$
$N = e(Q_B, d_A)$		$N = e(Q_A, d_B)$
$Z = (x_Ax_BP, x_Ar_BP, r_Ar_BP, r_Ax_BP)$		$Z = (x_Bx_AP, r_Bx_AP, r_Br_AP, x_Br_AP)$
$SK = H(A, B, E_A, E_B, Z, K, L, N)$		$SK = H(A, B, E_A, E_B, Z, K, L, N)$

Figura 5.4: *Protocolo LBG2-Gap, seguro no modelo LBG sob Gap-BDH*

LBG1 e LBG2 – BDH

Na Tabela 5.6, denotamos o protocolo de Lippold *et al.* (2009) por LBG1 e LBG2, que são duas versões possíveis para implementação; uma se refere à descrição original do protocolo, enquanto a outra substitui operações de exponenciação em G_T por multiplicações escalares em G. A diferença de desempenho entre LBG1 e LBG2 varia em função das escolhas dos parâmetros e da implementação de emparelhamento bilinear; em geral, LBG2 confere maior velocidade. As duas variantes usam os parâmetros estendidos, apresentados respectivamente nas tabelas 5.4 e 5.5 (pág. 66). O início das sessões e a fase de troca de mensagens são os mesmos encontrados na Tabela 5.1 (pág. 67).

Tabela 5.6: *Cálculo de chave de Lippold et al. (2009), em duas versões seguras sob BDH*

LBG1	LBG2
$K_1 = e(Q_{B_1}, r_AsP) \cdot e(d_{A_1}, r_BP)$	$K_1 = e(Q_{B_1}, sP)^{r_A} \cdot e(d_{A_1}, r_BP)$
$K_2 = e(Q_{B_2}, r_AsP) \cdot e(d_{A_2}, r_BP)$	$K_2 = e(Q_{B_2}, sP)^{r_A} \cdot e(d_{A_2}, r_BP)$
$L_1 = e(Q_{B_1}, x_AsP) \cdot e(d_{A_1}, x_BP)$	$L_1 = e(Q_{B_1}, sP)^{x_A} \cdot e(d_{A_1}, x_BP)$
$L_2 = e(Q_{B_2}, x_AsP) \cdot e(d_{A_2}, x_BP)$	$L_2 = e(Q_{B_2}, sP)^{x_A} \cdot e(d_{A_2}, x_BP)$
$N_1 = e(Q_{B_1}, d_{A_1})$	$N_1 = e(Q_{B_1}, d_{A_1})$
$N_2 = e(Q_{B_2}, d_{A_2})$	$N_2 = e(Q_{B_2}, d_{A_2})$
$Z = (x_Ax_BP, x_Ar_BP, r_Ar_BP, r_Ax_BP)$	$Z = (x_Ax_BP, x_Ar_BP, r_Ar_BP, r_Ax_BP)$
$SK = H(A, B, E_A, E_B, Z, K_1, K_2, L_1, L_2, N_1, N_2)$	$SK = H(A, B, E_A, E_B, Z, K_1, K_2, L_1, L_2, N_1, N_2)$

5.6.2 GOT1-Gap – Seguro no Modelo LBG sob Gap-BDH

Este protocolo é uma otimização de LBG1, para ter segurança sob o problema Gap-BDH. As fases de inicialização do sistema e de geração de chaves de usuário são relacionadas com os parâmetros básicos, apresentados respectivamente nas tabelas 5.2 e 5.3 (pág. 65). O início das sessões e a fase de troca de mensagens são os mesmos encontrados na Tabela 5.1 (pág. 67).

Para realizar o cálculo da chave secreta compartilhada, o participante A, assim que receber E_B de B, verifica a pertinência a G^2 e calcula

$K = e(r_B P + Q_B, r_A s P + d_A)$
$L = e(x_B P, d_A) \cdot e(Q_B, x_A s P)$
$N = e(Q_B, d_A)$
$Z = (x_A x_B P, x_A r_B P, r_A r_B P, r_A x_B P)$
$SK = H(A, B, E_A, E_B, Z, K, L, N)$

Então A calcula $Z = (x_A(x_B P), x_A(r_B P), r_A(r_B P), r_A(x_B P))$ e a chave secreta como sendo $SK = H(A, B, E_A, E_B, Z, K, M, N)$.

O participante B, de forma análoga, ao receber E_A de A, verifica a pertinência a G^2 e realiza cálculos similares, substituindo os valores de A pelos de B e vice-versa. O resumo do protocolo e o cálculo de chave por A e B encontram-se na Figura 5.5.

A		B
$r_A \xleftarrow{\$} \mathbb{Z}_q^*$		$r_B \xleftarrow{\$} \mathbb{Z}_q^*$
	$\xrightarrow{E_A=(r_A P, x_A P)}$	
	$\xleftarrow{E_B=(r_B P, x_B P)}$	
$K = e(r_B P + Q_B, r_A s P + d_A)$		$K = e(r_A P + Q_A, r_B s P + d_B)$
$L = e(x_B P, d_A) \cdot e(Q_B, x_A s P)$		$L = e(x_A P, d_B) \cdot e(Q_A, x_B s P)$
$N = e(Q_B, d_A)$		$N = e(Q_A, d_B)$
$Z = (x_A x_B P, x_A r_B P, r_A r_B P, r_A x_B P)$		$Z = (x_B x_A P, r_B x_A P, r_B r_A P, x_B r_A P)$
$SK = H(A, B, E_A, E_B, Z, K, L, N)$		$SK = H(A, B, E_A, E_B, Z, K, L, N)$

Figura 5.5: *Protocolo GOT1-Gap, seguro no modelo LBG sob Gap-BDH*

Comentários

Este protocolo é uma combinação de GNT2 e LBG1. As técnicas de construção usadas são as mesmas discutidas na Seção 5.5.1 (pág. 68). Ele foi projetado para ser dual do GOT1-BDH (na seção a seguir), mas essa versão dual foi superada por outro protocolo apresentado na Seção 5.6.4.

5.6.3 GOT1-BDH – Seguro no Modelo LBG sob BDH

A construção deste protocolo visa a otimização do LBG1 para segurança sob o problema BDH. Este protocolo é dual de GOT1-Gap. As fases de inicialização do sistema e de geração de chaves de usuário são relacionadas com os parâmetros estendidos, apresentados respectivamente nas tabelas 5.4 e 5.5 (pág. 66). O início das sessões e a fase de troca de mensagens são os mesmos encontrados na Tabela 5.1 (pág. 67).

Para realizar o cálculo da chave secreta compartilhada, o participante A, assim que receber E_B de B, verifica a pertinência a G^2 e calcula

$K_1 = e(r_B P + Q_{B_1}, r_A s P + d_{A_1})$
$K_2 = e(r_B P + Q_{B_2}, r_A s P + d_{A_2})$
$L_1 = e(Q_{B_1}, x_A s P) \cdot e(d_{A_1}, x_B P)$
$L_2 = e(Q_{B_2}, x_A s P) \cdot e(d_{A_2}, x_B P)$
$N_1 = e(Q_{B_1}, d_{A_1})$
$N_2 = e(Q_{B_2}, d_{A_2})$

Então A calcula $Z = (x_A(x_B P), x_A(r_B P), r_A(r_B P), r_A(x_B P))$ e a chave secreta como sendo $SK = H(A, B, E_A, E_B, Z, K, L, M, N)$.

O participante B, de forma análoga, ao receber E_A de A, verifica a pertinência a G^2 e realiza cálculos similares, substituindo os valores de A pelos de B e vice-versa. O resumo do protocolo e o cálculo de chave por A e B encontram-se na Figura 5.6.

A		B
$r_A \xleftarrow{\$} \mathbb{Z}_q^*$		$r_B \xleftarrow{\$} \mathbb{Z}_q^*$
	$\xrightarrow{E_A=(r_A P, x_A P)}$	
	$\xleftarrow{E_B=(r_B P, x_B P)}$	
$K_1 = e(r_B P + Q_{B_1}, r_A s P + d_{A_1})$		$K_1 = e(r_B P + Q_{B_1}, r_A s P + d_{A_1})$
$K_2 = e(r_B P + Q_{B_2}, r_A s P + d_{A_2})$		$K_2 = e(r_B P + Q_{B_2}, r_A s P + d_{A_2})$
$L_1 = e(Q_{B_1}, x_A s P) \cdot e(d_{A_1}, x_B P)$		$L_1 = e(Q_{B_1}, x_A s P) \cdot e(d_{A_1}, x_B P)$
$L_2 = e(Q_{B_2}, x_A s P) \cdot e(d_{A_2}, x_B P)$		$L_2 = e(Q_{B_2}, x_A s P) \cdot e(d_{A_2}, x_B P)$
$N_1 = e(Q_{B_1}, d_{A_1})$		$N_1 = e(Q_{B_1}, d_{A_1})$
$N_2 = e(Q_{B_2}, d_{A_2})$		$N_2 = e(Q_{B_2}, d_{A_2})$
$Z = (x_A x_B P, x_A r_B P, r_A r_B P, r_A x_B P)$		$Z = (x_B x_A P, r_B x_A P, r_B r_A P, x_B r_A P)$
$SK = H(A, B, E_A, E_B, Z, K_1, K_2, L_1, L_2, N_1, N_2)$		$SK = H(A, B, E_A, E_B, Z, K_1, K_2, L_1, L_2, N_1, N_2)$

Figura 5.6: *Protocolo GOT1-BDH, seguro no modelo LBG sob BDH*

Comentários

As variáveis K_1 e K_2, L_1 e L_2, e N_1 e N_2 são duais entre si para que o protocolo seja mostrado seguro sob o problema BDH, usando a técnica do teste do *trapdoor*. Na ocasião em que este protocolo foi elaborado, não acreditávamos que seria possível otimizá-lo ainda mais em relação ao LGB1. Posteriormente conseguimos eliminar a dependência das variáveis N_1 e N_2 nas provas de segurança, o que deu origem à versão nomeada GNT3, apresentada na seção a seguir.

5.6.4 GNT3 – Seguro no Modelo LBG sob BDH

Este protocolo é uma versão mais eficiente de GOT1-BDH, exibido na seção anterior. As fases de inicialização do sistema e de geração de chaves de usuário são relacionadas com os parâmetros estendidos, apresentados respectivamente nas tabelas 5.4 e 5.5 (pág. 66), pois o protocolo foi elaborado para ter segurança sob o problema BDH. O início das sessões e a fase de troca de mensagens são os mesmos encontrados na Tabela 5.1 (pág. 67).

Para realizar o cálculo da chave secreta compartilhada, o participante A, assim que receber E_B de B, verifica a pertinência a G^2 e calcula

$K = e(r_B P + Q_{B_1}, r_A s P + d_{A_1})$
$L = e(r_B P + Q_{B_2}, r_A s P + d_{A_2})$
$M = e(x_B P, d_{A_1}) \cdot e(Q_{B_1}, x_A s P)$
$N = e(x_B P, d_{A_2}) \cdot e(Q_{B_2}, x_A s P)$

Então A calcula $Z = (x_A(x_B P), x_A(r_B P), r_A(r_B P), r_A(x_B P))$ e a chave secreta como sendo $SK = H(A, B, E_A, E_B, Z, K, L, M, N)$.

O participante B, de forma análoga, ao receber E_A de A, verifica a pertinência a G^2 e realiza cálculos similares, substituindo os valores de A pelos de B e vice-versa. O resumo do protocolo e o cálculo de chave por A e B encontram-se na Figura 5.7.

A		B
$r_A \xleftarrow{\$} \mathbb{Z}_q^*$		$r_B \xleftarrow{\$} \mathbb{Z}_q^*$
	$\xrightarrow{E_A = (r_A P, x_A P)}$	
	$\xleftarrow{E_B = (r_B P, x_B P)}$	
$K = e(r_B P + Q_{B_1}, r_A s P + d_{A_1})$		$K = e(r_A P + Q_{A_1}, r_B s P + d_{B_1})$
$L = e(r_B P + Q_{B_2}, r_A s P + d_{A_2})$		$L = e(r_A P + Q_{A_2}, r_B s P + d_{B_2})$
$M = e(x_B P, d_{A_1}) \cdot e(Q_{B_1}, x_A s P)$		$M = e(x_A P, d_{B_1}) \cdot e(Q_{A_1}, x_B s P)$
$N = e(x_B P, d_{A_2}) \cdot e(Q_{B_2}, x_A s P)$		$N = e(x_A P, d_{B_2}) \cdot e(Q_{A_2}, x_B s P)$
$Z = (x_A x_B P, x_A r_B P, r_A r_B P, r_A x_B P)$		$Z = (x_B x_A P, r_B x_A P, r_B r_A P, x_B r_A P)$
$SK = H(A, B, E_A, E_B, Z, K, L, M, N)$		$SK = H(A, B, E_A, E_B, Z, K, L, M, N)$

Figura 5.7: *Protocolo GNT3, seguro no modelo LBG sob BDH*

Comentários

No protocolo GNT3, usamos a variável K para combinar os valores temporários de sessão com as chaves baseadas na identidade, da mesma forma que a encontrada no protocolo de Huang e Cao (2009). A variável M, que combina as partes relacionadas à chave pública com as baseadas em identidade, difere de K na estrutura e é computacionalmente mais cara; no entanto permite que, na prova de segurança, o adversário tenha acesso ao oráculo forte que revela chaves de sessão sobre sessões que envolvem um participante cuja chave pública fora substituída, mesmo que não se conheça o valor secreto correspondente.

As variáveis K e L são duais, para aplicação da técnica do teste do *trapdoor* proposta por Cash *et al.* (2009). Da mesma forma, M e N também são duais, o que permite que o protocolo seja mostrado seguro sob o problema BDH. As demais técnicas usadas para construção deste protocolo são as mesmas discutidas na Seção 5.5.1 (pág. 68).

5.6.5 YT – Seguro no Modelo LBG sob Gap-DH

Yang e Tan (2011a) apresentam um protocolo CL-AKA sem emparelhamento, com chave pública na formulação BSS (citada na Seção 3.4.2, pág. 46), em que a autoridade do sistema gera a chave

parcial secreta em função do valor da chave pública escolhida pelo usuário. O protocolo é mostrado seguro no modelo contra adversário Forte, sob Gap-DH.

No entanto, os autores adotam uma solução um tanto quanto excêntrica, para um protocolo no modelo de chave pública sem certificado. Yang e Tan atribuem ao KGC um par adicional de chaves que é usado explicitamente para assinatura e verificação. A chave pública de cada usuário é formada por uma tripla na forma (xP, C, σ). O valor xP é o principal componente da chave pública e tem a mesma função que nos protocolos anteriormente descritos. O par (C, σ) é resultado do algoritmo de assinatura que a autoridade aplica sobre a identidade do usuário concatenada com o compromisso C. A cada sessão de acordo de chave, ambos participantes trocam entre si uma quíntupla (ID, xP, C, σ, rP), em que ID se refere à identidade do dono da sessão e rP é o valor temporário de sessão. Cada participante da sessão deve verificar a assinatura σ de seu parceiro antes de proceder com o cálculo da chave secreta. Caso uma verificação de assinatura não seja bem sucedida, o acordo de chave é abortado. O esquema de assinatura empregado pode ser qualquer um, desde que seja não falsificável na noção EUF-CMA (Goldwasser *et al.*, 1988). Desse modo, um adversário que substitua uma chave pública supostamente não consegue modificar a parte (C, σ) sem resolver o problema computacional subjacente ao esquema de assinatura. Por outro lado, se o adversário substituir apenas o componente xP de uma chave pública e conseguir quebrar a segurança do protocolo, estará resolvendo o problema Gap-DH.

Em resumo, o protocolo *sem certificado* de Yang e Tan (2011a) exige que usuários transmitam, a cada sessão, uma quádrupla (ID, xP, C, σ) cuja assinatura σ precisa ser verificada com a chave pública da autoridade. Cabe ressaltar que esta chave pública de verificação precisa ser certificada. Ademais, se o esquema de assinatura (usado em composição com este CL-AKA) for provado seguro sob a hipótese de dificuldade de um problema computacional que não seja pelo menos tão difícil quanto o Gap-DH, então fica reduzida a força do protocolo inicial.

É preciso observar que, se o esquema de assinatura adotado for um convencional (que requer certificado digital para a chave pública de verificação), não é sensato considerar que o protocolo de Yang e Tan, como um todo, possa ser classificado como um de chave pública sem certificado.

5.7 Protocolos Seguros contra Autoridade Mal Intencionada

Aplicamos a extensão de segurança contra autoridade mal intencionada descrita na Seção 4.6.2 (pág. 59) em dois protocolos cuja segurança fora baseada em Gap-BDH. O primeiro protocolo, GNT2, foi concebido para ser seguro originalmente no modelo de Swanson e Jao melhorado, contra adversário de poder moderado (ver Seção 4.6.1, pág. 59). O segundo protocolo segue o modelo de segurança de Lippold *et al.* (2009) e é introduzido adiante, na Seção 5.7.2.

5.7.1 GNT2 – Seguro no Modelo Mal-SJ$^+$ sob Gap-BDH

O protocolo GNT2 descrito na Seção 5.5.1(pág. 68) também pode ser mostrado seguro na extensão de segurança contra autoridade mal intencionada que frauda na geração de parâmetros do sistema. A demonstração que acontece sob a hipótese de dificuldade do problema Gap-BDH é colocada no Capítulo 6. Portanto este protocolo é seguro contra adversário de poder moderado e resistente à geração desonesta de parâmetros do sistema.

5.7.2 GNT1 – Seguro no Modelo Mal-LBG sob Gap-BDH

As fases de inicialização do sistema e de geração de chaves de usuário são relacionadas com os parâmetros estendidos, apresentados respectivamente nas Tabelas 5.4 e 5.5 (pág. 66). O início das sessões e a fase de troca de mensagens são os mesmos encontrados na Figura 5.1 (pág. 67).

Para calcular a chave secreta compartilhada, o participante A, assim que receber E_B de B, verifica a pertinência a G^2 e calcula

$K = e(r_B P + Q_{B_1}, r_A s P + d_{A_1})$
$L = e(r_B P + Q_{B_2}, r_A s P + d_{A_2})$
$M = e(x_B P, d_{A_1}) \cdot e(Q_{B_1}, x_A s P)$
$Z = (x_A x_B P, x_A r_B P, r_A r_B P, r_A x_B P)$
$SK = H(A, B, E_A, E_B, K, L, M, Z)$

Então A calcula $Z = (x_A(x_B P), x_A(r_B P), r_A(r_B P), r_A(x_B P))$ e a chave secreta como sendo $SK = H(A, B, E_A, E_B, Z, K, M, N)$.

O participante B, de forma análoga, ao receber E_A de A, verifica a pertinência a G^2 e realiza cálculos similares, substituindo os valores de A pelos de B e vice-versa. O resumo do protocolo e o cálculo de chave por A e B encontram-se na Figura 5.8.

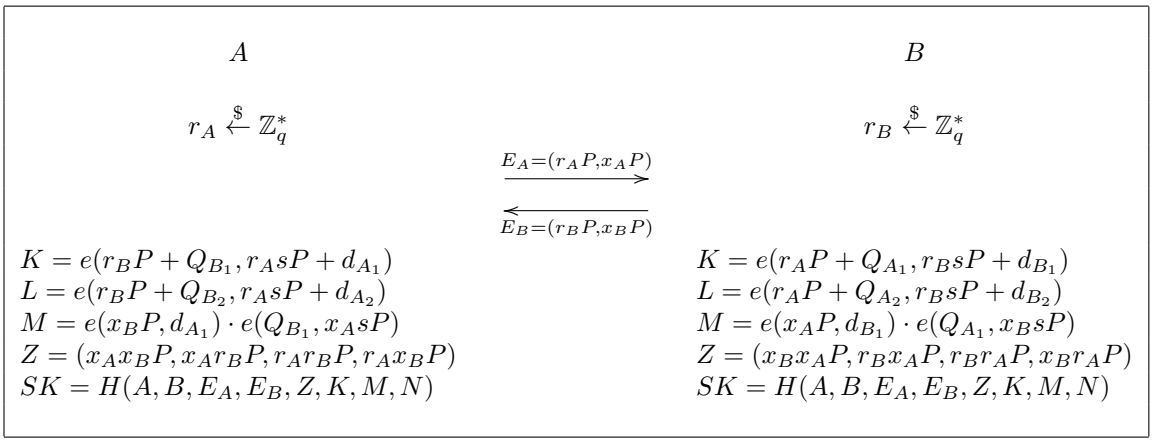


Figura 5.8: *Protocolo GNT1, seguro no modelo estendido sob Gap-BDH*

Comentários

Este protocolo é uma variante de GNT3. As técnicas de construção usadas são as mesmas discutidas na Seção 5.5.1 (pág. 68). Inicialmente estávamos tentando mostrar a segurança do protocolo GNT3 contra autoridade mal intencionada, mas não foi possível concretizar a prova sob a hipótese de dificuldade do problema BDH. Ao adotar parâmetros de sistema e formulações de chave básicos e estendidos, como os apresentados nas tabelas 5.2, 5.3, 5.4 e 5.5, não encontramos nenhum protocolo que fosse seguro contra autoridade mal intencionada sob o problema BDH.

Conjecturamos que é possível completar a prova de segurança dos protocolos GOT1-Gap e LBG2-Gap para que sejam mostrados seguros também na extensão contra autoridade mal intencionada, assumindo-se a dificuldade do problema Gap-BDH.

5.8 Protocolos Seguros contra Adversário Fraco

Os protocolos apresentados nesta seção são seguros contra adversário de poder fraco, isto é, que tem restrições para corromper os usuários no sistema e os participantes da sessão de Teste, conforme descrito na Seção 4.4.3 (pág. 57).

5.8.1 GOT2 – Seguro no Modelo Fraco sob Gap-BDH

O protocolo apresentado a seguir é uma concretização da conversão genérica de Boyd *et al.* (2009), com adaptações necessárias para o caso sem certificado. Ele é composto das seguintes fases:

- Inicialização do Sistema
- Geração de Chaves de Usuário
- Acordo de Chave
  - Fase de encapsulamento de temporário
  - Fase de comunicação
  - Fase de desencapsulamento de temporário
  - Cálculo da chave compartilhada

Inicialização do Sistema

Sejam G e G_T grupos de ordem prima q com $P \in G$ um gerador de G. Seja $e : G \times G \to G_T$ um emparelhamento bilinear admissível.

O KGC escolhe $s \xleftarrow{\$} \mathbb{Z}_q^*$ como sua chave secreta mestra e calcula sua chave mestra pública como sendo $mpk = sP$.

Para um parâmetro de segurança k, o KGC seleciona três funções de hash criptográficas:
$H : \{0,1\}^* \to \{0,1\}^k$
$H_1 : \{0,1\}^* \to G$
$H_2 : G \times G_T \to \{0,1\}^k$
O KGC publica os parâmetros públicos $\texttt{params} = \langle q, G, G_T, e, k, P, sP, H, H_1, H_2 \rangle$.

Geração de Chaves de Usuário

Cada usuário U:

- possui uma identidade única $ID_U \in \{0,1\}^*$ e um valor público $Q_U \in G$, onde $Q_U = H_1(ID_U)$
- escolhe $x_U \xleftarrow{\$} \mathbb{Z}_q^*$
- calcula sua chave pública $PK_U = x_U P$
- o KGC calcula e entrega de forma segura a chave parcial secreta $d_U \in G$, onde $d_U = sQ_U$.

Acordo de Chave

Um participante A que deseja computar uma chave secreta com B executa o protocolo com as seguintes fases:

1. Fase de encapsulamento de temporário:
   $t_A \xleftarrow{\$} \mathbb{Z}_q^*$
   $C_A \leftarrow t_A P$
   $Z_{A_1} \leftarrow t_A PK_B$
   $Z_{A_2} \leftarrow e(t_A \cdot mpk, Q_B)$
   $K_A \leftarrow H_2(Z_{A_1}, Z_{A_2})$

2. Fase de comunicação:
   $y_A \xleftarrow{\$} \mathbb{Z}_q^*$
   $Y_A \leftarrow y_A P$
   $E_A \leftarrow (A, PK_A, C_A, Y_A)$
   A envia E_A para B
   recebe $E_B = (B, PK_B, C_B, Y_B)$ de B
   verifica se C_B, Y_B estão em G e se PK_B está correto; se não, A aborta.

3. Fase de desencapsulamento de temporário:
   $Z_{B_1} \leftarrow x_A C_B$
   $Z_{B_2} \leftarrow e(C_B, d_A)$
   $K_B \leftarrow H_2(Z_{B_1}, Z_{B_2})$

4. Cálculo da chave compartilhada:
   $K_{AB} \leftarrow y_A Y_B$
   $M_1 \leftarrow (x_A PK_B, x_A Y_B, y_A PK_B)$
   $M_2 \leftarrow ((t_A + y_A)(C_B + Y_B), Z_{A_1}, Z_{B_1})$
   $sid \leftarrow (E_A, E_B)$
   $pid \leftarrow (ID_A, ID_B, app_{id})$
   $SK \leftarrow H(K_A, K_B, K_{AB}, M_1, M_2, sid, pid)$

O participante B executará os mesmos passos de forma análoga, bastando substituir os valores de A pelos de B e vice-versa, e calcular M_1 como $(x_B PK_A, y_B PK_A, x_B Y_A)$.

Comentários

Este protocolo segue uma abordagem totalmente diferente das anteriores. Usamos a conversão genérica de um mecanismo KEM para um protocolo de acordo de chaves com autenticação, apresentado por Boyd, Cliff, González Nieto e Paterson (Boyd *et al.*, 2009). Os autores demonstraram que, dado um esquema KEM baseado em identidade CCA-seguro, então a conversão produz um protocolo de acordo de chaves seguro sob o modelo de Canetti-Krawczyk (Canetti e Krawczyk, 2001), com a propriedade adicional de resistência a ataques KCI.

Como entrada da conversão de Boyd *et al.* (2009), usamos o mecanismo KEM sem certificado (CL-KEM) apresentado por Fiore, Gennaro e Smart (Fiore *et al.*, 2010), que foi demonstrado

ser CCA-seguro e é mais eficiente que os esquemas de Bentahar, Farshim, Malone-Lee e Smart (Bentahar *et al.*, 2008).

Conjecturamos que este protocolo pode ser demonstrado seguro no modelo descrito na Seção 4.4.3, sob a hipótese de dificuldade do problema Gap-BDH.

5.8.2 LG – Seguro no Modelo Fraco sob DBDH

Lippold e González Nieto (2010) também usaram a conversão de Boyd *et al.* (2009) com o objetivo de construir um protocolo CL-AKA que pudesse ser mostrado seguro sob o modelo padrão, sem oráculos aleatórios. Da mesma forma que nós, os autores precisaram restringir as ações do adversário para completar a prova de segurança. Na primeira versão deste trabalho (Lippold e González Nieto, 2010), o modelo de Canetti e Krawczyk (2001) serviu de base para a prova de segurança, uma vez que a transformação de Boyd *et al.* (2009) também se vale do modelo CK01.

Posteriormente, Lippold (2010) atualiza em sua tese de doutorado uma demonstração para uma forma enfraquecida de seu modelo de segurança (Lippold *et al.*, 2009), que é comparável ao modelo de adversário fraco aplicado ao protocolo GOT2 da seção anterior. O autor sugere duas possíveis implementações para sua construção no modelo padrão, uma com base em um CL-KEM de Huang e Wong (2007) e outra a partir do CL-KEM de Lippold *et al.* (2010). Com a primeira alternativa, o protocolo requer três cálculos de emparelhamento bilinear e 13 multiplicações escalares; na segunda, oito emparelhamentos e oito multiplicações (que podem ser reduzidas para três emparelhamentos e oito multiplicações, com armazenamento de valores pré-computados).

5.9 Análise

Nesta seção, realizamos análises sobre os protocolos que são mostrados seguros nos modelos descritos no Capítulo 4. Primeiramente na seção a seguir, discutimos como os protocolos se relacionam entre si, de acordo com o nível de segurança; então mostrarmos o desempenho de cada protocolo na Seção 5.9.2.

5.9.1 Nível de Segurança

As propriedades de segurança dos protocolos dependem essencialmente do modelo sob o qual são demonstrados seguros. Por exemplo, com exceção dos protocolos seguros contra adversários fracos, todos satisfazem os requisitos de segurança relacionados na Seção 4.3 (pág. 51).

A Tabela 5.7 classifica os protocolos de acordo com o poder do adversário (modelo de segurança) e problema computacional sob o qual eles são mostrados seguros. O problema Gap-BDH é inferior ao BDH; portanto quanto mais à direita da tabela um protocolo estiver, mais eficiente ele tende a ser, porém é teoricamente pior. Os modelos de segurança foram ordenados segundo o poder do adversário. As linhas superiores da tabela reúnem os adversários mais fortes, enquanto as inferiores, os mais fracos. Isto é, quanto mais acima estiver um protocolo, mais seguro deve ser (com exceção das linhas 2 e 3, que não se pode afirmar ao certo qual contém o adversário mais poderoso). Também, quanto mais abaixo se localiza um protocolo, mais eficiente deve ser em termos de tempo de processamento.

O protocolo de Yang e Tan (2011a) nos oferece uma curiosa referência para análise dos demais protocolos. Ao mesmo tempo que ele foi mostrado seguro contra adversário forte, é discutível se ele é

Tabela 5.7: *Protocolos segundo poder adversário e problema computacional*

	Adversário	Problema computacional	
		BDH	Gap-BDH
1	Forte + autoridade mal intencionada		GNT1 LBG2-Gap(?) GOT1-Gap(?)
2	Forte	LBG1, LBG2 GOT1 GNT3	GNT1 LBG2-Gap GOT1-Gap YT(*)
3	Moderado + autoridade mal intencionada		GNT2
4	Moderado	GNT4	
5	Fraco		GOT2 LG YT(*)

(?) É possível que seja seguro, mas falta prova.
(*) Protocolo mostrado seguro contra adversário forte, mas requer assinatura que seja sempre verificada.

de fato um protocolo sem certificado e seguro no modelo forte, conforme exposto na Seção 5.6.5 (pág. 74). Se ele puder ser composto com um esquema de assinatura sem certificado, sem emparelhamento bilinear, e seguro sob o problema Gap-DH, então terá um dos melhores desempenhos, comparável aos mais eficientes (que são seguros no modelo fraco).

5.9.2 Testes Comparativos

As tabelas 5.8 a 5.10 contabilizam a quantidade de operações que cada participante precisa calcular em uma sessão de acordo de chave, para cada protocolo. As linhas superiores da tabela apresentam as operações que tipicamente são mais caras computacionalmente.

As linhas com tempos em segundos referem-se a tomadas de tempo sobre codificações dos protocolos realizadas com a biblioteca criptográfica Relic (versão 0.2.3), que é escrita em linguagem de programação C (Aranha e Gouvêa, 2010). As tabelas 5.8 a 5.10 apresentam tempos que se referem às execuções empregando as curvas binárias B-271 e B-1223, presentes na biblioteca. Essas curvas são supersingulares e apresentam nível de segurança mínimo de 70 bits e 128 bits respectivamente. Os tempos são contados desde o momento da escolha do valor secreto temporário rP até o cálculo da chave de sessão SK. O ambiente de testes para simulação dos protocolos inclui um computador com processador Intel Core 2 Duo T5450 (1.66Ghz e 2MB de cache L2), 2GB de memória RAM e sistema operacional Ubuntu 11.04. Apesar do processador ter dois núcleos, os programas são executados em apenas uma única *thread*. Os programas são compilados e executados em 64 bits.

Nessas mesmas tabelas, encontram-se dados relativos a dois cenários diferentes. O cenário normal, sem pré-computação, exibe os protocolos da maneira como eles são definidos e apresentados neste capítulo. O cenário com pré-computação é considerado quando um usuário A se comunica frequentemente com um usuário B; assim do ponto de vista do usuário A, alguns valores referentes a B não precisam ser computados novamente, bastando apenas serem armazenados. É importante notar os valores pré-calculados derivados da chave secreta devem ser armazenados de forma segura.

Tabela 5.8: *Protocolos seguros sob BDH, com duas versões para LBG*

		Sem pré-computação				
		LBG1	LBG2	GOT1	GNT3	GNT4
Emparelhamentos		10	8	8	6	4
Exponenciações em G_T		0	4	0	0	0
Multiplicações em G_T		4	4	2	2	0
Multiplicações em G		7	5	7	7	7
Adições em G		0	0	4	4	8
Elementos de G transmitidos(*)		2	2	2	2	2
Modelo de segurança		LBG	LBG	LBG	LBG	SJ^+
Poder do adversário		Forte	Forte	Forte	Forte	Moderado
Tempo (s)	B-271	0,145	0,124	0,117	0,090	0,063
	B-1223	8,268	6,857	6,660	5,047	3,435
Mais veloz que LBG2	B-271			5,65%	27,42%	49,19%
	B-1223			2,87%	26,40%	49,91%

(*) Pode ser reduzido (ver Seção 5.3)

		Com pré-computação				
		LBG1	LBG2	GOT1	GNT3	GNT4
Emparelhamentos		4	2	2	2	2
Exponenciações em G_T		0	2	0	0	0
Multiplicações em G_T		2	2	0	0	0
Multiplicações em G		5	4	5	5	5
Adições em G		0	0	4	4	4
Tempo (s)	B-271	0,06	0,036	0,033	0,033	0,033
	B-1223	3,38	1,865	1,768	1,765	1,761
Mais veloz que LBG2	B-271			8,33%	8,33%	8,33%
	B-1223			5,20%	5,36%	5,58%

Na Tabela 5.8, são agrupados os protocolos que são mostrados seguros sob o problema computacional BDH. Neste grupo, o protocolo GNT3 é o mais eficiente sob o modelo de segurança LBG de Lippold *et al.* (2009); em particular é cerca de 26% mais rápido que LBG2 (Lippold *et al.*, 2009). O protocolo GNT4 é o único seguro sob o modelo de Swanson e Jao (2009) melhorado, sob o problema BDH. O GNT4 é mais eficiente que GNT3, cerca de 31%, isto é, o relaxamento do modelo de segurança mais forte LBG para o moderado SJ^+ permitiu um ganho de 31% em desempenho. No cenário com pré-computação de valores, os protocolos apresentam comportamento similar; em relação ao LBG2, nossas propostas GOT1, GNT3 e GNT4 são cerca de 8% mais velozes.

A Tabela 5.9 apresenta os protocolos seguros sob Gap-BDH. Nesse grupo de protocolos não conseguimos melhorar o tempo computacional do protocolo original LBG2-Gap, mas acrescentamos a propriedade de segurança de resistência a ataque da autoridade que frauda na geração de parâmetros, com o protocolo GNT1. Não podemos afirmar que LBG2-Gap e GOT1-Gap não são seguros contra autoridade mal intencionada; é possível que os três protocolos (GNT1, LBG2-Gap e GOT1-Gap) sejam equivalentes em propriedades de segurança, mas é necessário revisar as provas de segurança dos dois últimos. O protocolo GNT2 alcança segurança contra autoridade mal intencionada, sob o modelo de Swanson e Jao (2009), e é cerca de 24% mais veloz que LBG2-Gap e GNT1 (estes dois últimos têm segurança contra adversário forte). O protocolo GOT2 é o mais eficiente de todos, mas é seguro contra adversário fraco; ele é cerca de 21% a 27% mais rápido que o GNT2 e

Tabela 5.9: *Protocolos seguros sob Gap-BDH*

	Sem pré-computação				
	LBG2-Gap	GOT1-Gap	GNT1	GNT2	GOT2
Emparelhamentos	4	4	4	3	2
Exponenciações em G_T	2	0	0	0	0
Multiplicações em G_T	2	1	1	0	0
Multiplicações em G	5	7	7	7	10
Adições em G	0	2	4	4	1
Elementos de G transmitidos	2	2	2	2	3+ID(*)
Modelo de segurança	LBG	LBG	Mal-LBG	Mal-SJ$^+$	Fraco
Seguro contra autoridade mal intencionada	?	?	sim	sim	não
Tempo (s) B-271	0,062	0,061	0,062	0,047	0,037
B-1223	3,504	3,432	3,442	2,625	1,911
Mais veloz que LBG2 B-271		1,61%	0,00%	24,19%	40,32%
B-1223		2,05%	1,77%	25,09%	45,46%

(*) ID=identidade; único que não pode ser reduzido conforme Seção 5.3

	Com pré-computação				
	LBG2-Gap	GOT1-Gap	GNT1	GNT2	GOT2
Emparelhamentos	1	1	2	1	1
Exponenciações em G_T	1	0	0	0	1
Multiplicações em G_T	1	0	0	0	0
Multiplicações em G	4	5	5	5	8
Adições em G	0	2	4	2	1
Tempo (s) B-271	0,019	0,019	0,033	0,019	0,024
B-1223	0,992	0,955	1,769	0,955	1,112

cerca de 40% a 45% mais eficiente que LBG2-Gap.

Vale observar que todos os tempos nas tabelas são viáveis para uso prático, especialmente se forem aplicadas as adaptações para implementação de emparelhamento assimétrico (discutidas na seção a seguir).

Na Tabela 5.10, mostramos os protocolos LG (Lippold e González Nieto, 2010) e GOT2 que são seguros contra adversários fracos. O protocolo YT (Yang e Tan, 2011a) é demonstrado seguro contra adversário forte, no entanto embute premissas restritivas em seu projeto, que podem enfraquecê-lo com relação ao problema computacional subjacente ou, dependendo do esquema de assinatura escolhido, torná-lo menos eficiente e com características de protocolos que requerem infraestrutura de chaves públicas, uma vez que uma chave pública e uma assinatura são transmitidos e verificados *em todo* início de sessão.

5.10 Adaptação para Emparelhamento Assimétrico

Dentre os protocolos descritos neste capítulo, o único que pode ser implementado diretamente sobre emparelhamentos simétricos e assimétricos é o GOT2. Os demais precisam de adaptações para que seja possível usá-los também com um emparelhamento assimétrico na forma $e : G_1 \times G_2 \to G_T$, onde $G_1 \neq G_2$.

As adaptações requeridas sobre o protocolo de Lippold *et al.* (2009) são simples. Por exemplo,

Tabela 5.10: *Protocolos seguros em modelos mais fracos*

	Sem pré-computação			Com pré-computação		
	LG	YT	GOT2	LG	YT	GOT2
Emparelhamentos	3	0(*)	2	3	0(*)	1
Exponenciações em G_T	0	0	0	0	0	1
Multiplicações em G_T	0	0	0	0	0	0
Multiplicações em G	13	8(*)	10	13	5(*)	8
Adições em G	0	4	1	0	2	1
Problema computacional	DBDH	Gap-DH	Gap-BDH	DBDH	Gap-DH	Gap-BDH
Demonstração	padrão	ROM	ROM	padrão	ROM	ROM

(*) mais uma verificação de assinatura, usando chave pública certificada da autoridade

para o LBG2 sob Gap-BDH, essencialmente basta um hash adicional H_2 que mapeia identidades sobre o grupo G_2; e o valor público da identidade ID_A, por exemplo, passa a ser um par $(Q_{A_1}, Q_{A_2}) = (H_1(ID_A), H_2(ID_A)) \in G_1 \times G_2$. A chave secreta parcial passa a ter dois componentes, isto é, $d_A = (d_{A_1}, d_{A_2}) = (sQ_{A_1}, sQ_{A_2}) \in G_1 \times G_2$. Durante uma sessão, A calcula $K = e(sP, Q_{B_2})^{r_A} \cdot e(r_B P, d_{A_2})$, enquanto B obtém o mesmo valor calculando $K = e(sP, Q_{A_2})^{r_B} \cdot e(r_A P, d_{B_2})$. Os cálculos de L e N são ajustados de forma análoga. Com essas medidas, o protocolo LBG2-Gap pode ficar cerca de 27 vezes mais rápido com emparelhamento assimétrico (e curvas BN) em nível de segurança de 128 bits, quando comparado com implementação sobre curvas binárias e emparelhamento simétrico (usando a biblioteca Relic).

Para adaptar os demais protocolos para uso com emparelhamento assimétrico, são necessárias mais modificações, e isso é um ponto de desvantagem. Por exemplo, o protocolo GNT2 teria as fases de inicialização do sistema e de geração de chaves de usuário reescritas como indicado nas tabelas 5.11 e 5.12, pelas quais se vê que os parâmetros públicos são maiores.

Tabela 5.11: *Inicialização do sistema, sobre emparelhamento assimétrico*

Parâmetro de segurança	k
Grupos algébricos	G_1, de ordem prima q e gerador P_1 G_2, de ordem prima q e gerador P_2 G_T, de ordem prima q
Emparelhamento bilinear	$e : G_1 \times G_2 \to G_T$
Par de chaves mestra	$s \xleftarrow{\$} \mathbb{Z}_q^*$ (msk, chave mestra secreta) $(sP_1, sP_2) \in G_1 \times G_2$ (mpk, chave mestra pública)
Funções de hash	$H : \{0,1\}^* \to \{0,1\}^k$ $H_1 : \{0,1\}^* \to G_1$ $H_2 : \{0,1\}^* \to G_2$
Parâmetros públicos	$\langle q, G_1, G_2, G_T, e, k, P_1, P_2, sP_1, sP_2, H, H_1, H_2 \rangle$

A cada sessão, para estabelecerem uma chave secreta em comum, os usuários A e B escolhem seus valores secretos temporários, respectivamente $r_A \xleftarrow{\$} \mathbb{Z}_q^*$ e $r_B \xleftarrow{\$} \mathbb{Z}_q^*$. Considerando-se que A é

Tabela 5.12: *Chaves de usuário, sobre emparelhamento assimétrico*

Valores públicos	$ID_U \in \{0,1\}^*$ $Q_{U_1} = H_1(ID_U) \in G_1$ $Q_{U_2} = H_2(ID_U) \in G_2$	identidade do usuário
	$x_U P_1 \in G_1$ $x_U P_2 \in G_2$	chave pública, gerada pelo usuário
Valores secretos	$d_U = (d_{U_1}, d_{U_2}) \in G_1 \times G_2$ $d_{U_1} = sQ_{U_1} \in G_1$ $d_{U_2} = sQ_{U_2} \in G_2$	chave secreta parcial, gerada por KGC
	$x_U \xleftarrow{\$} \mathbb{Z}_q^*$	valor secreto, escolhido pelo usuário

quem inicia a comunicação e B responde, A calcula $r_A P_1$ e $r_A P_2$ enquanto B calcula $r_B P_1$. Então trocam as seguintes mensagens:

$$A \to B : E_A = (r_A P_1, r_A P_2, x_A P_1, x_A P_2)$$

$$B \to A : E_B = (r_B P_1, x_B P_1)$$

Após a fase de troca de mensagens, o participante A calcula a chave compartilhada para GNT2 da maneira abaixo:

$K = e(r_B P_1 + Q_{B_1}, r_A s P_2 + d_{A_2})$
$M = e(x_B P_1 + Q_{B_1}, x_A s P_2 + d_{A_2})$
$N = e(Q_{B_1}, d_{A_2})$
$Z = (x_A x_B P_1, x_A r_B P_1, r_A r_B P_1, r_A x_B P_1)$
$SK = H(A, B, E_A, E_B, Z, K, M, N)$.

E B, como receptor, calcula a chave secreta como:

$K = e(r_B s P_1 + d_{B_1}, r_A P_2 + Q_{A_2})$
$M = e(x_B s P_1 + d_{B_1}, x_A P_2 + Q_{A_2})$
$N = e(d_{B_1}, Q_{A_2})$
$Z = (x_B x_A P_1, r_B x_A P_1, r_B r_A P_1, x_B r_A P_1)$
$SK = H(A, B, E_A, E_B, Z, K, M, N)$.

Os protocolos que usam parâmetros estendidos (ver Seção 5.2.2, pág. 65) precisam de adaptações análogas, porém com valores públicos como $(Q_{A_{11}}, Q_{A_{12}}, Q_{A_{21}}, Q_{A_{22}}) \in G_1^2 \times G_2^2$.

5.11 Síntese e Conclusões Parciais

Neste capítulo, apresentamos protocolos de acordo de chave CL-AKA com autenticação sem certificado que são mostrados seguros contra adversários de poder variado, do mais fraco ao mais forte, até o caso da autoridade desonesta. Com as propostas dos protocolos GOT1, GOT2, GNT1, GNT2, GNT3 e GNT4, exploramos formas possíveis de otimizar o desempenho dos protocolos mais

seguros e aumentar a segurança nos casos mais fracos. Com os testes preliminares de desempenho, mostramos a viabilidade prática de uso de tais protocolos, mesmo em níveis de segurança mais elevados, especialmente se for feito uso de algoritmos otimizados para cálculo de emparelhamentos assimétricos.

Detectamos uma dificuldade peculiar em se construir protocolos seguros contra autoridade mal intencionada sob o problema computacional BDH. Suspeitamos que, com a chave pública sem certificado na formulação AP, não é possível preencher essa lacuna. Isto é, acreditamos não ser viável chegar a um protocolo seguro contra autoridade desonesta sem recair na hipótese do problema Gap-BDH, para chaves de usuário definidas como na Seção 5.2. Tal suspeita surge quando analisamos as provas de segurança para o caso da autoridade mal intencionada. Retomaremos esse assunto no capítulo a seguir, onde apresentamos as provas formais de segurança dos protocolos deste capítulo.

Capítulo 6

Segurança dos Protocolos CL-AKA

O objetivo deste capítulo é mostrar as provas de segurança por redução de problemas computacionais dos protocolos apresentados no Capítulo 5. Começaremos com a apresentação das demonstrações que envolvem os modelos mais fortes pois são as mais completas; a partir dessas, mostramos as provas sob modelos contra autoridade mal intencionada e os casos restantes, retomando partes das provas anteriores sempre que possível.

6.1 Segurança do GNT3 no Modelo LBG sob BDH

Nesta seção, apresentamos uma redução do problema Diffie-Hellman Bilinear (BDH) para o problema de se construir um algoritmo que diferencie um número aleatório de uma chave secreta calculada pelo protocolo GNT3. A redução indica que se houver um adversário com vantagem não negligenciável contra o protocolo, sob o modelo de adversário forte da (ver Seção 4.4.1, pág. 54), então existe algoritmo de tempo polinomial que resolve o BDH. A seguir, enunciamos o teorema que relaciona a vantagem do adversário com a do solucionador do BDH; na sequência, apresentamos sua demonstração, sob o modelo de oráculo aleatório (Bellare e Rogaway, 1993).

Teorema 6.1 *Sob a hipótese de dificuldade do problema BDH, se as funções H, H_1 e H_2 são modeladas como oráculos aleatórios, então GNT3 é um protocolo CL-AKA seguro no modelo LBG.*

6.1.1 Demonstração do Teorema 6.1

Suponha, por absurdo, que existe um algoritmo $\mathcal{A}$ de tempo polinomial probabilístico com vantagem não negligenciável em quebrar o protocolo. Vamos mostrar como construir um algoritmo $\mathcal{S}$ que resolve o problema BDH.

O algoritmo $\mathcal{S}$ recebe como entrada uma quádrupla (P, aP, bP, cP) referente a um desafio BDH e gera como resposta o valor $e(cP, abP)$. $\mathcal{S}$ simula uma execução real do protocolo e interage com o adversário $\mathcal{A}$, nos moldes do jogo descrito na Seção 4.4.1; por esse motivo, $\mathcal{S}$ será chamado de simulador. Se $\mathcal{A}$ for incapaz de detectar que $\mathcal{S}$ não é real, prossegue até o final do jogo e obtém vantagem contra o protocolo. O simulador usa os passos do adversário para calcular, então, a resposta ao desafio BDH, que é armazenada na variável $answerBDH$.

Suponha que q_0 denota o número máximo de sessões nas quais um usuário pode participar e q_1 denota o máximo de consultas que $\mathcal{A}$ pode fazer a H_1, com q_0 e q_1 relacionados polinomialmente com o parâmetro de segurança k que gerou os parâmetros do sistema.

Tabela 6.1: *Casos válidos de corrompimento dos participantes da sessão de Teste*

Adversário	1		2		3		4		5		6		7		8		9	
consulta	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B
RevealPartial	r	r	r	r	r	r	r	r		r	r			r	r			
RevealSV *ou*			r	r		r	r		r			r	r	r	r	r	r	r
ReplacePK			m	m		m	m		m			m	m	m	m	m	m	m
RevealEph *ou*	r	r			r			r	r	r	r	r	r			r	r	r
Adv. é ativo	r	m			r			m	r	m	r	m	r			m	r	m

(r)–revela componente secreto (m)–modifica/escolhe valor do componente secreto

Fase de Preparo

Antes do início do jogo entre o simulador $\mathcal{S}$ e o adversário $\mathcal{A}$, o simulador tenta adivinhar qual será a sessão sobre a qual o adversário realizará a consulta de Teste e quem serão os participantes dessa sessão. Considere um conjunto de usuários honestos previamente estabelecidos $\mathcal{U} = \{U_1, \ldots, U_n\}$, com $n \leq q_1$, e uma lista correspondente com valores $(ID_i, x_{ID_i}, x_{ID_i}P)$. O simulador escolhe $I, J \in \{1, \ldots, n\}$, com $I \neq J$, e a sessão $\Pi^t_{I,J}$, onde $t \in \{1, \ldots, q_0\}$. A probabilidade de $\mathcal{S}$ fazer as escolhas corretas dos participantes e obter sucesso é:

$$Pr[sucesso] \geq \frac{1}{q_1(q_1 - 1)} > \frac{1}{q_1^2}$$

A probabilidade do simulador acertar a sessão é $1/q_0$. Portanto, $\mathcal{S}$ adivinha corretamente a sessão de Teste com probabilidade maior que

$$\frac{1}{q_0 q_1^2}$$

Se $\mathcal{S}$ fizer escolhas incorretas, aborta o jogo. Por outro lado, se o adversário realizar alguma operação não permitida, o jogo também é abortado por $\mathcal{S}$. No entanto, se $\mathcal{A}$ apresenta vantagem não negligenciável, é porque realiza a consulta de Teste sobre uma sessão *fresh*, respeitando a definição de segurança. Em outras palavras, o adversário revela (ou modifica) no máximo dois dos três componentes secretos associados a cada participante.

Na Tabela 6.1, são listadas as possibilidades que o adversário possui para revelar ou modificar valores secretos associados à sessão de Teste e seus participantes, sem corromper integralmente a sessão. Os participantes da sessão de Teste são denotados por A e B, que equivalem respectivamente às identidades ID_I e ID_J. Sem perda de generalidade, considere que A é quem inicia a sessão e B é quem responde. Ao todo são nove casos válidos. O simulador tenta adivinhar qual desses casos o adversário explorará para quebrar o protocolo. Se $\mathcal{S}$ errar na escolha, aborta o jogo. Se acertar, então a probabilidade de $\mathcal{S}$ completar o jogo será maior que

$$\frac{1}{9 q_0 q_1^2}$$

Apenas os casos em que a sessão de Teste possui uma sessão com *matching* $\Pi^s_{J,I}$ são apresentados na Tabela 6.1. O simulador trata a situação em que a sessão de Teste não possui *matching*, como um subcaso dos casos em que B tem no máximo um dentre os dois de seus segredos de longa duração comprometidos (valor secreto ou chave secreta parcial).

O simulador seleciona os parâmetros do sistema. Para tratar os casos 1 a 4, $\mathcal{S}$ escolhe ao acaso a chave mestra secreta s e inicia a execução do adversário $\mathcal{A}$ com $mpk = sP$ e demais parâmetros públicos. Para os casos 5 a 9, $\mathcal{S}$ define $mpk = aP$ e inicia a simulação.

Respostas aos Oráculos

Uma vez iniciado o jogo, o simulador deve responder às consultas realizadas pelo adversário aos oráculos disponíveis. O comportamento do simulador para tratar essas consultas varia de acordo com cada um dos nove casos. Os oráculos H e StrongRevealSessionKey, que respectivamente calcula e revela a chave de sessão, são os mais críticos a serem tratados pelo simulador. Por esse motivo, eles serão descritos mais detalhadamente na seção 6.1.2. Os demais oráculos são tratados como segue:

H$_1(ID_i)$ Se $\mathcal{S}$ está simulando os casos 1 a 4, esse oráculo é executado como numa real execução de H_1, isto é, $\mathcal{S}$ sorteia e responde um elemento aleatório de G. Nos casos 5 a 9, o simulador convenientemente define $H_1(A)$ e/ou $H_1(B)$ como uma das entradas do desafio BDH:

- casos 5, 7 e 9: $H_1(A) = bP$, ou seja, Q_{A_1} é definido como bP
- casos 6 e 8: $H_1(B) = bP$, ou seja, Q_{B_1} é definido como bP
- caso 9: $H_1(B) = cP$, ou seja, Q_{B_1} é definido como cP

Para os demais participantes, nos casos 5 a 9, o simulador escolhe $l_i \in \mathbb{Z}_q$ ao acaso e responde l_iP. Todas as respostas a H_1 e os valores l_i são armazenadas em uma lista, para posterior uso por $\mathcal{S}$.

H$_2(ID_i)$ O simulador responde de forma análoga à H_1, porém sorteia $z, y \in \mathbb{Z}_q$ (ou sorteia $z, y_A, y_B \in \mathbb{Z}_q$ para o caso 9) e define:

- casos 5 e 7: $H_2(A) = yP - zbP$, ou seja, Q_{A_2} é definido como $yP - zQ_{A_1}$
- casos 6 e 8: $H_2(B) = yP - zbP$, ou seja, Q_{B_2} é definido como $yP - zQ_{B_2}$
- caso 9: $H_2(A) = y_AP - zbP$, isto é, $Q_{A_2} = y_AP - zQ_{A_1}$ e $H_2(B) = y_BP - zcP$, ou seja, $Q_{B_2} = y_BP - zQ_{B_1}$

Para os demais participantes, nos casos 5 a 9, o simulador escolhe $p_i \in \mathbb{Z}_q$ ao acaso e responde p_iP. Todas as respostas a H_2 e os valores p_i são armazenadas em uma lista.

EstablishParty$(ID_i, x_{ID_i}P)$ O simulador cria o usuário desonesto com identificador (ID_i), caso ainda não exista, chamando os oráculos H_1, H_2. $\mathcal{S}$ registra a chave pública $x_{ID_i}P$, define $x_{ID_i} = \perp$ (pois $\mathcal{S}$ desconhece o valor secreto) e entrega ao adversário a chave secreta parcial $(d_{ID_{i1}} = l_iaP, d_{ID_{i2}} = p_iaP)$.

Send$(\Pi^s_{i,j}, m)$ Se a sessão $\Pi^s_{i,j}$ ainda não existe, cria uma sessão para dono ID_i e parceiro ID_j, com estado *indefinido* e com o papel de

- emissor, se $m = \lambda$, ou
- receptor, em caso contrário.

Se $m = \lambda$

- Se $\Pi_{i,j}^s$ é sessão de Teste e $\mathcal{S}$ está tratando:
  - caso 2 ou 4, então define $r_{ID_i}P = aP$ e $r_{ID_i} = \perp$
  - caso 8, então define $r_{ID_i}P = cP$ e $r_{ID_i} = \perp$
- caso contrário escolhe ao acaso $r_{ID_i} \in \mathbb{Z}_q$ e calcula $r_{ID_i}P$
- entrega $r_{ID_i}P$ ao adversário

Se $m \neq \lambda$

- Se o papel da sessão é de receptor
  - Se $\Pi_{i,j}^s$ é sessão de Teste e $\mathcal{S}$ está tratando:
    - caso 2 ou 3, então define $r_{ID_i}P = bP$ e $r_{ID_i} = \perp$
    - caso 7, então define $r_{ID_i}P = cP$ e $r_{ID_i} = \perp$
  - caso contrário escolhe ao acaso $r_{ID_i} \in \mathbb{Z}_q$ e calcula $r_{ID_i}P$
  - $\mathcal{S}$ verifica a validade de m:
    - se $m \notin G^2$, atualiza o estado da sessão para *rejeitado*.
    - caso contrário, define $m = (m_1, m_2) \in G^2$, $r_{ID_j}P = m_1$, se $m_2 \neq x_{ID_j}$ chama ReplacePublicKey(ID_j, m_2) e atualiza o estado da sessão para *aceito*
  - entrega $r_{ID_i}P$ ao adversário
- Se o papel da sessão é de emissor, $\mathcal{S}$ verifica a validade de m:
  - se $m \notin G^2$, atualiza o estado da sessão para *rejeitado*.
  - caso contrário, define $m = (m_1, m_2) \in G^2$, $r_{ID_j}P = m_1$, se $m_2 \neq x_{ID_j}$ chama ReplacePublicKey(ID_j, m_2), e atualiza o estado da sessão para *aceito*.

RevealMasterKey Se $\mathcal{S}$ está simulando os casos 1, 2, 3 ou 4, responde a chave mestra secreta s, caso contrário aborta.

RevealPartial(ID_i) Se $ID_i = A$ e $\mathcal{S}$ está tratando os casos 5, 7 ou 9, aborta. Se $ID_i = B$ e $\mathcal{S}$ está tratando os casos 6, 8 ou 9, aborta. Caso contrário, responde a chave secreta parcial $(d_{ID_{i1}} = l_i aP, d_{ID_{i2}} = p_i aP)$.

RevealSecretValue(ID_i) Se $x_{ID_i} = \perp$, aborta. Se $ID_i = A$ e $\mathcal{S}$ está tratando os casos 1, 3 ou 6, aborta. Se $ID_i = B$ e $\mathcal{S}$ está tratando os casos 1, 4 ou 5, aborta. Caso contrário, responde o valor secreto x_{ID_i}.

ReplacePublicKey$(ID_i, x_{ID_i}P)$ Se $ID_i = A$ e $\mathcal{S}$ está tratando os casos 1, 3 ou 6, aborta. Se $ID_i = B$ e $\mathcal{S}$ está tratando os casos 1, 4 ou 5, aborta. Caso contrário, substitui a chave pública por $x_{ID_i}P$ e define $x_{ID_i} = \perp$.

RevealEphemeral$(\Pi_{i,j}^s)$ Se $ID_i = A$ e $\mathcal{S}$ está tratando os casos 2, 4 ou 8, aborta. Se $ID_i = B$ e $\mathcal{S}$ está tratando os casos 2, 3 ou 7, aborta. Caso contrário, devolve o valor secreto r_{ID_i} da sessão. Supomos que o adversário não consulta esse oráculo nos casos em que pode escolher ativamente o valor temporário da sessão.

StrongRevealSessionKey$(\Pi_{i,j}^s)$ Se $(\Pi_{i,j}^s)$ for a sessão de Teste, aborta. Caso contrário, prossegue com os passos descritos na seção 6.1.2.

H$(ID_i, ID_j, E_i, E_j, Z, K, L, M, N)$ Prossegue com os passos descritos na seção 6.1.2.

Test$(\Pi^s_{i,j})$ Se $\Pi^s_{i,j}$ não é a sessão de Teste, aborta. Senão, $\mathcal{S}$ joga uma moeda $b \in \{0,1\}$. Se $b = 0$, devolve a chave de sessão; caso contrário sorteia e devolve um número aleatório $r \in \{0,1\}^k$ para o adversário.

Finalização do Jogo

Assim que $\mathcal{A}$ finaliza o jogo, $\mathcal{S}$ devolve $answerBDH$. Se a probabilidade de sucesso de $\mathcal{A}$ é $Pr[\mathcal{A}]$, então a probabilidade de sucesso de $\mathcal{S}$ é $Pr[\mathcal{S}] \geq Pr[\mathcal{A}]/(9q_0q_1^2)$. Se $\mathcal{A}$ tem vantagem não negligenciável em diferenciar um número aleatório da chave de sessão é porque $\mathcal{A}$ consultou H com valores corretos de Z, K, L, M, N. Nesse caso, $answerBDH$ contém o valor $e(cP, abP)$, conforme cálculos apresentados na seção 6.1.2. Então $\mathcal{S}$ resolve o problema Gap-BDH em tempo polinomial em k, o que contradiz a hipótese de dificuldade do Gap-BDH. □

6.1.2 Oráculos H e StrongRevealSessionKey

Quando o oráculo StrongRevealSessionKey é consultado pelo adversário, $\mathcal{S}$ deve informar o valor da chave de uma dada sessão $\Pi^s_{i,j}$, caso ela não seja a sessão de Teste. Entretanto, nas sessões que não são a de Teste e que envolvem os participantes A e/ou B do Teste, o simulador não é capaz de calcular por si só o valor da chave de sessão, pois desconhece alguns valores secretos. Se $\mathcal{S}$ informar valores diferentes de chave de sessão para duas sessões *matching*, o adversário detecta a incoerência e aborta o jogo. Se isso ocorre, $\mathcal{S}$ é incapaz de aproveitar a vantagem de $\mathcal{A}$ para resolver BDH; por isso passamos a descrever como $\mathcal{S}$ procede de modo a ser sempre consistente.

O simulador mantém uma lista H^{lst}, inicialmente vazia, com entradas na forma $(ID_i, ID_j, E_i, E_j, Z, K, L, M, N, SK)$. Como H é função, $\mathcal{S}$ deve responder o mesmo valor SK de chave de sessão para as mesmas entradas.

Se $\mathcal{A}$ consultar H antes de eventualmente solicitar StrongRevealSessionKey, basta $\mathcal{S}$ calcular o valor da chave de sessão SK e atualizar a lista. Se StrongRevealSessionKey é consultado antes e $\mathcal{S}$ não é capaz de calcular todos os valores Z, K, L, M, N, então $\mathcal{S}$ sorteia SK e insere um novo registro na lista H^{lst} com os valores $(ID_i, ID_j, E_i, E_j, \ldots, SK)$, preenchendo tanto quanto possível os valores corretos de Z, K, L, M, N. Em uma eventual consulta posterior a H referente a uma sessão *matching*, $\mathcal{S}$ atualiza H^{lst} com os dados faltantes Z, K, L, M, N e responde o mesmo valor SK.

A seguir, vamos dividir a análise em dois grandes casos, em que H e StrongRevealSessionKey são consultados sobre sessões que:

1. não envolvem os participantes A e B da sessão de Teste e
2. envolvem o participante A e/ou B da sessão de Teste

Sessões que Não Envolvem A e B Alvos do Teste

Considere que $\mathcal{A}$ consulta H ou StrongRevealSessionKey sobre participantes ID_i e ID_j, ambos diferentes de A, B. Sem perda de generalidade, suponha que ID_i é quem inicia a sessão.

O simulador conhece as chaves secretas de ID_i e de ID_j a não ser nos seguintes casos:

Tabela 6.2: *Casos possíveis e inserção do desafio BDH*

	1		2		3		4		5		6		7		8		9	
Chave	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B
ID: Q_1									bP			bP	bP			bP	bP	cP
PK: xP	aP	bP	x	x	aP	x	x	bP	x	cP	cP	x	x	x	x	x	x	x
Eph: rP		x	aP	bP		bP	aP	x		x		x		cP	cP	x		x
Desafio em:	Z_1		Z_3		Z_2		Z_4		$mpk = aP$									
									M_1		M_2		K_1		K_2		K_3	

(aP, bP, cP)–desafio BDH (x)–simulador desconhece componente secreto

- $\mathcal{A}$ substitui a chave pública de ID_i e, portanto, $\mathcal{S}$ não conhece x_i
- $\mathcal{A}$ substitui a chave pública de ID_j e, portanto, $\mathcal{S}$ não conhece x_j
- $\mathcal{A}$ escolhe ativamente o valor temporário de ID_j e, portanto, $\mathcal{S}$ não conhece r_j

Podemos supor que $\mathcal{S}$ conhece r_i, pois, se $\mathcal{A}$ ativamente alterar r_iP e entregar outro valor a ID_j, não haverá sessão com *matching* (a não ser com probabilidade negligenciável).

Sejam os componentes de Z nomeados como segue:

$$Z = (\underbrace{x_i x_j P}_{Z_1}, \underbrace{x_i r_j P}_{Z_2}, \underbrace{r_i r_j P}_{Z_3}, \underbrace{r_i x_j P}_{Z_4})$$

No pior dos casos, $\mathcal{A}$ consulta StrongRevealSessionKey antes de consultar H e $\mathcal{S}$ é incapaz de calcular os valores Z_1, Z_2. Porém $\mathcal{S}$ pode ainda responder de forma consistente, numa eventual consulta posterior de $\mathcal{A}$ a $H(ID_i, ID_j, E_i, E_j, (Z_1, Z_2, Z_3, Z_4), K, L, M, N)$, verificando se $e(x_iP, x_jP) \stackrel{?}{=} e(Z_1, P)$ e se $e(x_iP, r_jP) \stackrel{?}{=} e(Z_2, P)$. Se ambas igualdades valem e demais entradas de H correspondem aos valores que $\mathcal{S}$ consegue calcular, então $\mathcal{S}$ responde a mesma chave SK, pois se trata de sessões com *matching*.

Sessões que Envolvem A ou B Alvos do Teste

O simulador embute os valores do desafio BDH em pontos estratégicos do protocolo, de modo a induzir o adversário a realizar cálculos com esses valores. Para cada uma das nove possibilidades que $\mathcal{A}$ possui para tentar quebrar a segurança do protocolo, $\mathcal{S}$ insere os componentes do desafio em parâmetros específicos dos participantes A e B da sessão de Teste, ou em parâmetros do sistema. Na Tabela 6.2, são relacionadas as possíveis estratégias que $\mathcal{A}$ pode empregar e que chaves são associadas aos valores do desafio.

A última linha da Tabela 6.2 indica as variáveis que ocorrem no protocolo e que capturam o cálculo de $e(cP, abP)$, ou seja, a resposta do desafio. Obviamente $\mathcal{S}$ não sabe calcular essa resposta, mas mostraremos que se $\mathcal{A}$ apresenta vantagem não negligenciável, é porque conhece elementos suficientes para que a solução seja calculada. Tais elementos são entregues por $\mathcal{A}$ ao simulador, sempre que ele realiza consultas ao oráculo H. Observe que as variáveis K e L do protocolo podem ser reescritas como indicado na Tabela 6.3. O fatores de M e N e os componentes de Z também são nomeados para facilitar a leitura dos cálculos.

Quando $\mathcal{S}$ embute uma das entradas do desafio BDH em uma chave, automaticamente torna-se desconhecido o respectivo valor secreto. Por exemplo, quando aP é atribuído como valor de chave pública de A, isto é, $x_AP = aP$, $\mathcal{S}$ desconhece x_A por desconhecer a. Nos casos 5 a 9, a chave

pública mestra recebe o valor aP e, por isso, $\mathcal{S}$ não tem acesso ao valor da chave mestra secreta. Quando $Q_{A_1} = bP$, nos casos 5, 7 e 9, $\mathcal{S}$ desconhece a chave parcial secreta $d_{A_1} = abP$, pois não sabe calcular ab. Na Tabela 6.2, são indicados com "x" outros componentes secretos aos quais $\mathcal{S}$ não tem acesso. São os casos em que $\mathcal{A}$ é permitido substituir a chave pública ou escolher ativamente o valor temporário de sessão, preservando ainda a característica de sessão *fresh* e com *matching*.

Tabela 6.3: *Nomenclatura das Variáveis*

$$K = \underbrace{e(r_BP, d_{A_1})}_{K_1} \cdot \underbrace{e(Q_{B_1}, r_AsP)}_{K_2} \cdot \underbrace{e(Q_{B_1}, d_{A_1})}_{K_3} \cdot \underbrace{e(r_BP, r_AsP)}_{K_4}$$

$$L = \underbrace{e(r_BP, d_{A_2})}_{L_1} \cdot \underbrace{e(Q_{B_2}, r_AsP)}_{L_2} \cdot \underbrace{e(Q_{B_2}, d_{A_2})}_{L_3} \cdot \underbrace{e(r_BP, r_AsP)}_{L_4(=K_4)}$$

$$M = \underbrace{e(x_BP, d_{A_1})}_{M_1} \cdot \underbrace{e(Q_{B_1}, x_AsP)}_{M_2}$$

$$N = \underbrace{e(x_BP, d_{A_2})}_{N_1} \cdot \underbrace{e(Q_{B_2}, x_AsP)}_{N_2} \quad Z = (\underbrace{x_Ax_BP}_{Z_1}, \underbrace{x_Ar_BP}_{Z_2}, \underbrace{r_Ar_BP}_{Z_3}, \underbrace{r_Ax_BP}_{Z_4})$$

De acordo com a definição de sessão *fresh*, se não houver sessão *matching*, o participante que responde ao iniciador não pode ser totalmente corrompido. O simulador precisa tratar corretamente as situações em que B se envolve em sessões integralmente corrompidas, isto é, com um participante C desonesto. Esse cenário é tratado como subcaso dos casos 1, 4, 5, 6, 8 e 9. Analogamente, $\mathcal{S}$ deve lidar corretamente com as sessões em que A interage com o participante C desonesto; essa situação é tratada como subcaso de todos os nove casos. Na Tabela 6.4, esses cenários são detalhados para os casos 5 a 9, que são os mais críticos para $\mathcal{S}$.

Na sequência, vamos analisar separadamente cada um dos nove casos sobre sessões que envolvem A e/ou B, participantes do Teste. Para o caso 1, descrevemos em detalhes os passos de $\mathcal{S}$; os casos 2, 3 e 4 são análogos ao 1. Os casos 5 a 9 são similares entre si, pois neles fazemos uso dos teoremas *Trapdoor Test* para contornar as dificuldades do simulador em calcular valores que são dependentes de dados que desconhece, garantindo respostas sempre consistentes.

Caso 1

Se $\mathcal{S}$ adivinhar que $\mathcal{A}$ usará como estratégia o caso 1, o desafio BDH é embutido em Z_1. Para tanto, antes do início do jogo, $\mathcal{S}$ define $x_AP = aP$ e $x_BP = bP$, isto é, as chaves públicas dos alvos do Teste recebem as entradas do desafio.

(a) Se $\mathcal{A}$ consulta $H(A, B, E_A, E_B, Z, K, L, M, N)$
$\mathcal{S}$ verifica se Z_1 fornecido em Z contém abP, testando se $e(aP, bP) \stackrel{?}{=} e(P, Z_1)$
caso sim, $answerBDH \leftarrow e(cP, Z_1)$
$\mathcal{S}$ procura $(A, B, E_A, E_B, Z, K, L, M, N, *)$ na lista H^{lst}
se encontrar, responde o valor SK armazenado
caso contrário, procura $(A, B, E_A, E_B, (*, *, Z_3, Z_4), K, L, M, N, *)$
se encontrar, atualiza os valores Z_1, Z_2 e responde o valor SK armazenado
caso contrário, sorteia $SK \in \{0,1\}^k$, cria novo registro em H^{lst} e responde SK.
Se $\mathcal{A}$ consulta $H(A, C, E_A, E_C, Z, K, L, M, N)$, $\mathcal{S}$ realiza operações similares.

Tabela 6.4: *Casos na redução: cenário para o simulador*

Caso	Desafio BDH=(aP, bP, cP) embutido em $e(cP, abP)$, com $(mpk = aP)$ e:	Simulador desconhece (além de $msk = a$), nas sessões envolvendo:		
		A e B (alvos de Teste)	A e C $(C \neq B)$	C e B $(C \neq A)$
5	$Q_{A_1} = bP$ $Q_{A_2} = yP - zbP$ $x_B P = cP$ $M_1 = e(x_B P, d_{A_1})$	d_{A_1}, d_{A_2} x_A, x_B, r_B $x_A x_B P, x_A r_B P$ K_1, L_1, M_1, N_1	d_{A_1}, d_{A_2} x_A, x_C, r_A, r_C Z, K_4 K_1, L_1, M_1, N_1	x_C, x_B r_C, r_B Z, K_4
6	$Q_{B_1} = bP$ $Q_{B_2} = yP - zbP$ $x_A P = cP$ $M_2 = e(x_A P, d_{B_1})$	d_{B_1}, d_{B_2} x_A, x_B, r_B $x_A r_B P$ M_2, N_2	x_A, x_C r_A, r_C Z, K_4	d_{B_1}, d_{B_2} x_C, x_B, r_C, r_B Z, K_4 K_2, L_2, M_2, N_2
7	$Q_{A_1} = bP$ $Q_{A_2} = yP - zbP$ $r_B P = cP$ $K_1 = e(r_B P, d_{A_1})$	idem caso 5	idem caso 5	não se aplica: B não pode ser totalmente revelado
8	$Q_{B_1} = bP$ $Q_{B_2} = yP - zbP$ $r_A P = cP$ $K_2 = e(r_A P, d_{B_1})$	d_{B_1}, d_{B_2} x_A, x_B, r_A, r_B Z, K_4 K_2, L_2, M_2, N_2	idem caso 6	idem caso 6
9	$Q_{A_1} = bP$ $Q_{A_2} = y_A P - zbP$ $Q_{B_1} = cP$ $Q_{B_2} = y_B P - zcP$ $K_3 = e(Q_{B_1}, d_{A_1})$	d_{A_1}, d_{A_2} d_{B_1}, d_{B_2} x_A, x_B, r_B $x_A x_B P, x_A r_B P$ K_1, K_3, L_1, L_3, M, N	idem caso 5	idem caso 6

Se $\mathcal{A}$ consulta $H(C, B, E_C, E_B, Z, K, L, M, N)$ e $\mathcal{S}$ não encontrar essas entradas na lista,
procura $(C, B, E_C, E_B, *, *, L, M, N, *)$
se encontrar, atualiza os valores Z, K e responde o valor SK armazenado
caso contrário, sorteia $SK \in \{0,1\}^k$, cria novo registro em H^{lst} e responde SK.

(b) Se $\mathcal{A}$ consulta StrongRevealSessionKey(A, B, s)
$\mathcal{S}$ calcula Z_3, Z_4, K, L, M, N, usando dados que conhece
procura $(A, B, E_A, E_B, (*, *, Z_3, Z_4), K, L, M, N, *)$ em H^{lst}
se encontrar registros na forma $(A, B, E_A, E_B, (\overline{Z}_1, \overline{Z}_2, Z_3, Z_4), K, L, M, N, \overline{SK})$,
testa se $e(x_A P, x_B P) \stackrel{?}{=} e(\overline{Z}_1, P)$ e se $e(x_A P, r_B P) \stackrel{?}{=} e(\overline{Z}_2, P)$
se ambas igualdades valem, responde $\overline{SK}$
caso contrário, sorteia SK, cria novo registro com valores nulos para Z_1, Z_2, e responde SK.
Se $\mathcal{A}$ consulta StrongRevealSessionKey(A, C, s), $\mathcal{S}$ realiza operações similares.
Se $\mathcal{A}$ consulta StrongRevealSessionKey(C, B, s) e $\mathcal{S}$ não encontrar as entradas em H^{lst},

$\mathcal{S}$ calcula K_1, K_2, K_3, L, M, N, usando dados que conhece
procura $(C, B, E_C, E_B, (*, *, *, *), *, L, M, N, *)$
se encontrar registros na forma $(C, B, E_C, E_B, (\overline{Z}_1, \overline{Z}_2, \overline{Z}_3, \overline{Z}_4), \overline{K}, L, M, N, \overline{SK})$,
testa se $e(x_C P, x_B P) \stackrel{?}{=} e(\overline{Z}_1, P)$, se $e(x_C P, r_B P) \stackrel{?}{=} e(\overline{Z}_2, P)$,
se $e(r_C P, r_B P) \stackrel{?}{=} e(\overline{Z}_3, P)$, e se $e(r_C P, x_B P) \stackrel{?}{=} e(\overline{Z}_4, P)$
calcula $K = K_1 \cdot K_2 \cdot K_3 \cdot e(\overline{Z}_3, aP)$ e testa se $K \stackrel{?}{=} \overline{K}$
calcula $L = L_1 \cdot L_2 \cdot L_3 \cdot e(\overline{Z}_3, aP)$ e testa se $L \stackrel{?}{=} \overline{L}$
se todas igualdades valem, responde $\overline{SK}$
caso contrário, sorteia SK, cria novo registro com valores nulos para Z, K, e responde SK.

Caso 2

Similar ao caso 1, porém o desafio BDH é embutido em Z_3. $\mathcal{S}$ não sabe calcular Z_3, K_4 (além de Z_4 nos subcasos) mas é capaz responder de forma consistente, usando os dados conhecidos para calcular os demais valores e realizar os testes necessários.

Caso 3

Similar ao caso 1, porém o desafio BDH é embutido em Z_2. $\mathcal{S}$ não sabe calcular Z_2 (além de Z_1, Z_3, Z_4, K_4 nos subcasos), mas é capaz responder de forma consistente, usando as mesmas estratégias do caso 1.

Caso 4

Similar ao caso 1, porém o desafio BDH é embutido em Z_4. $\mathcal{S}$ não sabe calcular Z_3, Z_4, K_4 (além de Z_1, Z_2 nos subcasos), mas é capaz responder de forma consistente, usando as mesmas estratégias do caso 1.

Caso 5

Se $\mathcal{S}$ adivinhar que $\mathcal{A}$ usará como estratégia o caso 5, o desafio BDH é embutido em M_1. Para tanto, antes do início do jogo, $\mathcal{S}$ define $mpk = aP$, $Q_{A_1} = bP$ e $x_B P = cP$, isto é, a chave pública de B guarda uma das entradas do desafio e a chave secreta parcial d_{A_1} guarda abP. Para se tirar proveito do teorema *Trapdoor Test*, o oráculo H_2 é definido para que $Q_{A_2} = yP - zQ_{A_1}$, o que implica que a chave secreta parcial $d_{A_2} = yP - zd_{A_1}$. Os valores z e y são escolhidos e armazenados por $\mathcal{S}$. Por meio do *Trapdoor Test*, mesmo sem conhecer os valores de d_{A_1} e d_{A_2}, $\mathcal{S}$ é capaz de detectar quando $\mathcal{A}$ fornece $M_1 = e(cP, d_{A_1})$ e $N_1 = (cP, d_{A_2})$.

(a) Se $\mathcal{A}$ consulta $H(A, B, E_A, E_B, Z, K, L, M, N)$
$\mathcal{S}$ verifica se M_1 contém a resposta ao desafio, calculando
M_2, N_2 e $M_1 = \frac{M}{M_2}$ e $N_1 = \frac{N}{N_2}$ e testando se $(M_1)^z \cdot N_1 \stackrel{?}{=} e(aP, cP)^y$ (*Trapdoor Test*)
caso sim, $answerBDH \leftarrow M_1$
$\mathcal{S}$ verifica se já foi calculada chave para a sessão em questão ou uma *matching*
caso sim, atualiza entradas incompletas no registro de H^{lst} se for preciso e devolve a chave
caso contrário, sorteia SK, cria novo registro em H^{lst} e responde SK.
Se $\mathcal{A}$ consulta $H(A, C, E_A, E_C, Z, K, L, M, N)$ ou $H(C, B, E_C, E_B, Z, K, L, M, N)$

$\mathcal{S}$ realiza operações similares às do caso 1.

(b) Se $\mathcal{A}$ consulta StrongRevealSessionKey(A, B, s)
$\mathcal{S}$ calcula as variáveis de que é capaz e procura $(A, B, E_A, E_B, (*, *, Z_3, Z_4), *, *, *, *, *)$ em H^{lst}
se encontrar registros na forma $(A, B, E_A, E_B, \overline{Z}_1, \overline{Z}_2, Z_3, Z_4, \overline{K}, \overline{L}, \overline{M}, \overline{N}, \overline{SK})$,
calcula $K_1 = \frac{\overline{K}}{K_2 \cdot K_3 \cdot K_4}$ $L_1 = \frac{\overline{L}}{L_2 \cdot L_3 \cdot L_4}$ $M_1 = \frac{\overline{M}}{M_2}$ e $N_1 = \frac{\overline{N}}{N_2}$
testa se $(K_1)^z \cdot L_1 \stackrel{?}{=} e(aP, r_B P)^y$ e se $(M_1)^z \cdot N_1 \stackrel{?}{=} e(aP, cP)^y$ (*Trapdoor Test*)
$e(x_A P, x_B P) \stackrel{?}{=} e(\overline{Z}_1, P)$ e se $e(x_A P, r_B P) \stackrel{?}{=} e(\overline{Z}_2, P)$
se todas igualdades valem, responde $\overline{SK}$
caso contrário, sorteia SK, cria novo registro com valores nulos para as variáveis que
$\mathcal{S}$ desconhece e responde SK.
Se $\mathcal{A}$ consulta StrongRevealSessionKey(A, C, s), analogamente,
$\mathcal{S}$ calcula K_1, L_1, M_1, N_1 da mesma forma e
testa se $(K_1)^z \cdot L_1 \stackrel{?}{=} e(aP, r_c P)^y$ e se $(M_1)^z \cdot N_1 \stackrel{?}{=} e(aP, x_c P)^y$ (*Trapdoor Test*)
e prossegue igualmente.

Se $\mathcal{A}$ consulta StrongRevealSessionKey(C, B, s) e $\mathcal{S}$ não encontrar as entradas em H^{lst},
$\mathcal{S}$ calcula as variáveis de que é capaz e procura $(A, B, E_A, E_B, *, *, *, M, N, *)$ em H^{lst}
se encontrar registros na forma $(A, B, E_A, E_B, \overline{Z}, \overline{K}, \overline{L}, M, N, \overline{SK})$,
testa a consistência dos componentes de $\overline{Z}$ da mesma forma que no caso 1
calcula $K = K_1 \cdot K_2 \cdot K_3 \cdot e(\overline{Z}_3, aP)$ e testa se $K \stackrel{?}{=} \overline{K}$
calcula $L = L_1 \cdot L_2 \cdot L_3 \cdot e(\overline{Z}_3, aP)$ e testa se $L \stackrel{?}{=} \overline{L}$
se todas igualdades valem, responde $\overline{SK}$
caso contrário, sorteia SK, cria novo registro com valores nulos para as variáveis que
$\mathcal{S}$ desconhece e responde SK.

Caso 6

Similar ao caso 5, porém o desafio BDH é embutido em M_2, isto é, $\mathcal{S}$ define $mpk = aP$, $Q_{B_1} = bP$, $x_A P = cP$ e $Q_{B_2} = yP - zQ_{B_1}$. Vamos detalhar apenas o caso em que $\mathcal{A}$ solicita a chave de uma sessão que envolve B e um participante C desonesto.

(b) Se $\mathcal{A}$ consulta StrongRevealSessionKey(C, B, s) e $\mathcal{S}$ não encontrar as entradas em H^{lst},
$\mathcal{S}$ calcula as variáveis de que é capaz e procura $(C, B, E_C, E_B, *, *, *, *, *, *)$ em H^{lst}
se encontrar registros na forma $(C, B, E_C, E_B, \overline{Z}, \overline{K}, \overline{L}, \overline{M}, \overline{N}, \overline{SK})$,
testa a consistência dos componentes de $\overline{Z}$ da mesma forma que no caso 1
calcula $K_2 = \frac{\overline{K}}{K_1 \cdot K_3 \cdot e(\overline{Z}_3, aP)}$ $L_2 = \frac{\overline{L}}{L_1 \cdot L_3 \cdot e(\overline{Z}_3, aP)}$ $M_2 = \frac{\overline{M}}{M_1}$ e $N_2 = \frac{\overline{N}}{N_1}$
testa se $(K_2)^z \cdot L_2 \stackrel{?}{=} e(aP, r_c P)^y$ e se $(M_2)^z \cdot N_2 \stackrel{?}{=} e(aP, x_c P)^y$ (*Trapdoor Test*)
se todas igualdades valem, responde $\overline{SK}$
caso contrário, sorteia SK, cria novo registro com valores nulos para as variáveis que
$\mathcal{S}$ desconhece e responde SK.

Caso 7

Similar ao caso 5, porém o desafio BDH é embutido em K_1, isto é, $\mathcal{S}$ define $mpk = aP, Q_{A_1} = bP$, $r_B P = cP$ e $Q_{A_2} = yP - zQ_{A_1}$.

Caso 8

Similar ao caso 5, porém o desafio BDH é embutido em K_2, isto é, $\mathcal{S}$ define $mpk = aP, Q_{B_1} = bP$, $r_A P = cP$ e $Q_{B_2} = yP - zQ_{B_1}$. Os cálculos que testam a consistência dos dados fornecidos por $\mathcal{A}$ e que permitem $\mathcal{S}$ responder de forma coerente são similares aos apresentados no caso 6 (b).

Caso 9

Similar ao caso 5, porém o desafio BDH é embutido em K_3, isto é, $\mathcal{S}$ define $mpk = aP$, $Q_{A_1} = bP$ e $Q_{B_1} = cP$. Para poder realizar os testes de consistência, define $Q_{A_2} = y_A P - zQ_{A_1}$ e $Q_{B_2} = y_B P - zQ_{B_1}$.

(a) Se $\mathcal{A}$ consulta $H(A, B, E_A, E_B, Z, K, L, M, N)$
$\mathcal{S}$ verifica se K_3 contém a resposta ao desafio, calculando as variáveis que conhece e
$K' = \frac{K}{K_2 \cdot K_4}$ $L' = \frac{L}{L_2 \cdot L_4}$ $U_2 = [L']^{\frac{-1}{z}}$ $V_2 = [K']^{-z}$
$U_1 = \left[e(aP, y_A cP + y_B bP - \frac{y_A}{z}(r_B P + y_B P))\right]^{\frac{1}{1+z}}$
$V_1 = e(Q_{B_2}, y_A aP)^{\frac{1}{1+z}} \left[\frac{e(bP, y_B aP)}{e(r_B P, y_A P)}\right]^{\frac{z}{1+z}}$
$W_1 = \frac{1}{U_1} \cdot \left[\frac{K'}{U_2}\right]^{\frac{1}{1+z}}$ $W_2 = V_1 \cdot \left[\frac{L'}{V_2}\right]^{\frac{z}{1+z}}$
e testando se $\frac{W_2}{{W_1}^{z^2}} \stackrel{?}{=} \frac{e(aP,P)^{y_A y_B}}{[e(aP,cP)^{y_A} \cdot e(aP,bP)^{y_B}]^z}$ (*teste do trapdoor BDH – variante 1, pág. 42*)
e se $M^z N \stackrel{?}{=} e(aP, x_A P)^{y_B} \cdot e(aP, x_B P)^{y_A}$ (*teste do trapdoor BDH – variante 2, pág. 43*)
caso passe em todos os testes, $answerBDH \leftarrow W_1$
$\mathcal{S}$ verifica se já foi calculada chave para a sessão em questão ou uma *matching*
caso sim, atualiza entradas incompletas no registro de H^{lst} se for preciso e devolve a chave
caso contrário, sorteia SK, cria novo registro em H^{lst} e responde SK.
Se $\mathcal{A}$ consulta $H(A, C, E_A, E_C, Z, K, L, M, N)$ ou $H(C, B, E_C, E_B, Z, K, L, M, N)$
$\mathcal{S}$ realiza operações similares às do caso 1.

(b) Se $\mathcal{A}$ consulta StrongRevealSessionKey(A, B, s)
$\mathcal{S}$ calcula as variáveis de que é capaz e procura $(A, B, E_A, E_B, (*, *, Z_3, Z_4), *, *, *, *, *)$
se encontrar registros na forma $(A, B, E_A, E_B, \overline{Z}_1, \overline{Z}_2, Z_3, Z_4, \overline{K}, \overline{L}, \overline{M}, \overline{N}, \overline{SK})$,
verifica a consistência de $\overline{Z}_1$ e $\overline{Z}_2$ da mesma forma que no caso 5 (b) e
testa a consistência de $\overline{K}, \overline{L}, \overline{M}, \overline{N}$ efetuando os mesmos cálculos e testes do caso 9 (a)
caso passe em todos os testes, responde $\overline{SK}$
caso contrário, sorteia SK, cria novo registro com valores nulos para as variáveis que
$\mathcal{S}$ desconhece e responde SK.

Se $\mathcal{A}$ consulta StrongRevealSessionKey(A, C, s)
$\mathcal{S}$ realiza operações similares às do caso 5 (b).

Se $\mathcal{A}$ consulta StrongRevealSessionKey(C, B, s)
$\mathcal{S}$ realiza operações similares às do caso 6 (b).

Cálculos Usados no Caso 9

Para tratar o caso 9, usamos os seguintes cálculos, de forma a viabilizar a aplicação do teste do *trapdoor*:
$K' = \frac{K}{K_2 \cdot K_4} = e(r_B P, d_{A_1}) \cdot e(Q_{B_1}, d_{A_1}) = e(r_B P, abP) \cdot e(cP, abP)$
$L' = \frac{L}{L_2 \cdot L_4} = e(r_B P, d_{A_2}) \cdot e(Q_{B_2}, d_{A_2}) = e(r_B P, abP)^{-z} \cdot e(r_B P, y_A aP) \cdot e(Q_{B_2}, d_{A_2})$
$U_2 = [L']^{\frac{-1}{z}} = e(r_B P, abP) \cdot [e(r_B P, y_A aP) \cdot e(Q_{B_2}, d_{A_2})]^{\frac{-1}{z}}$

$$\begin{aligned}
W_1 &= U_1 \cdot \left[\frac{K'}{U_2}\right]^{\frac{1}{1+z}} \\
&= U_1 \cdot \left[e(cP, abP) \cdot [e(r_B P, y_A aP) \cdot e(Q_{B_2}, d_{A_2})]^{\frac{1}{z}}\right]^{\frac{1}{1+z}} \\
&= U_1 \cdot \left[e(cP, abP)^{(1+z)} \cdot [e(r_B P, y_A aP) \cdot e(y_A y_B P - y_A z cP - y_B z bP, aP)]^{\frac{1}{z}}\right]^{\frac{1}{1+z}} \\
&= U_1 \cdot \left[e(\frac{y_A}{z}(r_B P + y_B P) - y_A cP - y_B bP, aP)\right]^{\frac{1}{1+z}} \cdot e(cP, abP) \\
&= e(cP, abP)
\end{aligned}$$

$V_2 = [K']^{-z} = [e(r_B P, abP) \cdot e(Q_{B_1}, d_{A_1})]^{-z}$

$$\begin{aligned}
W_2 &= V_1 \cdot \left[\frac{L'}{V_2}\right]^{\frac{z}{1+z}} \\
&= V_1 \cdot \left[\frac{e(r_B P, y_A aP) \cdot e(Q_{B_2}, d_{A_2})}{e(Q_{B_1}, d_{A_1})^{-z}}\right]^{\frac{z}{1+z}} \\
&= V_1 \cdot \left[e(r_B P, y_A aP) \cdot e(Q_{B_2}, d_{A_2}) \cdot e(y_B P - Q_{B_2}, y_A aP - d_{A_2})^{1/z}\right]^{\frac{z}{1+z}} \\
&= V_1 \cdot \left[e(r_B P, y_A aP) \cdot [e(zcP, y_A aP) \cdot e(y_B P, zabP - y_A aP)]^{\frac{1}{z}}\right]^{\frac{z}{1+z}} \cdot e(Q_{B_2}, d_{A_2}) \\
&= V_1 \cdot \left[e(r_B P, y_A aP) \cdot [e(zcP, y_A aP) \cdot e(y_B bP, zaP) \cdot e(y_B P, -y_A aP)]^{\frac{1}{z}}\right]^{\frac{z}{1+z}} \cdot e(Q_{B_2}, d_{A_2}) \\
&= V_1 \cdot [e(z r_B P, y_A aP) \cdot e(zcP, y_A aP) \cdot e(y_B bP, zaP) \cdot e(y_B P, -y_A aP)]^{\frac{1}{1+z}} \cdot e(Q_{B_2}, d_{A_2}) \\
&= V_1 \cdot [e(z r_B P - Q_{B_2}, y_A aP) \cdot e(y_B bP, zaP)]^{\frac{1}{1+z}} \cdot e(Q_{B_2}, d_{A_2}) \\
&= e(Q_{B_2}, d_{A_2})
\end{aligned}$$

6.2 Segurança do GNT1 no Modelo Mal-LBG sob Gap-BDH

Para a segurança do protocolo GNT1, apresentamos uma redução do problema Diffie-Hellman Bilinear Lacunar (Gap-BDH) para o problema de se construir um algoritmo que diferencie um número aleatório de uma chave secreta calculada pelo protocolo. A redução indica que se houver um adversário com vantagem não negligenciável contra o protocolo, sob o modelo de adversário estendido da Seção 4.6.3, então existe algoritmo de tempo polinomial que resolve o Gap-BDH. A seguir, enunciamos o teorema sobre a segurança do protocolo e apresentamos sua demonstração sob

o modelo de oráculo aleatório (Bellare e Rogaway, 1993).

Teorema 6.2 *Sob a hipótese de dificuldade do problema Gap-BDH, se as funções H, H_1 e H_2 são modeladas como oráculos aleatórios, então GNT1 é um protocolo CL-AKA seguro no modelo Mal-LBG.*

6.2.1 Demonstração do Teorema 6.2

Suponha, por absurdo, que existe um algoritmo $\mathcal{A}$ de tempo polinomial probabilístico com vantagem não negligenciável em quebrar o protocolo. Vamos mostrar como construir um algoritmo $\mathcal{S}$ que recebe como entrada uma quádrupla (P, aP, bP, cP) referente a um desafio BDH e gera como resposta o valor $e(cP, abP)$ com a ajuda de um oráculo de decisão DBDH. $\mathcal{S}$ simula uma execução real do protocolo e interage com o adversário $\mathcal{A}$, nos moldes do jogo descrito no modelo de segurança de Lippold *et al.* (2009) com as ampliações descritas na Seção 4.6.3. Se o jogo não for abortado, $\mathcal{A}$ prossegue até o final e obtém vantagem contra o protocolo. O simulador usa os passos do adversário para calcular, então, a resposta ao desafio BDH, que é armazenada na variável *answer*.

Suponha que q_0 denota o número máximo de sessões nas quais um usuário pode participar e q_1 denota o máximo de consultas que $\mathcal{A}$ pode fazer a H_1, com q_0 e q_1 relacionados polinomialmente com o parâmetro de segurança k que gerou os parâmetros do sistema.

Antes do início do jogo entre o simulador $\mathcal{S}$ e o adversário $\mathcal{A}$, $\mathcal{S}$ tenta adivinhar qual será a sessão sobre a qual $\mathcal{A}$ realizará a consulta de Teste e quem serão os participantes dessa sessão. Considere um conjunto de usuários honestos previamente estabelecidos $\mathcal{U} = \{U_1, \ldots, U_n\}$, com $n \leq q_1$, e uma lista correspondente com valores (ID_i, x_i, x_iP). O simulador escolhe $I, J \in \{1, \ldots, n\}$, com $I \neq J$, e a sessão $\Pi^t_{I,J}$, onde $t \in \{1, \ldots, q_0\}$. A probabilidade de $\mathcal{S}$ fazer escolhas corretas é maior que $1/(q_0 q_1^2)$.

Se $\mathcal{S}$ fizer escolhas incorretas, será obrigado a abortar o jogo em algum momento. Por outro lado, se o adversário realizar alguma operação não permitida, o jogo também é abortado por $\mathcal{S}$. No entanto, se $\mathcal{A}$ apresenta vantagem não negligenciável, é porque realiza a consulta de Teste sobre uma sessão *fresh*, respeitando a definição de segurança. Em outras palavras, o adversário revela (ou modifica) no máximo dois dos três componentes secretos associados a cada participante da sessão de Teste.

Na Tabela 6.11, são listadas as possibilidades que o adversário possui para revelar ou modificar valores secretos associados à sessão de Teste e seus participantes, sem corromper integralmente a sessão. Os participantes da sessão de Teste são denotados por A e B, que equivalem respectivamente às identidades ID_I e ID_J. Sem perda de generalidade, considere que A é quem inicia a sessão e B é quem responde.

Os casos 1 a 4 representam aqueles em que o adversário pode ser a própria autoridade de geração de chaves que, na situação mais crítica, é um KCG que gera os parâmetros de forma desonesta. Para mostrar que o adversário não é capaz de tirar vantagem em escolher parâmetros de forma mal intencionada, $\mathcal{S}$ permite que $\mathcal{A}$ selecione arbitrariamente os parâmetros do sistema; $\mathcal{A}$ entrega para o simulador os parâmetros gerados junto com a chave mestra pública mpk e não revela a chave mestra secreta msk. Os casos 5 a 9 representam aqueles em que o adversário é externo. Nesses casos, $\mathcal{S}$ seleciona os parâmetros do sistema e define $mpk = aP$.

O simulador tenta adivinhar qual desses nove casos o adversário explorará para quebrar o protocolo. Se $\mathcal{S}$ errar na escolha, o jogo será abortado em algum momento. Se acertar, então a

Tabela 6.5: *Casos válidos de corrompimento dos participantes da sessão de Teste*

Adversário	1		2		3		4		5		6		7		8		9	
consulta	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B
RevealPartial *ou* KGC mal intenc.	e	e	e	e	e	e	e	e		r	r			r	r			
RevealSV *ou* ReplacePK			r e	r e		r e	r e		r e			r e	r e	r e	r e	r e	r e	r e
RevealEph *ou* adversário é ativo	r r	r e			r r			r e	r r	r e	r r	r e	r r			r e	r r	r e

Adversário pode revelar(r) ou escolher(e)/modificar valor do componente secreto

probabilidade de $\mathcal{S}$ completar o jogo será maior que $1/(9q_0q_1^2)$. $\mathcal{S}$ ainda estabelece que $x_AP = aP$ (casos 1 e 3), $x_BP = bP$ (casos 1 e 4), $x_AP = cP$ (caso 6) e $x_BP = cP$ (caso 5); quando for o caso, $\mathcal{S}$ faz $x_A = \perp$ ou $x_B = \perp$. $\mathcal{S}$ inicia, então, a simulação.

Respostas aos Oráculos

Uma vez iniciado o jogo, o simulador deve responder às consultas realizadas pelo adversário aos oráculos disponíveis. O comportamento do simulador para tratar essas consultas varia de acordo com cada um dos nove casos. Os oráculos H e RevealSessionKey, que respectivamente calcula e revela a chave de sessão, são os mais críticos a serem tratados pelo simulador. Por esse motivo, eles serão descritos mais detalhadamente na Seção 6.2.2. Os demais oráculos são tratados como segue:

H$_1(ID_i)$. Se $\mathcal{S}$ está simulando os casos 5 a 9, $\mathcal{S}$ embute convenientemente as entradas do desafio BDH:

- casos 5, 7 e 9: $H_1(A) = bP$, ou seja, Q_{A_1} é definido como bP
- casos 6 e 8: $H_1(B) = bP$, ou seja, Q_{B_1} é definido como bP
- caso 9: $H_1(B) = cP$, ou seja, Q_{B_1} é definido como cP

Para os demais participantes nos casos 5 a 9 e para os casos 1 a 4, $\mathcal{S}$ escolhe $l_i \in \mathbb{Z}_q$ ao acaso e responde l_iP. Todas as respostas e os valores l_i são armazenados em uma lista.

H$_2(ID_i)$. Análogo a H_1, porém $\mathcal{S}$ sorteia $z, y \in \mathbb{Z}_q$ (ou z, y_A, y_B, para o caso 9) e define:

- casos 5 e 7: $H_2(A) = yP - zbP$, ou seja, Q_{A_2} é definido como $yP - zQ_{A_1}$
- casos 6 e 8: $H_2(B) = yP - zbP$, ou seja, Q_{B_2} é definido como $yP - zQ_{B_2}$
- caso 9: $H_2(A) = y_AP - zbP$, isto é, $Q_{A_2} = y_AP - zQ_{A_1}$ e $H_2(B) = y_BP - zcP$, ou seja, $Q_{B_2} = y_BP - zQ_{B_1}$

Para os demais participantes nos casos 5 a 9 e para os casos 1 a 4, $\mathcal{S}$ escolhe $p_i \in \mathbb{Z}_q$ ao acaso e responde p_iP. Todas as respostas e os valores p_i são armazenados em uma lista.

RequestPublicKey(ID_i). $\mathcal{S}$ responde x_iP.

EstablishParty(ID_i, x_iP). O simulador cria o usuário desonesto com identificador (ID_i), caso ainda não exista, chamando os oráculos H_1, H_2. $\mathcal{S}$ registra a chave pública x_iP, define $x_i = \perp$. Nos casos 5 a 9, entrega ao adversário a chave secreta parcial $(d_{i_1} = l_iaP, d_{i_2} = p_iaP)$; nos casos 1 a 4, $\mathcal{S}$ recebe a chave secreta parcial como entrada à consulta ao oráculo e a armazena.

Send$(\Pi^s_{i,j}, m)$. Se a sessão $\Pi^s_{i,j}$ ainda não existe, $\mathcal{S}$ cria uma sessão para *dono* ID_i e *parceiro* ID_j, com estado *indefinido* e com o papel de emissor (se $m = \lambda$) ou receptor (em caso contrário). Se $m = \lambda$ e $\Pi^s_{i,j}$ é sessão de Teste, então define $r_iP = aP$ (nos casos 2 e 4) ou $r_iP = cP$ (no caso 8). Se $m \neq \lambda$, $\Pi^s_{i,j}$ é sessão de Teste com papel de receptor, então define $r_iP = bP$ (nos casos 2 e 3) ou $r_iP = cP$ (no caso 7). Nos demais casos, $\mathcal{S}$ escolhe r_i ao acaso. $\mathcal{S}$ executa o protocolo, extraindo r_BP de m quando necessário, atualiza o estado da sessão e entrega r_iP ao adversário.

RevealPartialKey(ID_i). Se $\mathcal{S}$ está tratando os casos 1 a 4, aborta. Se $ID_i = A$ e $\mathcal{S}$ está tratando os casos 5, 7 ou 9, aborta. Se $ID_i = B$ e $\mathcal{S}$ está tratando os casos 6, 8 ou 9, aborta. Caso contrário, responde a chave secreta parcial $(d_{i_1} = l_iaP, d_{i_2} = p_iaP)$.

RevealSecretValue(ID_i). Se $x_i = \perp$, aborta. Se $ID_i = A$ e $\mathcal{S}$ está tratando os casos 1, 3 ou 6, aborta. Se $ID_i = B$ e $\mathcal{S}$ está tratando os casos 1, 4 ou 5, aborta. Caso contrário, responde o valor secreto x_i.

ReplacePublicKey(ID_i, x_iP). Se $ID_i = A$ e $\mathcal{S}$ está tratando os casos 1, 3 ou 6, aborta. Se $ID_i = B$ e $\mathcal{S}$ está tratando os casos 1, 4 ou 5, aborta. Caso contrário, substitui a chave pública por x_iP e define $x_i = \perp$.

RevealEphemeral$(\Pi^s_{i,j})$. Se $ID_i = A$ e $\mathcal{S}$ está tratando os casos 2, 4 ou 8, aborta. Se $ID_i = B$ e $\mathcal{S}$ está tratando os casos 2, 3 ou 7, aborta. Caso contrário, devolve o valor secreto r_i da sessão.

RevealSessionKey$(\Pi^s_{i,j})$. Se $(\Pi^s_{i,j})$ for a sessão de Teste, aborta. Caso contrário, prossegue com os passos descritos na Seção 6.2.2. Nos casos 1 a 4, $\mathcal{S}$ recebe a chave secreta parcial do participante ID_i como entrada para a consulta.

H$(ID_i, ID_j, E_i, E_j, K, L, M, Z)$. Prossegue com os passos descritos na Seção 6.2.2.

Test$(\Pi^s_{i,j})$. Se $\Pi^s_{i,j}$ não é a sessão de Teste, aborta. Caso contrário, $\mathcal{S}$ joga uma moeda $b \in \{0,1\}$. Se $b = 0$, devolve a chave de sessão; caso contrário sorteia e devolve um número aleatório $r \in \{0,1\}^k$ para o adversário.

Finalização do Jogo

Assim que $\mathcal{A}$ finaliza o jogo, $\mathcal{S}$ devolve *answer*. Se a probabilidade de sucesso de $\mathcal{A}$ é $Pr[\mathcal{A}]$, então a probabilidade de sucesso de $\mathcal{S}$ é $Pr[\mathcal{S}] \geq Pr[\mathcal{A}]/(9q_0q_1^2)$. Se $\mathcal{A}$ tem vantagem não negligenciável em diferenciar um número aleatório da chave de sessão é porque $\mathcal{A}$ consultou H com valores corretos de K, L, M e Z. Nesse caso, *answer* contém o valor $e(cP, abP)$, conforme cálculos apresentados na seção 6.2.2. Então $\mathcal{S}$ resolve o problema Gap-BDH em tempo polinomial em k, o que contradiz a hipótese de dificuldade do Gap-BDH. □

6.2.2 Oráculos H e RevealSessionKey

Quando o oráculo RevealSessionKey é consultado pelo adversário, $\mathcal{S}$ deve informar o valor da chave de uma dada sessão $\Pi^s_{i,j}$, caso ela não seja a sessão de Teste. Entretanto, nas sessões que não são a de Teste e que envolvem os participantes A e/ou B do Teste, o simulador não é capaz de calcular por si só o valor da chave de sessão, pois desconhece alguns valores secretos. Se $\mathcal{S}$

informar valores diferentes de chave de sessão para duas sessões com *matching*, o adversário detecta a incoerência e aborta o jogo. Se isso ocorre, $\mathcal{S}$ é incapaz de aproveitar a vantagem de $\mathcal{A}$ para resolver Gap-BDH. Por esse motivo, passamos a descrever como $\mathcal{S}$ procede de modo a ser sempre consistente.

O simulador mantém uma lista H^{lst}, inicialmente vazia, com entradas na forma $(ID_i, ID_j, E_i, E_j, K, L, M, Z, SK)$. Como H é função, $\mathcal{S}$ deve responder o mesmo valor SK de chave de sessão para as mesmas entradas. Se $\mathcal{A}$ consultar H antes de eventualmente solicitar RevealSessionKey, basta $\mathcal{S}$ calcular o valor da chave de sessão SK e atualizar a lista. Se RevealSessionKey é consultado antes e $\mathcal{S}$ não é capaz de calcular todos os valores K, L, M, Z, então $\mathcal{S}$ sorteia SK e insere um novo registro na lista H^{lst} com os valores $(ID_i, ID_j, E_i, E_j, \ldots, SK)$, preenchendo tanto quanto possível os valores corretos de K, L, M, Z. Em uma eventual consulta posterior a H referente a uma sessão com *matching*, $\mathcal{S}$ atualiza H^{lst} com os dados faltantes e responde o mesmo valor SK. A seguir, vamos dividir a análise em dois grandes casos, onde H e RevealSessionKey são consultados sobre sessões que:

(a) não envolvem os participantes A e B da sessão de Teste e

(b) envolvem o participante A e/ou B da sessão de Teste

(a) Sessões que Não Envolvem A e B Alvos do Teste

Considere que $\mathcal{A}$ consulta H ou RevealSessionKey sobre participantes ID_i e ID_j, ambos diferentes de A, B. Sem perda de generalidade, suponha que ID_i é quem inicia a sessão. O simulador conhece as chaves secretas de ID_i e de ID_j, a não ser nos seguintes casos:

- $\mathcal{A}$ substitui a chave pública de ID_i e, portanto, $\mathcal{S}$ não conhece x_i
- $\mathcal{A}$ substitui a chave pública de ID_j e, portanto, $\mathcal{S}$ não conhece x_j
- $\mathcal{A}$ escolhe ativamente o valor temporário de ID_j e, portanto, $\mathcal{S}$ não conhece r_j

Podemos supor que $\mathcal{S}$ conhece r_i porque, se $\mathcal{A}$ ativamente alterar r_iP e entregar outro valor a ID_j, não haverá sessão com *matching* (a não ser com probabilidade negligenciável). No pior dos casos, $\mathcal{A}$ consulta RevealSessionKey antes de consultar H e $\mathcal{S}$ é incapaz de calcular os valores Z_1, Z_2. Em uma eventual consulta posterior de $\mathcal{A}$ a H, $\mathcal{S}$ verifica se $e(x_iP, x_jP) \stackrel{?}{=} e(\overline{x_ix_j}P, P)$ e se $e(x_iP, r_jP) \stackrel{?}{=} e(\overline{x_ir_j}P, P)$, onde $\overline{x_ix_j}P$ e $\overline{x_ir_j}P$ são informados por $\mathcal{A}$. Se ambas igualdades valem e demais entradas de H correspondem aos valores que $\mathcal{S}$ consegue calcular, então $\mathcal{S}$ responde a mesma chave SK, pois se trata de sessão com *matching*, caso contrário, sorteia nova chave. $\mathcal{S}$ atualiza H^{lst} conforme necessário.

(b) Sessões que Envolvem A ou B Alvos do Teste

O simulador embute os valores do desafio em pontos estratégicos do protocolo, de modo a induzir o adversário a realizar cálculos com esses valores. Na Tabela 6.6, são relacionadas as possíveis estratégias que $\mathcal{A}$ pode empregar para quebrar a segurança do protocolo e que chaves são associadas aos valores do desafio. A última linha da Tabela 6.6 indica as variáveis que ocorrem no protocolo e que capturam o cálculo de $e(cP, abP)$, ou seja, a resposta do desafio. Obviamente $\mathcal{S}$ não sabe calcular essa resposta, mas mostraremos que se $\mathcal{A}$ apresenta vantagem não negligenciável, é porque

Tabela 6.6: *Casos válidos para o adversário e inserção do desafio BDH*

	1		2		3		4		5		6		7		8		9	
Chave	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B	A	B
ID: Q_1	x	x	x	x	x	x	x	x	bP			bP	bP			bP	bP	cP
PK: xP	aP	bP	x	x	aP	x	x	bP	x	cP	cP	x	x	x	x	x	x	x
Eph: rP		x	aP	bP		bP	aP	x		x		x		cP	cP	x		x
Desafio em:	Z_1		Z_3		Z_2		Z_4		$mpk = aP$									
									M_1		M_2		K_1		K_2		K_3	

(aP, bP, cP)–desafio BDH (x)–simulador desconhece componente secreto

Tabela 6.7: *Nomenclatura das variáveis*

$$K = \underbrace{e(r_B P, d_{A_1})}_{K_1} \cdot \underbrace{e(Q_{B_1}, r_A s P)}_{K_2} \cdot \underbrace{e(Q_{B_1}, d_{A_1})}_{K_3} \cdot \underbrace{e(r_B P, r_A s P)}_{K_4}$$

$$L = \underbrace{e(r_B P, d_{A_2})}_{L_1} \cdot \underbrace{e(Q_{B_2}, r_A s P)}_{L_2} \cdot \underbrace{e(Q_{B_2}, d_{A_2})}_{L_3} \cdot \underbrace{e(r_B P, r_A s P)}_{L_4 (= K_4)}$$

$$M = \underbrace{e(x_B P, d_{A_1})}_{M_1} \cdot \underbrace{e(Q_{B_1}, x_A s P)}_{M_2} \quad Z = (\underbrace{x_A x_B P}_{Z_1}, \underbrace{x_A r_B P}_{Z_2}, \underbrace{r_A r_B P}_{Z_3}, \underbrace{r_A x_B P}_{Z_4})$$

conhece elementos suficientes para que a solução seja calculada. Tais elementos são entregues ao simulador sempre que $\mathcal{A}$ realiza consultas ao oráculo H.

Observe que as variáveis K e L do protocolo podem ser reescritas como indicado na Tabela 6.7. O fatores de M e os componentes de Z também são nomeados para facilitar a leitura dos cálculos. Quando $\mathcal{S}$ embute uma das entradas do desafio BDH em uma chave, automaticamente torna-se desconhecido o respectivo valor secreto. Por exemplo, quando aP é atribuído como valor de chave pública de A, isto é, $x_A P = aP$, $\mathcal{S}$ desconhece x_A por desconhecer a. Nos casos 5 a 9, a chave pública mestra recebe o valor aP e, por isso, $\mathcal{S}$ não tem acesso ao valor da chave mestra secreta. Quando $Q_{A_1} = bP$, nos casos 5, 7 e 9, $\mathcal{S}$ desconhece a chave parcial secreta $d_{A_1} = abP$, pois não sabe calcular ab. Na Tabela 6.6, são indicados com "x" outros componentes secretos aos quais $\mathcal{S}$ não tem acesso. São os casos em que $\mathcal{A}$ é permitido substituir a chave pública ou escolher ativamente o valor temporário de sessão, preservando ainda a característica de sessão *fresh* e com *matching*.

De acordo com a definição de sessão *fresh*, se não houver sessão com *matching*, o receptor não pode ser totalmente corrompido. $\mathcal{S}$ precisa tratar corretamente as situações em que B se envolve em sessões integralmente corrompidas, isto é, com um participante C desonesto. Esse cenário é tratado como subcaso dos casos 1, 4, 5, 6, 8 e 9. Analogamente, $\mathcal{S}$ deve lidar corretamente com as sessões em que A interage com o participante C desonesto; essa situação é tratada como subcaso de todos os nove casos.

Na sequência, vamos analisar os nove casos sobre sessões que envolvem A e/ou B, participantes do Teste. Nos casos 5 a 9, fazemos uso do oráculo de decisão BDH, suposto existente no problema Gap-BDH. No caso 9, usamos duas variantes do problema Gap-BDH, que mostramos serem equivalentes ao Gap-BDH, na Seção 3.3.2, pág. 44.

Caso 1. O desafio é embutido em Z_1. Se $\mathcal{A}$ consulta $H(A, B, E_A, E_B, K, L, M, (Z_1, Z_2, Z_3, Z_4))$, $\mathcal{S}$ verifica se $e(aP, bP) \stackrel{?}{=} e(P, Z_1)$, caso sim, $answerBDH \leftarrow e(cP, Z_1)$. $\mathcal{S}$ procede como no caso (a) para manter H^{lst} atualizada.

Casos 2, 3 e 4. O desafio é embutido respectivamente em Z_3, Z_2 e Z_4. $\mathcal{S}$ procede de forma semelhante ao caso 1.

Caso 5. O desafio é embutido em M_1.

Se $\mathcal{A}$ consulta $H(A, B, E_A, E_B, K, L, M, Z)$, $\mathcal{S}$ verifica se M_1 contém a resposta ao desafio, calculando $M_2 = e(d_{B_1}, x_A P)$, $M_1 = M/M_2$ e submetendo $\langle aP, bP, cP, M_1\rangle$ ao oráculo DBDH; se o oráculo responder positivamente, $answerBDH \leftarrow M_1$. $\mathcal{S}$ verifica se já foi calculada chave para a sessão em questão ou uma com *matching*; em caso positivo, atualiza entradas incompletas no registro de H^{lst} se for preciso, caso contrário, sorteia SK e cria novo registro em H^{lst}. Responde SK.

Se $\mathcal{A}$ consulta RevealSessionKey(A, B, s), $\mathcal{S}$ calcula as variáveis de que é capaz e procura $(A, B, E_A, E_B, *, *, *, (*, *, Z_3, Z_4), *)$ em H^{lst}; se encontrar registros na forma $(A, B, E_A, E_B, \overline{K}, \overline{L}, \overline{M}, (\overline{Z}_1, \overline{Z}_2, Z_3, Z_4), \overline{SK})$, calcula $K_1 = \frac{\overline{K}}{K_2 \cdot K_3 \cdot K_4}$ $M_1 = \frac{\overline{M}}{M_2}$ e fornece $\langle aP, bP, r_B P, K_1\rangle$ e $\langle aP, bP, cP, M_1\rangle$ ao oráculo DBDH; se o oráculo responder positivamente em ambas consultas, $\mathcal{S}$ calcula $L_1 = \frac{\overline{L}}{L_2 \cdot L_3 \cdot L_4}$ e verifica se valem todas as seguintes igualdades: se $L_1 \stackrel{?}{=} e(r_B P, yaP) \cdot K_1^{-z}$, se $e(x_A P, x_B P) \stackrel{?}{=} e(\overline{Z}_1, P)$ e se $e(x_A P, r_B P) \stackrel{?}{=} e(\overline{Z}_2, P)$; em caso positivo, $\mathcal{S}$ responde $\overline{SK}$, caso contrário, sorteia novo SK.

Se $\mathcal{A}$ consulta RevealSessionKey(A, C, s), $\mathcal{S}$ procede de forma análoga e captura consistência fornecendo $\langle aP, r_c P, yP, K_1\rangle$ e $\langle aP, x_c P, yP, M_1\rangle$ ao oráculo DBDH.

Se $\mathcal{A}$ consulta RevealSessionKey(C, B, s), $\mathcal{S}$ testa a consistência dos componentes de $\overline{Z}$ e testa se $\overline{K} \stackrel{?}{=} K_1 \cdot K_2 \cdot K_3 \cdot e(\overline{Z}_3, aP)$ e se $\overline{L} \stackrel{?}{=} L_1 \cdot L_2 \cdot L_3 \cdot e(\overline{Z}_3, aP)$; se todas igualdades valem, responde $\overline{SK}$, caso contrário, sorteia novo SK.

Casos 6, 7 e 8. O desafio é embutido respectivamente em M_2, K_1 e K_2. $\mathcal{S}$ procede de forma semelhante ao caso 5.

Caso 9. Similar ao caso 5, mas o desafio é embutido em K_3. Se $\mathcal{A}$ consulta H, $\mathcal{S}$ fornece

$\langle aP, bP, cP, r_A P, r_B P, K\rangle$ ao oráculo de decisão-BDH$^+$ (ver Seção 3.3.2, pág. 44); se o oráculo responder positivamente, $\mathcal{S}$ calcula $K' = \frac{K}{K_2 \cdot K_4}$ $L' = \frac{L}{L_2 \cdot L_4}$ $U_1 = \left[e(\frac{y_A}{z}(r_B P + y_B P) - y_A cP - y_B bP, aP)\right]^{\frac{-1}{1+z}}$ $U_2 = [L']^{\frac{-1}{z}}$ $K_3 = U_1 \cdot \left[\frac{K'}{U_2}\right]^{\frac{1}{1+z}}$ e $answerBDH \leftarrow K_3$.

Se $\mathcal{A}$ consulta RevealSessionKey(A, B, s), $\mathcal{S}$ testa K como acima e fornece $\langle aP, bP, cP, x_A P, x_B P, M\rangle$ ao oráculo de decisão-BDH*; se o oráculo responder positivamente e K também passar no teste, $\mathcal{S}$ responde $\overline{SK}$, caso contrário, sorteia novo SK. Se $\mathcal{A}$ consulta RevealSessionKey para (A, C, s) ou (C, B, s), $\mathcal{S}$ procede de forma análoga ao caso 5.

6.3 Segurança do GOT1 no Modelo LBG sob BDH

Uma vez que o protocolo GNT3 foi construído para ser uma otimização do GOT1, pela eliminação da dependência das variáveis N_1 e N_2 na prova de segurança, é possível reaplicar a mesma prova do Teorema 6.1 para a demonstração do teorema abaixo.

Teorema 6.3 *Sob a hipótese de dificuldade do problema BDH, se as funções H, H_1 e H_2 são modeladas como oráculos aleatórios, então GOT1 é um protocolo CL-AKA seguro no modelo LBG.*

6.4 Segurança do GNT2 no Modelo Mal-SJ$^+$ sob Gap-BDH

Para mostrar a segurança do protocolo GNT2, apresentamos uma redução do problema Diffie-Hellman Bilinear Lacunar (Gap-BDH) para o problema de se construir um algoritmo que diferencie um número aleatório de uma chave secreta calculada pelo protocolo. A redução indica que se houver um adversário com vantagem não negligenciável contra o protocolo, sob o modelo de adversário estendido da Seção 4.6.4 (pág. 60), então existe algoritmo de tempo polinomial que resolve o Gap-BDH, no modelo de oráculos aleatórios. Dividimos a prova em dois casos: o da autoridade mal intencionada e o caso geral (de adversário externo e interno que corrompeu a chave mestra). Seguem o enunciado do teorema e respectiva prova.

Teorema 6.4 *Sob a hipótese de dificuldade do problema Gap-BDH, se as funções H, H_1 e H_2 são modeladas como oráculos aleatórios, então GNT2 é um protocolo CL-AKA seguro no modelo Mal-SJ$^+$.*

6.4.1 Demonstração do Teorema 6.4

Suponha, por absurdo, que existe um algoritmo $\mathcal{A}$ de tempo polinomial probabilístico com vantagem não negligenciável em quebrar o protocolo. Vamos mostrar como construir um algoritmo $\mathcal{S}$ que recebe como entrada uma quádrupla (P, aP, bP, cP) referente a um desafio BDH e gera como resposta o valor $e(cP, abP)$ com a ajuda de um oráculo de decisão DBDH. $\mathcal{S}$ simula uma execução real do protocolo e interage com o adversário $\mathcal{A}$, nos moldes do jogo descrito no modelo de segurança de Swanson e Jao (2009) com as melhorias descritas nas seções 4.6.1 e 4.6.4. Se o jogo não for abortado, $\mathcal{A}$ prossegue até o final e obtém vantagem contra o protocolo. O simulador usa os passos do adversário para calcular, então, a resposta ao desafio BDH, que é armazenada na variável *answer*.

Suponha que q_0 denota o número máximo de sessões nas quais um usuário pode participar e q_1 denota o máximo de consultas que $\mathcal{A}$ pode fazer a H_1, com q_0 e q_1 relacionados polinomialmente com o parâmetro de segurança k que gerou os parâmetros do sistema.

Fase de Preparo

Antes do início do jogo entre o simulador $\mathcal{S}$ e o adversário $\mathcal{A}$, $\mathcal{S}$ tenta adivinhar qual será a sessão sobre a qual $\mathcal{A}$ realizará a consulta de Teste e quem serão os participantes dessa sessão. Considere um conjunto de usuários honestos previamente estabelecidos $\mathcal{U} = \{U_1, \ldots, U_n\}$, com $n \leq q_1$, e uma lista correspondente com valores (ID_i, x_i, x_iP). O simulador escolhe $I, J \in \{1, \ldots, n\}$, com $I \neq J$, e a sessão $\Pi^t_{I,J}$, onde $t \in \{1, \ldots, q_0\}$. A probabilidade de $\mathcal{S}$ fazer escolhas corretas é maior que $1/(q_0 q_1^2)$.

Considere que $\mathcal{A} = (\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2)$ e o atacante executa $\mathcal{M}_0$ com a entrada 1^k. $\mathcal{M}_0$ gera como saída a chave mestra pública mpk e demais parâmetros que são entregues a $\mathcal{S}$. $\mathcal{A}$ conhece as chaves parciais secretas de todos usuários e pode corromper no máximo mais um componente secreto (o valor secreto gerado pelo usuário e o segredo temporário de sessão) de cada um dos participantes da sessão de Teste para obter vantagem. Na Tabela 6.8, listamos quatro possibilidades para $\mathcal{A}$ revelar ou modificar chaves relacionadas com a sessão de Teste ou seus participantes (denotados por A e B), preservando ainda a característica de sessão *fresh*. Sem perda de generalidade, considere que A

Tabela 6.8: *Casos válidos para uma autoridade mal intencionada corromper os participantes da sessão de Teste*

	1		2		3		4	
Consultas ou ações válidas	A	B	A	B	A	B	A	B
RevealSecretValue *ou* ReplacePublicKey			r/m	r/m		r/m	r/m	
RevealEphemeral *or* KGC é ativo	r/m	r/m			r/m			r/m
KGC é ativo, mas a sessão de Teste possui *matching*	r	m			r			m
KGC calcula todas as chaves secretas parciais	m	m	m	m	m	m	m	m

(r)–revela um segredo (m)–modifica/escolhe um segredo

é quem inicia a sessão e B é quem responde e que A equivale a ID_I e B a ID_J. Quando a sessão de Teste tem uma sessão com *matching*, sem perda de generalidade considere que $\mathcal{A}$ não modifica o temporário secreto de A. $\mathcal{S}$ trata o caso em que não há sessão com *matching* como um subcaso em que B não é totalmente corrompido, mas se envolve com um participante integralmente corrompido C.

O simulador tenta adivinhar qual desses quatro casos o adversário explorará para quebrar o protocolo. Se $\mathcal{S}$ errar na escolha, o jogo será abortado em algum momento. Se acertar, então a probabilidade de $\mathcal{S}$ completar o jogo será maior que $1/(4q_0q_1^2)$.

Respostas aos Oráculos

O adversário, por meio de $\mathcal{M}_1$ e $\mathcal{M}_2$, interage com $\mathcal{S}$ consultando os oráculos descritos abaixo. Para ser consistente com suas respostas, $\mathcal{S}$ as armazena em listas.

RequestPublicKey(ID_i). $\mathcal{S}$ mantém uma lista com valores $(ID_i, x_i, x_iP, \hat{x}_iP)$, onde $\hat{x}_iP = \perp$ no começo do jogo. Nos casos 1 e 3, $\mathcal{S}$ define $x_AP = aP$ e $x_A = \perp$; nos casos 1 e 4, $x_BP = bP$ e $x_B = \perp$. Para definir os pares dos demais participantes, $\mathcal{S}$ escolhe ao acaso x_i e calcula x_iP. $\mathcal{S}$ responde x_iP.

Send$(\Pi^s_{i,j}, m)$. Se a sessão $\Pi^s_{i,j}$ ainda não existe, $\mathcal{S}$ cria uma sessão para *dono* ID_i e *parceiro* ID_j, com estado *indefinido* e com o papel de emissor (se $m = \lambda$) ou receptor (em caso contrário). Se $m = \lambda$ e $\Pi^s_{i,j}$ é sessão de Teste, então define $r_iP = aP$ (nos casos 2 e 4). Se $m \neq \lambda$, $\Pi^s_{i,j}$ é sessão de Teste com papel de receptor, então define $r_iP = bP$ (nos casos 2 e 3). Nos demais casos, $\mathcal{S}$ usa m para definir o temporário de sessão ou escolhe r_i ao acaso (se $m = \lambda$). $\mathcal{S}$ atualiza o estado da sessão (*aceito* ou *rejeitado*, de acordo com o protocolo) e entrega r_iP ao adversário quando necessário.

RevealEphemeral$(\Pi^s_{i,j})$. $\mathcal{S}$ aborta se $ID_i = A$ (nos casos 2 e 4), ou $ID_i = B$ (nos casos 2 e 3), ou se o estado da sessão for *indefinido*. Caso contrário devolve o valor temporário secreto r_i.

RevealSecretValue(ID_i). $\mathcal{S}$ aborta se $x_i = \perp$ ou $ID_i = A$ (nos casos 1 e 3) ou $ID_i = B$ (nos casos 1 e 4). Caso contrário devolve o valor secreto x_i.

ReplacePublicKey(ID_i, xP). $\mathcal{S}$ aborta se $ID_i = A$ (nos casos 1 e 3) ou $ID_i = B$ (nos casos 1 e 4). Caso contrário, registra a nova chave pública $\hat{x}_iP = xP$.

PPK-RevealSessionKey$(\Pi^s_{i,j}, d_i)$. Se $\Pi^s_{i,j}$ é a sessão de Teste ou se o estado da sessão não é *aceito*, $\mathcal{S}$ aborta. Caso contrário $\mathcal{S}$ segue os passos descritos na Seção 6.4.2. Para enfatizar

quando $\mathcal{A}$ consulta este oráculo sobre sessões envolvendo os participantes A e/ou B, usamos a notação simplificada como PPK-RevealSessionKey(A, B, t, d_A).

H$(ID_i, ID_j, E_i, E_j, Z, K, M, N)$. $\mathcal{S}$ prossegue com os passos descritos na Seção 6.4.2.

Test$(\Pi^s_{i,j})$. Se $\Pi^s_{i,j}$ não é a sessão de Teste, aborta. Caso contrário, $\mathcal{S}$ joga uma moeda $b \in \{0,1\}$. Se $b = 0$, devolve a chave de sessão; caso contrário sorteia e devolve um número aleatório $r \in \{0,1\}^k$ para o adversário.

Finalização do Jogo

Assim que $\mathcal{A}$ finaliza o jogo, $\mathcal{S}$ devolve *answer*. Se a probabilidade de sucesso de $\mathcal{A}$ é $Pr[\mathcal{A}]$, então a probabilidade de sucesso de $\mathcal{S}$ é $Pr[\mathcal{S}] \geq Pr[\mathcal{A}]/(4q_0q_1^2)$. Se $\mathcal{A}$ tem vantagem não negligenciável em diferenciar um número aleatório da chave de sessão é porque $\mathcal{A}$ consultou H com valores corretos de Z. Nesse caso, *answer* contém o valor $e(cP, abP)$, conforme cálculos apresentados na seção 6.4.2. Então $\mathcal{S}$ resolve o problema Gap-BDH em tempo polinomial em k, o que contradiz a hipótese de dificuldade do Gap-BDH. □

6.4.2 Oráculos H e PPK-RevealSessionKey

Quando $\mathcal{A}$ consulta PPK-RevealSessionKey sobre uma sessão que não é a de Teste, $\mathcal{S}$ deve fornecer respostas consistentes sempre que houver uma sessão com *matching*. O simulador mantém uma lista H^{lst}, inicialmente vazia, com entradas na forma $(ID_i, ID_j, E_i, E_j, Z, K, M, N, SK)$ que é atualizada sempre que $\mathcal{A}$ consulta H ou PPK-RevealSessionKey. Há dois grandes casos a serem tratados por $\mathcal{S}$:

1) Sessões que não envolvem os participantes A e B da sessão de Teste

Suponhas que $\mathcal{A}$ consulta H ou RevealSessionKey sobre os participantes ID_i e ID_j, ambos diferentes dos participantes da sessão de Teste e, sem perda de generalidade, considere que ID_i é quem inicia. $\mathcal{S}$ pode não conhecer $\hat{x}_i$ ou $\hat{x}_j$ (se $\mathcal{A}$ tiver substituído a chave pública de ID_i ou de ID_j) nem r_j (se $\mathcal{A}$ tiver escolhido o valor temporário para ID_j).

Podemos supor que $\mathcal{S}$ conhece r_i, se houver sessão com *matching*, e x_i é usado em todos os cálculos mesmo que $\hat{x}_iP \neq \perp$. Assim, $\mathcal{S}$ pode calcular Z. E uma vez que $\mathcal{A}$ entrega d_i a $\mathcal{S}$ quando consulta o oráculo PPK-RevealSessionKey, $\mathcal{S}$ sempre pode calcular K, M, N e é capaz de responder consistentemente.

2) Sessões que envolvem os participantes A ou B da sessão de Teste

O simulador embute os valores do desafio em pontos estratégicos do protocolo, de modo a induzir o adversário a realizar cálculos com esses valores. Na Tabela 6.9, são relacionadas as possíveis estratégias que $\mathcal{A}$ pode empregar para quebrar a segurança do protocolo e que chaves são associadas aos valores do desafio. Nessa tabela, são indicados com "x" outros componentes secretos aos quais $\mathcal{S}$ não tem acesso. São os casos em que $\mathcal{A}$ é permitido substituir a chave pública ou escolher ativamente o valor temporário de sessão. Observe que as variáveis K e M do protocolo podem ser reescritas como indicado na Tabela 6.10. Os componentes de Z recebem os nomes dados na última linha da

Tabela 6.9: *Casos para a autoridade mal intencionada e onde o desafio BDH é embutido*

	1		2		3		4	
Componente de chave	A	B	A	B	A	B	A	B
Chave pública gerada pelo usuário: xP	aP	bP	x	x	aP	x	x	bP
Temporário de sessão: rP		x	aP	bP		bP	aP	x
aP, bP do desafio BDH embutido em:	$Z_1 = x_A x_B P$		$Z_3 = r_A r_B P$		$Z_2 = x_A r_B P$		$Z_4 = r_A x_B P$	

Tabela 6.10: *Nomes das variáveis*

$$K = \underbrace{e(r_B P + Q_B, r_A \cdot mpk)}_{K_1} \cdot \underbrace{e(r_B P + Q_B, d_A)}_{K_2}$$

$$M = \underbrace{e(x_B P + Q_B, x_A \cdot mpk)}_{M_1} \cdot \underbrace{e(x_B P + Q_B, d_A)}_{M_2}$$

Tabela 6.9, que são as variáveis que ajudam na captura do cálculo de $e(cP, abP)$, ou seja, a resposta do desafio. Vamos analisar cada caso:

Caso 1.

O desafio BDH é embutido em Z_1, isto é, as chaves públicas de A e B são definidas para serem respectivamente aP e bP. Se $\mathcal{A}$ consulta $H(A, B, E_A, E_B, Z, K, M, N)$, $\mathcal{S}$ verifica se $e(aP, bP) \stackrel{?}{=} e(Z_1, P)$; caso sim, define $answer \leftarrow e(cP, Z_1)$. Se a chave de sessão já estiver definida, então $\mathcal{S}$ a devolve para $\mathcal{A}$, caso contrário $\mathcal{S}$ escolhe uma nova e atualiza H^{lst}.

Se $\mathcal{A}$ consulta PPK-RevealSessionKey(A, B, t, d_A), $\mathcal{S}$ não sabe calcular Z_1, Z_2 e M. $\mathcal{S}$ pode detectar consistência ao calcular Z_3, Z_4, K, N, e buscando $(A, B, E_A, E_B, (*, *, Z_3, Z_4), K, *, N, *)$ em H^{lst}. Para registros na forma $(A, B, E_A, E_B, (\overline{Z}_1, \overline{Z}_2, Z_3, Z_4), K, \overline{M}, N, \overline{SK})$, $\mathcal{S}$ testa se $e(aP, bP) \stackrel{?}{=} e(\overline{Z}_1, P)$, e se $e(aP, r_B P) \stackrel{?}{=} e(\overline{Z}_2, P)$. Se ambos testes são satisfeitos, $\mathcal{S}$ calcula M_2 (como indicado na Tabela 6.10) e submete $\langle Q_B + bP, aP, mpk, \overline{M}/M_2 \rangle$ ao oráculo de decisão DBDH. Se o oráculo responder positivamente, então $\mathcal{S}$ devolve $\overline{SK}$, caso contrário $\mathcal{S}$ sorteia novo SK e atualiza H^{lst} conforme necessário.

Se $\mathcal{A}$ consulta PPK-RevealSessionKey sobre as entradas (A, C, t, d_A) ou (C, B, t, d_C), onde C é um participante totalmente corrompido, $\mathcal{S}$ pode não saber calcular K, M ou Z, mas $\mathcal{S}$ detecta consistência testando, por exemplo, se $e(x_A P, \hat{x}_C P) \stackrel{?}{=} e(\overline{Z}_1, P)$, para um $\overline{Z}_1 = x_A \hat{x}_C P$ armazenado em H^{lst}, e submetendo $\langle Q_B + r_B P, r_C P, mpk, \overline{K}/K_2 \rangle$ e $\langle Q_B + bP, \hat{x}_C P, mpk, \overline{M}/M_2 \rangle$ ao oráculo de decisão DBDH.

Caso 2.

É similar ao caso 1, mas o desafio BDH é embutido em Z_3 e $\mathcal{S}$ define $answer \leftarrow e(cP, Z_3)$ sempre que $e(aP, bP) = e(Z_3, P)$ para algum valor de Z_3 fornecido pelo adversário numa consulta a H.

Se $\mathcal{A}$ consulta PPK-RevealSessionKey (A, B, t, d_A), $\mathcal{S}$ calcula M, N e usa $\hat{x}_B P$, se $\hat{x}_B P \neq \perp$, ou usa o $x_B P$ original nos cálculos a seguir. Como A é o dono da sessão, $\mathcal{S}$ usa o valor secreto x_A original, mesmo se $\hat{x}_A P \neq \perp$ (isto é, a chave pública de A foi substituída).

Tabela 6.11: *Casos válidos para um adversário externo corromper os participantes da sessão de Teste*

	5		6		7		8		9	
Adversário consulta	A	B	A	B	A	B	A	B	A	B
RevealPartialKey		r	r			r	r			
RevealSecretValue *ou*	r			r	r	r	r	r	r	r
ReplacePublicKey	m			m	m	m	m	m	m	m
RevealEphemeral *ou* adversário é ativo	r	m	r	m	r			m	r	m

(r)–revela um segredo (m)–modifica/escolhe um segredo

Para registros na forma $(A, B, E_A, E_B, (\overline{Z}_1, \overline{Z}_2, \overline{Z}_3, \overline{Z}_4), \overline{K}, M, N, \overline{SK})$, $\mathcal{S}$ testa se $e(x_A P, \hat{x}_B P) \stackrel{?}{=} e(\overline{Z}_1, P)$, $e(x_A P, bP) \stackrel{?}{=} e(\overline{Z}_2, P)$, $e(aP, bP) \stackrel{?}{=} e(\overline{Z}_3, P)$, e se $e(aP, \hat{x}_B P) \stackrel{?}{=} e(\overline{Z}_4, P)$. Se todos os testes forem satisfeitos, $\mathcal{S}$ calcula K_2 (como na Tabela 6.10) e submete $\langle Q_B + bP, aP, mpk, \overline{K}/K_2 \rangle$ ao oráculo de decisão DBDH. Caso o oráculo responda positivamente, $\mathcal{S}$ devolve $\overline{SK}$, caso contrário $\mathcal{S}$ sorteia novo SK e atualiza H^{lst}.

Casos 3 e 4.

Esses casos são similares aos casos 1 e 2, mas o desafio BDH é embutido em Z_2 e Z_4, respectivamente.

6.4.3 Caso Geral

No caso geral, consideramos adversários externos, que não têm acesso à chave mestra secreta, e aqueles internos que corromperam o segredo principal do sistema. Esse segundo tipo pode ser tratado como um caso particular da autoridade mal intencionada e, na análise dos casos 1 a 4 da seção anterior, os procedimentos se repetem, mas são desnecessárias as consultas ao oráculo de decisão DBDH.

Os casos em que um adversário externo pode corromper os participantes da sessão de Teste são listados na Tabela 6.11 e numerados de 5 a 9. Antes de iniciar o jogo, o simulador escolhe os parâmetros do sistema e define a chave mestra pública como sendo $mpk = aP$. Para tratar os casos 5 a 9, as respostas aos oráculos são ligeiramente adaptadas como segue:

H$_1(ID_i)$ $\mathcal{S}$ define:

- nos casos 5, 7 e 9: $H_1(A) = bP$, isto é, $Q_A = bP$
- nos casos 6 e 8: $H_1(B) = bP$, isto é, $Q_B = bP$
- no caso 9: $H_1(B) = cP$, isto é, $Q_B = cP$

Para os demais participantes, nos casos 5 a 9, $\mathcal{S}$ sorteia $l_i \in \mathbb{Z}_q^*$ e devolve $l_i P$. Todas as respostas a H_1 e os valores l_i são armazenados em uma lista.

RequestPublicKey(ID_i). Nos casos 5 e 6, $x_B P = cP$ e $x_A P = cP$, respectivamente, e $x_A = x_B = \perp$.

Send$(\Pi^s_{i,j}, m)$ Se $m = \lambda$ e $\Pi^s_{i,j}$ é a sessão de Teste, então $\mathcal{S}$ define $r_i P = cP$ no caso 8; se além disso a sessão tiver papel de receptor, no caso 7 se define $r_i P = cP$.

Tabela 6.12: *Casos para um adversário externo e onde o desafio BDH é embutido*

	5		6		7		8		9	
Componente de chave	A	B	A	B	A	B	A	B	A	B
Identidade: Q_1	bP			bP	bP			bP	bP	cP
Chave pública: xP	x	cP	cP	x	x	x	x	x	x	x
Temporário: rP		x		x		cP	cP	x		x
Desafio BDH	$mpk = aP$									
embutido em:	M_1		M_2		K_1		K_2		N	

(aP, bP, cP)–desafio BDH (x)–$\mathcal{S}$ desconhece componente secreto

RevealPartialKey(ID_i) $\mathcal{S}$ aborta se $ID_i = A$ (nos casos 5, 7 ou 9), ou $ID_i = B$ (nos casos 6, 8 ou 9). Caso contrário devolve $(d_{i_1} = l_i aP, d_{i_2} = p_i aP)$.

RevealSecretValue(ID_i) $\mathcal{S}$ aborta se $x_i = \perp$ ou $ID_i = A$ (no caso 6) ou $ID_i = B$ (no caso 5). Caso contrário $\mathcal{S}$ devolve o valor secreto x_i.

ReplacePublicKey$(ID_i, x_i P)$ $\mathcal{S}$ aborta se $ID_i = A$ (no caso 6) ou $ID_i = B$ (no caso 5). Caso contrário $\mathcal{S}$ substitui a chave pública por $x_i P$ e define $x_i = \perp$.

RevealEphemeral$(\Pi^s_{i,j})$ $\mathcal{S}$ aborta se $ID_i = A$ (no caso 8) ou $ID_i = B$ (no caso 7) ou o estado da sessão é não *indefinido*. Caso contrário devolve o valor secreto temporário r_i.

RevealSessionKey$(\Pi^s_{i,j})$ Prossegue com os passos da Seção 6.4.4.

H$(ID_i, ID_j, E_i, E_j, Z, K, M)$ Prossegue com os passos da Seção 6.4.4.

Analogamente aos casos 1 a 4, $\mathcal{S}$ captura a resposta ao desafio na variável *answer*, com probabilidade de sucesso $Pr[\mathcal{S}] \geq Pr[\mathcal{A}]/(5q_0q_1^2)$.

6.4.4 H e PPK-RevealSessionKey quando A ou B é Envolvido

Analogamente aos casos 1 a 4, na Tabela 6.12 são listadas as variáveis onde o desafio BDH é embutido, nos casos 5 a 9. Vamos analisar cada um desses casos:

Caso 5.

O desafio BDH é embutido em M_1. Se $\mathcal{A}$ consulta $H(A, B, E_A, E_B, Z, K, M, N)$, $\mathcal{S}$ verifica se M_1 contém a resposta ao desafio calculando M_2, M_3, M_4 (com os valores que conhece), e $M_1 = \frac{M}{M_2 \cdot M_3 \cdot M_4}$. $\mathcal{S}$ submete $\langle aP, bP, cP, M_1 \rangle$ ao oráculo DBDH; se o oráculo responder positivamente, então $answer \leftarrow M_1$.

Se $\mathcal{A}$ consulta RevealSessionKey(A, B, t), $\mathcal{S}$ pode detectar consistência calculando Z e N e buscando $(A, B, E_A, E_B, Z, *, *, N, *)$ na H^{lst}. Com registros na forma $(A, B, E_A, E_B, Z, \overline{K}, \overline{M}, N, \overline{SK})$, $\mathcal{S}$ calcula $K_1 = \frac{\overline{K}}{K_2 \cdot K_3 \cdot K_4}$ e $M_1 = \frac{\overline{M}}{M_2 \cdot M_3 \cdot M_4}$, onde os valores K_i, M_i, $i \in \{2, 3, 4\}$, são computados por $\mathcal{S}$ com o valor x_A original. $\mathcal{S}$ submete $\langle r_B P, aP, bP, K_1 \rangle$ e $\langle aP, bP, cP, M_1 \rangle$ ao oráculo DBDH; se a resposta for positiva em ambos testes, então $\mathcal{S}$ devolve $\overline{SK}$. Caso contrário (nenhum registro passou nos testes) $\mathcal{S}$ sorteia SK e atualiza H^{lst}.

Se $\mathcal{A}$ consulta RevealSessionKey(A, C, t) or RevealSessionKey(C, B, t), $\mathcal{S}$ procede de forma análoga, já que conhece d_C.

Caso 6.

É similar ao caso 5, mas o desafio BDH é embutido em M_2. Se $\mathcal{A}$ consulta $H(A, B, E_A, E_B, Z, K, M, N)$, $\mathcal{S}$ verifica se M_2 contém a resposta ao desafio testanto se $e(cP, \hat{x}_B P) \stackrel{?}{=} e(Z_1, P)$, caso sim, $\mathcal{S}$ computa M_1, M_3 e $M_2 = \frac{M}{M_1 \cdot M_3 \cdot e(Z_1, aP)}$. Quando o oráculo DBDH responder positivamente às entradas $\langle aP, bP, cP, M_2 \rangle$, então $answer \leftarrow M_2$.

Se $\mathcal{A}$ consulta RevealSessionKey(A, B, t), $\mathcal{S}$ calcula K e N e, com os registros na forma $(A, B, E_A, E_B, \overline{Z}, K, \overline{M}, N, \overline{SK})$, $\mathcal{S}$ testa os componentes em $\overline{Z}$ e $\overline{M}$ para responder $\overline{SK}$ ou uma nova chave sorteada.

Casos 7 e 8.

São similares aos casos 5 e 6, mas o desafio BDH é embutido em K_1, K_2 respectivamente.

Caso 9.

É similar aos casos 5 e 6, mas o desafio BDH é embutido em N. Se $\mathcal{A}$ consulta $H(A, B, E_A, E_B, Z, K, M, N)$, $\mathcal{S}$ submete $\langle aP, bP, cP, N \rangle$ ao oráculo DBDH, e define $answer \leftarrow N$ sob uma resposta afirmativa. Se $\mathcal{A}$ consulta RevealSessionKey(A, B, t), $\mathcal{S}$ testa registros na forma $(A, B, E_A, E_B, \overline{Z}, \overline{K}, \overline{M}, \overline{N}, \overline{SK})$, $\mathcal{S}$ verifica os componentes em $\overline{Z}$ e submete $\langle r_B P + cP, aP, bP, \frac{\overline{K}}{K_2 \cdot K_4} \rangle$, $\langle \hat{x}_B P + cP, aP, bP, \frac{\overline{M}}{M_2 \cdot M_4} \rangle$ e $\langle aP, bP, cP, N \rangle$ ao oráculo DBDH. Se todas as consultas tiverem retorno afirmativo, então $\mathcal{S}$ devolve $\overline{SK}$, caso contrário $\mathcal{S}$ seleciona novo SK e atualiza H^{lst}.

6.5 Segurança do GNT4 no Modelo SJ$^+$ sob BDH

Para mostrar a segurança de GNT4 sob BDH e no modelo contra adversário moderado, mesclamos as técnicas empregadas para as provas dos teoremas relacionados à segurança de GNT2 e GNT3. Enunciamos o teorema para, em seguida, esboçar sua demonstração.

Teorema 6.5 *Sob a hipótese de dificuldade do problema BDH, se as funções H, H_1 e H_2 são modeladas como oráculos aleatórios, então GNT4 é um protocolo CL-AKA seguro no modelo SJ$^+$.*

6.5.1 Demonstração do Teorema 6.5

De forma análoga à demonstração do Teorema 6.1, a prova para o protocolo GNT4 se dá com base na observação que K, L, M, N, Z podem ser reescritos como

$$K = \underbrace{e(r_B P, d_{A_1})}_{K_1} \cdot \underbrace{e(Q_{B_1}, r_A s P)}_{K_2} \cdot \underbrace{e(Q_{B_1}, d_{A_1})}_{K_3} \cdot \underbrace{e(r_B P, r_A s P)}_{K_4}$$

$$L = \underbrace{e(r_B P, d_{A_2})}_{L_1} \cdot \underbrace{e(Q_{B_2}, r_A s P)}_{L_2} \cdot \underbrace{e(Q_{B_2}, d_{A_2})}_{L_3} \cdot \underbrace{e(r_B P, r_A s P)}_{L_4 (= K_4)}$$

$$M = \underbrace{e(x_B P, d_{A_1})}_{M_1} \cdot \underbrace{e(Q_{B_1}, x_A s P)}_{M_2} \cdot \underbrace{e(Q_{B_1}, d_{A_1})}_{M_3 (= K_3)} \cdot \underbrace{e(x_B P, x_A s P)}_{M_4}$$

$$N = \underbrace{e(x_B P, d_{A_2})}_{N_1} \cdot \underbrace{e(Q_{B_2}, x_A s P)}_{N_2} \cdot \underbrace{e(Q_{B_2}, d_{A_2})}_{N_3(=L_3)} \cdot \underbrace{e(x_B P, x_A s P)}_{N_4(=M_4)}$$

$$Z = (\underbrace{x_A x_B P}_{Z_1}, \underbrace{x_A r_B P}_{Z_2}, \underbrace{r_A r_B P}_{Z_3}, \underbrace{r_A x_B P}_{Z_4})$$

A principal diferença entre as provas deste e do Teorema 6.1 é com relação ao tratamento dos oráculos H, RevealSessionKey e ReplacePublicKey que acontece de maneira similar à prova do Teorema 6.4, onde o valor secreto associado à chave pública do dono de uma sessão sempre é conhecido do simulador. Para responder às consultas a H e RevealSessionKey, $\mathcal{S}$ usa o par original $(x_i, x_i P)$ do dono da sessão consultada, enquanto para os outros participantes é usado $\hat{x}_i P$ informado por $\mathcal{A}$ num evento de substituição de chave pública.

Detalharemos apenas o caso 9, uma vez que é o mais complexo de todos; os demais casos são similares ou mais simples que este. Se $\mathcal{A}$ consulta H sobre participantes da sessão de Teste (A, B), $\mathcal{S}$ verifica se o desafio BDH está em K_3, usando os mesmos cálculos presentes na prova do Teorema 6.1.

Se $\mathcal{A}$ consulta RevealSessionKey(A, B, s), $\mathcal{S}$ detecta consistência calculando Z e todas as variáveis de que for capaz. Para registros em H^{lst} na forma $(A, B, E_A, E_B, Z, \overline{K}, \overline{L}, \overline{M}, \overline{N}, \overline{SK})$, $\mathcal{S}$ computa W_1 e W_2, da mesma forma que na prova de GNT3 e realiza os mesmos testes de *trapdoor*: $\frac{W_2}{{W_1}^{z^2}} \stackrel{?}{=} \frac{e(aP,P)^{y_A y_B}}{[e(aP,cP)^{y_A} \cdot e(aP,bP)^{y_B}]^z}$. Então $\mathcal{S}$ calcula M_2, N_2 e $x_A aP$, com a chave pública original de A e $M_1 = \frac{\overline{M}}{M_2 \cdot W_1 \cdot e(\hat{x}_B P, x_A aP)}$ $N_1 = \frac{\overline{N}}{N_2 \cdot W_2 \cdot e(\hat{x}_B P, x_A aP)}$ onde $\hat{x}_B P$ é a chave pública substituída de B (se $\hat{x}_B P \neq \perp$ ou, em caso contrário, $\mathcal{S}$ usa a original). $\mathcal{S}$ verifica se $(M_1)^z \cdot N_1 \stackrel{?}{=} e(aP, \hat{x}_B P)^{y_A}$. Sob resposta positiva, $\mathcal{S}$ devolve $\overline{SK}$, caso contrário sorteia novo SK e atualiza H^{lst} conforme necessário.

6.6 Segurança do GOT2 no Modelo Fraco sob Gap-BDH

Diferentemente do que fizemos para os demais protocolos, para o GOT2 apenas descreveremos, nesta seção, as ideias gerais de como as propriedades de segurança são alcançadas, no modelo de adversário fraco, sob a hipótese de dificuldade do problema Gap-BDH. A justificativa para isso é que, quando de sua apresentação em Goya *et al.* (2010b), erroneamente o adversário era mais forte. Na necessidade de se enfraquecer o poder do adversário para mostrá-lo seguro, perde-se o interesse por este protocolo. Vamos, então, discutir os princípios de projeto dele, apenas para registro.

O modelo CK de segurança para protocolos de acordo de chaves modela as propriedades de segurança descritas na Seção 4.3, com exceção de KCI, KGC *Forward Secrecy* e resistência ao vazamento de segredos temporários para o KGC. Descreveremos, portanto, nos parágrafos a seguir, como o protocolo alcança essas últimas propriedades.

A propriedade KGC *Forward Secrecy* é alcançada por conta do segundo segredo temporário sorteado durante a sessão (y_A). Se o KGC não recuperar os valores secretos temporários y_A ou y_B, não será capaz de recuperar a chave de sessão acordada entre A e B, conhecendo apenas os dados públicos E_A e E_B (mais especificamente, Y_A e Y_B), pois em caso contrário estaria resolvendo o problema do Logaritmo Discreto. Formalmente, esse aspecto é demonstrado no Teorema 2 de Boyd *et al.* (2009).

Também em Boyd *et al.* (2009) é demonstrado que a conversão alcança a propriedade KCI, no caso de protocolos baseados em identidade. Para o caso sem certificado, introduzimos os valores M_1 e M_2, para que, mesmo que um adversário substitua chaves públicas por falsos valores à sua escolha, ou simplesmente corrompa os valores secretos x_A e x_B, não realize ataques de personificação do tipo KCI. Os valores M_1 e M_2 refletem as combinações possíveis dos valores secretos dos participantes, x_A e x_B, com os efêmeros t_A, t_B, y_A, y_B. Assim, se o adversário substituir uma chave pública ou corromper o x associado e obtiver vantagem sobre o criptossistema, há dois casos a considerar: (1) o adversário corrompeu os valores efêmeros e (2) corrompeu a chave secreta parcial (ambos os casos, simultaneamente com o comprometimento de x não é possível, pois nesse caso o adversário não alcançaria vantagem na sessão de teste e perderia o jogo). No caso (1), a vantagem do adversário é usada para se construir um algoritmo eficiente que resolve o problema Diffie-Hellman bilinear; em (2) de forma análoga se resolve o Diffie-Hellman computacional.

A resistência ao vazamento de segredos temporários para o KGC é obtida por causa das definições de M_1 e Z_{B_1}, que requerem o conhecimento do valor secreto x_A que gerou a chave pública, pois em caso contrário seria possível resolver o problema computacional Diffie-Hellman.

Os protocolos LBG2 e GOT1 foram demonstrados seguros segundo o modelo de Lippold *et al.* (2009), que é uma extensão do modelo CK, conhecida por eCK e definida em LaMacchia *et al.* (2007). Swanson Swanson (2008) e Lippold et al. estenderam eCK para incluir adaptações para o caso sem certificado. Os modelos CK e eCK diferem fundamentalmente no modo como são tratados os vazamentos dos segredos temporários. Enquanto em CK *todos* os segredos temporários de uma sessão podem ser revelados de uma única vez (por meio do oráculo *SessionStateReveal*), em eCK os temporários podem ser comprometidos um a um, à escolha do adversário.

Para formalizar a demonstração de segurança de GOT2, é preciso reescrever o modelo CK com as mesmas adaptações realizadas em Swanson (2008) e Lippold *et al.* (2009) e definir o oráculo *SessionStateReveal* para revelar simultaneamente os valores efêmeros t_U e y_U do participante U, para uma dada sessão.

6.6.1 Construção de Fiore, Gennaro e Smart

Fiore, Gennaro e Smart (Fiore *et al.*, 2010) desenvolveram uma construção genérica de mecanismos de encapsulamento de chave sem certificado (CL-KEM) a partir de protocolos de acordo de chave com autenticação baseados em identidade.

Em uma versão preliminar de Fiore *et al.* (2010), colocada no *Cryptology ePrint Archive*, os autores apresentaram dois exemplos de protocolos aplicando a conversão proposta por eles e afirmaram que ambos exemplos eram CCA-seguros. O primeiro exemplo foi criado a partir de SCK-2 (Chen *et al.*, 2007) e o segundo, a partir de FG (Fiore e Gennaro, 2010). No entanto, Yang e Tan (2011b) mostraram um ataque para o segundo exemplo de CL-KEM, construído a partir do protocolo FG; também apresentaram uma versão corrigida.

O protocolo GOT2 usa como ponto de partida o protocolo SCK-2, cuja variante segura, que chamaremos de SCK2-OWA, é proposta e detalhada na subseção a seguir.

Por fim, aplicando-se a conversão de Fiore, Gennaro e Smart ao SCK2-OWA, de acordo com os Teoremas 2, 3 e 4 de Fiore *et al.* (2010) obtemos um CL-KEM CCA-seguro que satisfaz as condições necessárias à aplicação da conversão seguinte, de Boyd *et al.* (2009).

6.6.2 Protocolo SCK2-OWA e Sua Segurança

O protocolo de acordo de chaves baseado em identidade com autenticação mútua SCK-2 foi originalmente proposto por Smart e melhorado por Chen e Kudla (Chen *et al.*, 2007). A variante SCK2-OWA garante autenticação apenas do participante que inicia a comunicação (*one-way authenticated*). Protocolos de acordo de chave com autenticação unilateral têm como exemplo clássico o SSL/TLS que, nas implementações mais corriqueiras, diferentemente dos clientes, o fornecedor de serviços é previamente autenticado por meio de certificado digital. Na conversão de Fiore *et al.* (2010), a autenticação unilateral é requerida para aumentar a eficiência do CL-KEM resultante.

O SCK2-OWA que propomos é composto pelas fases a seguir.

Inicialização do Sistema

A autoridade do sistema escolhe um emparelhamento bilinear admissível $e : G \times G \to G_T$ e um gerador P de G. Escolhe sua chave secreta mestra s; calcula sua chave mestra pública como $mpk = sP$ e seleciona as funções de hash $H_1 : \{0,1\}^* \to G$ e $H : G \times G_T \times \{0,1\}^* \times G \times G \to \{0,1\}^k$

Geração de Chaves de Usuário

Cada usuário com identidade ID possui o valor público $Q_{ID} = H_1(ID)$ e sua chave secreta é $d_{ID} = sQ_{ID}$.

Acordo de Chave

Os participantes A e B executam os passos da Tabela 6.13, sempre que desejarem compartilhar uma chave de sessão.

Para que a conversão de Fiore, Gennaro e Smart funcione, o protocolo de entrada deve ser seguro sob um modelo mais forte que o convencional, no qual existe um oráculo chamado *Reveal**, que permite que o adversário descubra o valor de uma sessão com base apenas nas mensagens trocadas entre os participantes. Ainda assim, é possível seguir os passos principais da demonstração dos Teoremas 1 e 2 de Chen *et al.* (2007) e fazer uso da vantagem do adversário para capturar a solução para uma instância de BDH em $e(caP, bP)$, inserida na variável y.

Tabela 6.13: *SCK2-OWA*

A		B
$t_A \xleftarrow{\$} \mathbb{Z}_q^*$		$t_B \xleftarrow{\$} \mathbb{Z}_q^*$
$T_A \leftarrow t_A P$	$\xrightarrow{T_A}$	$T_B \leftarrow t_B P$
	$\xleftarrow{T_B}$	
$x \leftarrow t_A T_B$		$x \leftarrow t_B T_A$
$y \leftarrow e(T_B, d_A)$		$y \leftarrow e(t_B \cdot mpk, Q_A)$
$SK \leftarrow H(x, y, ID_A, T_A, T_B)$		$SK \leftarrow H(x, y, ID_A, T_A, T_B)$

6.7 Correções na Segurança do LBG2 no Modelo LBG sob BDH

A prova de segurança apresentada originalmente para o protocolo LBG2 de Lippold *et al.* (2009) contém algumas falhas, mas essas não comprometem a segurança do protocolo. Sintetizamos aqui

as correções necessárias.

Na prova da estratégia 9, há erros na equação relacionada com o teste do *trapdoor* para a variável N. A equação corrigida é:

$$\frac{N_2}{{N_1}^{z^2}} \stackrel{?}{=} \frac{e(aP,P)^{y_1 y_2}}{[e(aP,cP)^{y_1} \cdot e(aP,bP)^{y_2}]^z}$$

No artigo de Lippold *et al.* (2009), também há erros na equação para o teste do *trapdoor* para a variável L. A equação correta é:

$${L_1}^z L_2 \stackrel{?}{=} e(aP,x_iP)^{y_2} \cdot e(aP,x_jP)^{y_1}$$

Nesse caso, o teorema definido pelos autores não é aplicável. Para a correção, fizemos uso de nossa variante 2 do teste do *trapdoor*, definida pelo Teorema 3.3.

6.8 Síntese e Conclusões Parciais

Neste capítulo, apresentamos as provas de segurança dos protocolos de acordo de chave GOT1, GNT1, GNT2, GNT3 e GNT4, no modelo do oráculo aleatório, além de um esboço para a prova de GOT2.

Nas provas de segurança para os casos contra autoridade mal intencionada, sempre necessitamos acessar um oráculo de decisão DBDH, incluindo como entrada valores de chave pública e/ou temporários de sessão, ambos sob controle do adversário que os escolheu ativamente. Isso sugere que, ao menos na formulação de chaves AP, em que a chave pública sem certificado pode ser usada antes da geração da chave secreta completa (conforme descrito na Seção 5.2, pág. 64), não é possível evitar o uso de problemas do tipo Gap.

Nesse cenário, o caso da autoridade que frauda na geração de parâmetros do sistema parece ser um divisor entre os protocolos que podem ser seguros no problema principal (BDH) e os que requerem a hipótese mais forte de Gap. Quando Okamoto e Pointcheval (2001) propuseram a família de problemas Gap, citaram exemplos de problemas que não haviam sido, até então, resolvidos sem Gap.

No capítulo a seguir, apresentamos as conclusões gerais desta tese e discutimos o que ainda pode ser realizado.

Capítulo 7

Conclusões

Neste capítulo, escrevemos as conclusões gerais deste tese, retomamos os resultados obtidos e relacionamos possíveis trabalhos futuros.

7.1 Considerações Finais

Nesta tese, apresentamos um estudo sobre diferentes paradigmas de criptografia de chave pública que dispensam a distribuição de certificados de chave pública, no sentido convencional sobre ICPs. Discorremos sobre os modelos de chave pública baseada em identidade, chave pública auto-certificada, chave pública sem certificado e baseada em certificado. Expusemos as propriedades de cada modelo, bem como suas vantagens e desvantagens.

Em seguida, passamos a explorar modelos de segurança no modelo sem certificado, para protocolos de acordo de chave com autenticação mútua entre dois participantes.

Modelos formais de segurança para acordo de chave com autenticação sem certificado foram aprimorados objetivando-se:

1. elevar o nível de confiança que usuários podem depositar na autoridade geradora de chaves secretas parciais e

2. viabilizar protocolos eficientes computacionalmente e com propriedades de segurança relevantes, como segurança no futuro e resistência a ataques de adversários que têm total controle do canal de comunicação e que podem substituir chaves públicas de usuários por valores arbitrários.

Mostramos que as melhorias efetuadas nos modelos de segurança são realizáveis, apresentando novos protocolos sob esses modelos, que foram mostrados seguros, sob a técnica de redução de problemas computacionais (segurança demonstrável).

Também efetuamos otimizações sobre protocolos previamente existentes e, por meio de testes de desempenho sobre codificações dos protocolos em linguagem de programação C, mostramos que eles podem ser adotados em aplicações reais.

A análise dos modelos de segurança nos permite arriscar concluir que a diferença entre eles recai essencialmente sobre quais aspectos do registro dos usuários e de suas chaves se quer assegurar.

Quanto mais fortes são os modelos, mais podemos relaxar na confiança a ser depositada na infraestrutura de gerenciamento de chaves (e na autoridade de confiança). Quanto mais fraco for o

modelo de adversário, mais nos aproximamos da prática comum em criptografia de chave pública: usar protocolos criptográficos mais simples que delegam complexidade para os procedimentos extra-criptográficos de gerenciamento da infraestrutura.

Dito dessa forma, pode parecer que se trata de uma constatação óbvia. No entanto, talvez não seja tão evidente. Pesquisadores que estudam segurança de esquemas CLE nem sempre chegam a um consenso sobre se é preferível que o adversário possua acesso a um oráculo StrongDecrypt, ou se basta o acesso a um oráculo mais fraco, que pode responder erroneamente se a chave pública tiver sido substituída. A interpretação do modelo abstrato de segurança nem sempre nos dá a dica de qual é o comportamento correspondente do adversário num cenário real.

É interessante observar que tanto os esquemas CLS de assinatura quanto o CLE garantem autenticação unilateral (respectivamente: só do remetente e só do destinatário). Os modelos que estudamos para CL-AKA preveem a autenticação mútua tanto da origem quanto do destino. Nesse caso, as implicações do poder do adversário sobre a infraestrutura de apoio transpareceram um pouco mais facilmente. Essa conclusão que arriscamos traçar vem corroborar um resultado não muito intuitivo obtido por Farshim e Warinschi (2009), que mostra a equivalência de IBE e CLE com *Certified Encryption* de Boldyreva *et al.* (2007). O resultado teórico de Farshim e Warinschi (2009) é que IBE e CLE podem ser interpretados como esquemas de cifragem sobre ICPs, em que é permitido ao adversário interferir no processo de registro das chaves.

Repensar os protocolos de chave pública sem certificado, do ponto de vista da infraestrutura e dos aspectos que dela podem ser corrompidos, talvez seja um caminho mais adequado para melhor se compreender e aplicar o paradigma de Al-Riyami e Paterson (2003).

7.2 Síntese das Contribuições

Aprimoramos modelos formais de segurança para acordo de chave com autenticação sem certificado visando dois objetivos paralelos:

1. incremento do nível de confiança que usuários podem depositar na autoridade geradora de chaves secretas parciais e
2. concretização de protocolos eficientes computacionalmente e com propriedades de segurança relevantes, como resistência a ataques de personificação, segurança no futuro e resistência ao vazamento de segredos temporários.

Para atestar que as melhorias efetuadas são praticáveis propusemos novos protocolos e demonstramos sua segurança sob os modelos estendidos ou sob os modelos de base, aplicando técnicas de segurança demonstrável como reduções de problemas computacionais.

A seguir, listamos os trabalhos gerados e relacionados com esta tese.

7.2.1 Lista de Trabalhos Publicados

Goya et al. (2009a) Denise Goya, Mehran Misaghi, Vilc Rufino e Routo Terada. Modelos de criptografia de chave pública alternativos. Em Altair Santin, Raul Ceretta Nunes e Ricardo Dahab, editors, Minicursos SBSeg 2009 IX Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais, páginas 49–97. Sociedade Brasileira de Computação. Minicurso ministrado. (Goya *et al.*, 2009a).

Goya et al. (2009b) Denise Goya, Vilc Rufino e Routo Terada. Acordo de chave sem certificados sob emissão de múltiplas chaves públicas. Em SBSeg 2010 X Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais. Sociedade Brasileira de Computação. (Goya *et al.*, 2009b).

Goya et al.(2010a) Denise Goya, Cleber Okida e Routo Terada. A two-party certificateless authenticated key agreement protocol. Em SBSeg 2010, X Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais. Sociedade Brasileira de Computação. (Goya *et al.*, 2010a).

Goya et al.(2010b) Denise Goya, Cleber Okida e Routo Terada. Aplicação de acordo de chave com autenticação sem certificado digital. Em I2TS 2010, 9th International Information and Telecommunication Technologies Symposium. (Goya *et al.*, 2010b).

Goya et al.(2011) Denise Goya, Dionathan Nakamura e Routo Terada. Acordo de chave seguro contra autoridade mal intencionada. Em SBSeg 2011, XI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais, páginas 265-278. Sociedade Brasileira de Computação. (Goya *et al.*, 2011).

Okida et al.(2011) Cleber Okida, Denise Goya e Routo Terada. Biblioteca criptográfica em java para smartphones. Em I2TS 2011, 10th International Information and Telecommunication Technologies Symposium, páginas 288-295. (Okida *et al.*, 2011).

Okida et al.(2012) Cleber Okida, Denise Goya e Routo Terada. Java cryptographic library for smartphones. Latin America Transactions, IEEE (Revista IEEE America Latina), 10(1):1377-1384. (Okida *et al.*, 2012).

7.2.2 Lista de Trabalhos Submetidos

Goya et al.(2012b) Denise Goya, Dionathan Nakamura e Routo Terada. A certificateless authenticated key agreement protocol secure against malicious KGC. artigo submetido. (Goya *et al.*, 2012b);

Goya et al.(2012c) Denise Goya, Dionathan Nakamura e Routo Terada. Certificateless key agreement protocols under strong and realistic models. artigo submetido. (Goya *et al.*, 2012c);

Monteiro et al.(2012) Fabio Monteiro, Denise Goya e Routo Terada. Aprimoramento de protocolo de identificação baseado no problema $\mathcal{MQ}$. artigo submetido. (Monteiro *et al.*, 2012).

Goya et al.(2012a) Denise Goya, Ákio Barbosa, Cleber Okida e Wilson Ruggiero. Método para análise pericial em código de software sob suspeita de plágio. artigo submetido. (Goya *et al.*, 2012a)

7.2.3 Trabalhos Anteriores

Terada e Goya(2006) Routo Terada e Denise Goya. An improved certificateless public key encryption. Em Symposium on Cryptography and Information Security 2006, volume 2A2, páginas 2.1-2.6, Tokyo, Japan. IEICE Institute of Electronics, Information, and Communication Engineers. (Terada e Goya, 2006).

Terada e Goya(2007a) Routo Terada e Denise Goya. A certificateless signature scheme based on bilinear pairing functions. Em Symposium on Cryptography and Information Security 2007, volume 2C4-5, páginas 1-7, Tokyo, Japan. IEICE Institute of Electronics, Information, and Communication Engineers. (Terada e Goya, 2007a).

Terada e Goya(2007b) Routo Terada e Denise Goya. A signature scheme based on asymmetric bilinear. Em IV Congreso Iberoamericano de Seguridad Informática, CIBSI 2007, volume 4, páginas MT1.2.1-MT1.2.14. (Terada e Goya, 2007b).

7.2.4 Outros Trabalhos

Goya(2008) Denise Goya. Modelos de segurança em assinatura sem certificado. Relatório técnico. (Goya, 2008).

Souza et al.(2011) Wellington B. Souza, Denise Goya e Routo Terada. Medidas de contorno para ataques ao cartão mifare classic. artigo preliminar. (Souza *et al.*, 2011).

7.3 Trabalhos Futuros

Algumas questões que ficaram por serem respondidas e são objetos de trabalhos futuros:

- Como se comparam implementações de uma ICP com protocolos de acordo de chave com autenticação com os aqui propostos?
- Em Yang e Tan (2011a) é proposto CL-AKA sem uso de emparelhamentos bilineares, no entanto é embutida a verificação de assinaturas digitais (no modelo sobre ICPs). É possível a remoção de assinaturas digitais nesse contexto?
- A definição de chave pública autocertificada e as formulações de chave pública sem certificado CL-BSS e CL-LK podem ser demonstradas formalmente equivalentes?
- Nossos protocolos LBG2-Gap e GOT1-Gap podem ser mostrados seguros contra ataque da autoridade mal intencionada?

Referências Bibliográficas

Abdalla *et al.***(2006)** M. Abdalla, D. Catalano, A. Dent, John Malone-Lee e Nigel Smart. Identity-based encryption gone wild. Em *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006*, volume 4052 of *Lecture Notes in Computer Science*, páginas 300–311. Springer-Verlag. ISBN 3-540-35907-9. Citado na pág. 15

Abdalla *et al.***(2008)** Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier e Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *J. Cryptol.*, 21(3):350–391. ISSN 0933-2790. doi: http://dx.doi.org/10.1007/s00145-007-9006-6. Citado na pág. 17

Al-Riyami(2005) Sattam S. Al-Riyami. *Cryptographic Schemes based on Elliptic Curve Pairings*. Tese de doutorado, Department of Mathematics, Royal Holloway, University of London. URL http://www.rhul.ac.uk/mathematics/techreports. Citado na pág. 25, 28

Al-Riyami e Paterson(2003) Sattam S. Al-Riyami e Kenneth G. Paterson. Certificateless public key cryptography. Em *ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*. Springer. ISBN 3-540-20592-6. Disponível em Cryptology ePrint Archive, Report 2003/126. Citado na pág. xi, 2, 3, 16, 19, 22, 23, 24, 28, 35, 45, 46, 48, 57, 64, 118

Al-Riyami e Paterson(2005) Sattam S. Al-Riyami e Kenneth G. Paterson. CBE from CL-PKE: A generic construction and efficient schemes. Em *Public Key Cryptography - PKC 2005*, volume 3386 of *Lecture Notes in Computer Science*, páginas 398–415, Les Diablerets, Switzerland. Springer. ISBN 3-540-24454-9. Citado na pág. 34, 36, 46

Appenzeller e Lynn(2002) Guido Appenzeller e Ben Lynn. Minimal-overhead IP security using identity-based encryption, 2002. Disponível em: http://rooster.stanford.edu/~ben/pubs/ipibe.pdf. Citado na pág. 17

Aranha e Gouvêa(2010) D. F. Aranha e C. P. L. Gouvêa. RELIC is an Efficient Library for Cryptography. http://code.google.com/p/relic-toolkit/, 2010. versão 0.2.3. Citado na pág. 80

Asokan *et al.***(2007)** N. Asokan, Kari Kostiainen, Philip Ginzboorg, Jörg Ott e Cheng Luo. Applicability of identity-based cryptography for disruption-tolerant networking. Em *MobiOpp '07: Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking*, páginas 52–56, New York, NY, USA. ACM. ISBN 978-1-59593-688-2. doi: http://doi.acm.org/10.1145/1247694.1247705. Citado na pág. 17

Au *et al.***(2007a)** Man Ho Au, Joseph K. Liu, Willy Susilo e Tsz Hon Yuen. Certificate based (linkable) ring signature. Em *ISPEC*, volume 4464 of *Lecture Notes in Computer Science*, páginas 79–92. Springer. Citado na pág. 35

Au *et al.***(2007b)** Man Ho Au, Yi Mu, Jing Chen, Duncan S. Wong, Joseph K. Liu e Guomin Yang. Malicious KGC attacks in certificateless cryptography. Em *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*, páginas 302–311, New York, NY, USA. ACM. ISBN 1-59593-574-6. doi: http://doi.acm.org/10.1145/1229285.1266997. Disponível em Cryptology ePrint Archive, Report 2009/219. Citado na pág. 28, 36, 60

Baek *et al.*(**2005**) Joonsang Baek, Reihaneh Safavi-Naini e Willy Susilo. Certificateless public key encryption without pairing. Em *ISC*, volume 3650 of *Lecture Notes in Computer Science*, páginas 134–148. Springer. Disponível em http://www.uow.edu.au/~baek/publications/clpkewp_bss_final.pdf. Citado na pág. xi, 27, 46, 47, 48

Barreto *et al.*(**2008**) Paulo S. L. M. Barreto, Alexandre Machado Deusajute, Eduardo de Souza Cruz, Geovandro C. F. Pereira e Rodrigo Rodrigues da Silva. Toward efficient certificateless signcryption from (and without) bilinear pairings. Em *SBSeg 2008*. Citado na pág. 27

Bellare e Rogaway(1994) Mihir Bellare e Phillip Rogaway. Entity authentication and key distribution. Em *Proceedings of the 13th annual international cryptology conference on Advances in cryptology*, CRYPTO '93, páginas 232–249, New York, NY, USA. Springer-Verlag New York, Inc. ISBN 0-387-57766-1. Citado na pág. 4, 54

Bellare e Rogaway(1993) Mihir Bellare e Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. Em *First ACM Conference on Computer and Communications Security*, páginas 62–73, Fairfax, Virginia, USA. ACM. Citado na pág. 4, 40, 87, 99

Bentahar *et al.*(**2008**) K. Bentahar, P. Farshim, J. Malone-Lee e N. P. Smart. Generic constructions of identity-based and certificateless KEMs. *J. Cryptol.*, 21(2):178–199. ISSN 0933-2790. doi: http://dx.doi.org/10.1007/s00145-007-9000-z. Citado na pág. 79

Boldyreva *et al.*(**2007**) Alexandra Boldyreva, Marc Fischlin, Adriana Palacio e Bogdan Warinschi. A closer look at PKI: Security and efficiency. Em *PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, páginas 458–475. Springer. Citado na pág. 22, 118

Boldyreva *et al.*(**2008**) Alexandra Boldyreva, Vipul Goyal e Virendra Kumar. Identity-based encryption with efficient revocation. Em *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security*, páginas 417–426, New York, NY, USA. ACM. ISBN 978-1-59593-810-7. doi: http://doi.acm.org/10.1145/1455770.1455823. Citado na pág. 16

Boneh e Franklin(2003) Dan Boneh e Matthew Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Comput.*, 32(3):586–615. ISSN 0097-5397. doi: http://dx.doi.org/10.1137/S0097539701398521. Citado na pág. 15, 22, 35, 41

Boneh *et al.*(**2003**) Dan Boneh, Craig Gentry, Ben Lynn e Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. Em *Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques*, EUROCRYPT'03, páginas 416–432, Berlin, Heidelberg. Springer-Verlag. ISBN 3-540-14039-5. Citado na pág. 34

Boyd e Mathuria(2003) Colin Boyd e Anish Mathuria. *Protocols for Authentication and Key Establishment*. Springer, Berlin, Germany. Citado na pág. 49, 69

Boyd *et al.*(**2009**) Colin Boyd, Yvonne Cliff, Juan M. Gonzalez Nieto e Kenneth G. Paterson. One-round key exchange in the standard model. *Int. J. Appl. Cryptol.*, 1(3):181–199. ISSN 1753-0563. doi: http://dx.doi.org/10.1504/IJACT.2009.023466. Citado na pág. 77, 78, 79, 112, 113

Brown *et al.*(**2002**) Daniel R. L. Brown, Robert P. Gallant e Scott A. Vanstone. Provably secure implicit certificate schemes. Em *FC '01: Proceedings of the 5th International Conference on Financial Cryptography*, páginas 156–165, London, UK. Springer-Verlag. ISBN 3-540-44079-8. Citado na pág. 22

Canetti e Krawczyk(2001) Ran Canetti e Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. Em *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, páginas 453–474, London, UK. Springer-Verlag. ISBN 3-540-42070-3. Citado na pág. 4, 54, 58, 78, 79

Cash *et al.*(**2009**) David Cash, Eike Kiltz e Victor Shoup. The twin Diffie-Hellman problem and applications. *J. Cryptology*, 22(4):470–504. ISSN 0933-2790. doi: http://dx.doi.org/10.1007/s00145-009-9041-6. Citado na pág. 41, 42, 70, 74

Castro *et al.*(**2007**) Rafael Castro, Ricardo Dahab e Augusto Jun Devegili. Introdução à segurança demonstrável. Em *Minicursos SBSeg 2007 VII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, páginas 103–152. SBC. Citado na pág. 40

Chatterjee e Sarkar(**2007**) Sanjit Chatterjee e Palash Sarkar. Constant size ciphertext HIBE in the augmented selective-ID model and its extensions. *J. UCS*, 13(10):1367–1395. Citado na pág. 15

Chen *et al.*(**2007**) L. Chen, Z. Cheng e N. P. Smart. Identity-based key agreement protocols from pairings. *Int. J. Inf. Secur.*, 6(4):213–241. ISSN 1615-5262. doi: http://dx.doi.org/10.1007/s10207-006-0011-9. Citado na pág. 50, 68, 70, 113, 114

Cheng e Comley(**2005**) Zhaohui Cheng e Richard Comley. Efficient certificateless public key encryption. Cryptology ePrint Archive, Report 2005/012, 2005. Citado na pág. 45

Cheng *et al.*(**2007**) Zhaohui Cheng, Liqun Chen, Li Ling e Richard Comley. General and efficient certificateless public key encryption constructions. Em *Pairing*, volume 4575 of *Lecture Notes in Computer Science*, páginas 83–107. Springer. Citado na pág. 27

Chow(**2009**) Sherman Chow. Removing escrow from identity-based encryption - new security notions and key managment techiniques. Em *Public Key Cryptography - PKC 2009*, volume 5443 of *Lecture Notes in Computer Science*, páginas 256–276. Springer. Citado na pág. 16

Chow *et al.*(**2006**) Sherman S. M. Chow, Colin Boyd e Juan Manuel González Nieto. Security-mediated certificateless cryptography. Em *Public Key Cryptography PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, páginas 508–524, New York, NY, USA. Springer. Citado na pág. 28

Cocks(**2001**) Clifford Cocks. An identity based encryption scheme based on quadratic residues. Em *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, páginas 360–363, London, UK. Springer-Verlag. ISBN 3-540-43026-1. Citado na pág. 15

Crampton *et al.*(**2007**) Jason Crampton, Hoon Wei Lim e Kenneth G. Paterson. What can identity-based cryptography offer to web services? Em *SWS '07: Proceedings of the 2007 ACM workshop on Secure web services*, páginas 26–36, New York, NY, USA. ACM. ISBN 978-1-59593-892-3. doi: http://doi.acm.org/10.1145/1314418.1314424. Citado na pág. 15

Dent(**2008**) Alexander W. Dent. A survey of certificateless encryption schemes and security models. *Int. J. Inf. Secur.*, 7(5):349–377. ISSN 1615-5262. doi: http://dx.doi.org/10.1007/s10207-008-0055-0. Disponível em Cryptology ePrint Archive, Report 2006/211. Citado na pág. 2, 28, 57, 60

Dent(**2010**) Alexander W. Dent. A brief introduction to certificateless encryption schemes and their infrastructures. Em *Proceedings of the 6th European conference on Public key infrastructures, services and applications*, EuroPKI'09, páginas 1–16, Berlin, Heidelberg. Springer-Verlag. ISBN 3-642-16440-4, 978-3-642-16440-8. Citado na pág. 45, 47

Dent *et al.*(**2008**) Alexander W. Dent, Benoît Libert e Kenneth G. Paterson. Certificateless encryption schemes strongly secure in the standard model. Em *Public Key Cryptography - PKC 2008*, volume 4939 of *Lecture Notes in Computer Science*, páginas 344–359, Berlin/Heidelberg. Springer. Disponível em Cryptology ePrint Archive, Report 2007/121. Citado na pág. 27, 28

Diffie e Hellman(**1976**) Whitfield Diffie e Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654. Citado na pág. 1, 4, 49

Dodis e Katz(2005) Yevgeniy Dodis e Jonathan Katz. Chosen-ciphertext security of multiple encryption. Em *TCC*, volume 3378 of *Lecture Notes in Computer Science*, páginas 188–209. Springer. Citado na pág. 28, 34

Fan *et al.*(2008) Xinxin Fan, Guang Gong e David Jao. Speeding up pairing computations on genus 2 hyperelliptic curves with efficiently computable automorphisms. Em *Pairing '08: Proceedings of the 2nd international conference on Pairing-Based Cryptography*, páginas 243–264, Berlin, Heidelberg. Springer-Verlag. ISBN 978-3-540-85503-3. doi: http://dx.doi.org/10.1007/978-3-540-85538-5_17. Citado na pág. 16

Farshim e Warinschi(2009) Pooya Farshim e Bogdan Warinschi. Certified encryption revisited. Em *Proceedings of the 2nd International Conference on Cryptology in Africa: Progress in Cryptology*, AFRICACRYPT '09, páginas 179–197, Berlin, Heidelberg. Springer-Verlag. ISBN 978-3-642-02383-5. doi: http://dx.doi.org/10.1007/978-3-642-02384-2_12. Citado na pág. 118

Fiore *et al.*(2010) D. Fiore, R. Gennaro e N.P. Smart. Constructing certificateless encryption and ID-based encryption from ID-based key agreement. Em *Pairing 2010*. Springer. Disponível em Cryptology ePrint Archive, Report 2009/600. Citado na pág. 78, 113, 114

Fiore e Gennaro(2010) Dario Fiore e Rosario Gennaro. Making the Diffie-Hellman protocol identity-based. Em *CT-RSA*, volume 5985 of *Lecture Notes in Computer Science*, páginas 165–178. Springer. ISBN 978-3-642-11924-8. Citado na pág. 113

Fujisaki e Okamoto(1999) Eiichiro Fujisaki e Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. Em *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, páginas 537–554, London, UK. Springer-Verlag. ISBN 3-540-66347-9. Citado na pág. 35

Galindo *et al.*(2008) David Galindo, Paz Morillo e Carla Ràfols. Improved certificate-based encryption in the standard model. *J. Syst. Softw.*, 81(7):1218–1226. ISSN 0164-1212. doi: http://dx.doi.org/10.1016/j.jss.2007.09.009. Citado na pág. 34

Gentry(2003) Craig Gentry. Certificate-based encryption and the certificate revocation problem. Em *Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques*, EUROCRYPT'03, páginas 272–293, Berlin, Heidelberg. Springer-Verlag. ISBN 3-540-14039-5. Citado na pág. 3, 16, 29, 32, 33, 35, 48

Gentry e Silverberg(2002) Craig Gentry e Alice Silverberg. Hierarchical ID-based cryptography. Em *ASIACRYPT '02: Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security*, páginas 548–566, London, UK. Springer-Verlag. ISBN 3-540-00171-9. Citado na pág. 15

Girault(1991) Marc Girault. Self-certified public keys. Em *EUROCRYPT*, volume 547 of *Lecture Notes in Computer Science*, páginas 490–497. Springer. Citado na pág. 3, 10, 13, 18, 19, 20, 21, 23, 48

Goldwasser *et al.*(1988) Sha Goldwasser, Silvio Micali e Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17:281–308. Citado na pág. 75

Goldwasser e Micali(1984) Shafi Goldwasser e Silvio Micali. Probabilistic encryption. *Journal of Computer and Systems Sciences*, 28(2):270–299. Citado na pág. 4

Goya(2008) Denise Goya. Modelos de segurança em assinatura sem certificado, 2008. Disponível em http://www.ime.usp.br/∼dhgoya/relatorioTopicos.pdf. Citado na pág. 120

Goya *et al.*(**2009a**) Denise Goya, Mehran Misaghi, Vilc Rufino e Routo Terada. Modelos de criptografia de chave pública alternativos. Em Altair Santin, Raul Ceretta Nunes e Ricardo Dahab, editors, *Minicursos SBSeg 2009 IX Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, páginas 49–97. SBC. Citado na pág. 6, 118

Goya *et al.*(**2009b**) Denise Goya, Vilc Rufino e Routo Terada. Acordo de chave sem certificados sob emissão de múltiplas chaves públicas. Em *SBSeg 2010 X Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*. Sociedade Brasileira de Computação. Citado na pág. 119

Goya *et al.*(**2010a**) Denise Goya, Cleber Okida e Routo Terada. A two-party certificateless authenticated key agreement protocol. Em *SBSeg 2010 X Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*. Sociedade Brasileira de Computação. Citado na pág. 6, 119

Goya *et al.*(**2010b**) Denise Goya, Cleber Okida e Routo Terada. Aplicação de acordo de chave com autenticação sem certificado digital. Em *I2TS 2010, 9th International Information and Telecommunication Technologies Symposium*. Citado na pág. 6, 112, 119

Goya *et al.*(**2011**) Denise Goya, Dionathan Nakamura e Routo Terada. Acordo de chave seguro contra autoridade mal intencionada. Em *SBSeg 2011 XI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, páginas 265–278. Sociedade Brasileira de Computação. Citado na pág. 6, 119

Goya *et al.*(**2012a**) Denise Goya, Ákio Barbosa, Cleber Okida e Wilson Ruggiero. Método para análise pericial em código de software sob suspeita de plágio. artigo submetido. Citado na pág. 119

Goya *et al.*(**2012b**) Denise Goya, Dionathan Nakamura e Routo Terada. A certificateless authenticated key agreement protocol secure against malicious KGC. artigo submetido. Citado na pág. 6, 119

Goya *et al.*(**2012c**) Denise Goya, Dionathan Nakamura e Routo Terada. Certificateless key agreement protocols under strong and realistic models. artigo submetido. Citado na pág. 6, 119

Günther(1989) Christoph G. Günther. An identity-based key-exchange protocol. Em *EUROCRYPT*, volume 434 of *Lecture Notes in Computer Science*, páginas 29–37. Springer. Citado na pág. 19

Hu *et al.*(**2007**) Bessie Hu, Duncan Wong, Zhenfeng Zhang e Xiaotie Deng. Certificateless signature: a new security model and an improved generic construction. *Designs, Codes and Cryptography*, 42(2):109–126. ISSN 0925-1022. doi: http://dx.doi.org/10.1007/s10623-006-9022-9. Citado na pág. 28

Hu *et al.*(**2006**) Bessie C. Hu, Duncan S. Wong, Zhenfeng Zhang e Xiaotie Deng. Key replacement attack against a generic construction of certificateless signature. Em *Information Security and Privacy, 11th Australasian Conference, ACISP 2006*, volume 4058 of *Lecture Notes in Computer Science*, páginas 235–246. Springer. Citado na pág. 28

Huang e Cao(2009) Hai Huang e Zhenfu Cao. An ID-based authenticated key exchange protocol based on bilinear Diffie-Hellman problem. Em *ASIACCS '09: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, páginas 333–342, New York, NY, USA. ACM. ISBN 978-1-60558-394-5. doi: http://doi.acm.org/10.1145/1533057.1533101. Citado na pág. 70, 74

Huang e Wong(2007) Qiong Huang e Duncan S. Wong. Generic certificateless key encapsulation mechanism. Em *Proceedings of the 12th Australasian conference on Information security and privacy*, ACISP'07, páginas 215–229, Berlin, Heidelberg. Springer-Verlag. ISBN 978-3-540-73457-4. Citado na pág. 79

Hwang *et al.*(2008) Yong Ho Hwang, Joseph K. Liu e Sherman S.M. Chow. Certificateless public key encryption secure against malicious KGC attacks in the standard model. *Journal of Universal Computer Science*, 14(3):463–480. Disponível em http://www.jucs.org/jucs_14_3/certificateless_public_key_encryption. Citado na pág. 28

IEEE(2000) IEEE. *P1363 Standard Specifications for Public-Key Cryptography*, 2000. Citado na pág. 49

Joux(2000) Antoine Joux. A one round protocol for tripartite Diffie-Hellman. Em *ANTS-IV: Proceedings of the 4th International Symposium on Algorithmic Number Theory*, volume 1838 of *Lecture Notes in Computer Science*, páginas 385–394, London, UK. Springer-Verlag. ISBN 3-540-67695-3. Citado na pág. 15

Kang e Park(2005) Bo Gyeong Kang e Je Hong Park. Is it possible to have CBE from CL-PKE? Cryptology ePrint Archive, Report 2005/431, 2005. Citado na pág. 34, 36

Kang *et al.*(2004) Bo Gyeong Kang, Je Hong Park e Sang Geun Hahn. A certificate-based signature scheme. Em *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, páginas 99–111. Springer. ISBN 3-540-20996-4. Citado na pág. 35

Kim *et al.*(1999) S. Kim, S. Oh, S. Park e D. Won. Verifiable self-certified public keys. Em *WCC'99 : Workshop on Coding and Cryptography*, páginas 139–148, Le Chesnay, França. INRIA. Citado na pág. 18

Koblitz e Menezes(2004) Neal Koblitz e Alfred Menezes. Another look at "provable security". Cryptology ePrint Archive, Report 2004/152, 2004. URL http://eprint.iacr.org/. Citado na pág. 40

Krawczyk(2005) Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. Em *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, páginas 546–566. Springer. Citado na pág. xi, 4, 49, 51, 52

Lai e Kou(2007) Junzuo Lai e Weidong Kou. Self-generated-certificate public key encryption without pairing. Em *Proceedings of the 10th international conference on Practice and theory in public-key cryptography*, PKC'07, páginas 476–489, Berlin, Heidelberg. Springer-Verlag. ISBN 978-3-540-71676-1. Citado na pág. xi, 46, 47, 48

LaMacchia *et al.*(2007) Brian LaMacchia, Kristin Lauter e Anton Mityagin. Stronger security of authenticated key exchange. Em *ProvSec'07: Proceedings of the 1st international conference on Provable security*, volume 4784 of *Lecture Notes in Computer Science*, páginas 1–16, Berlin, Heidelberg. Springer-Verlag. ISBN 3-540-75669-8, 978-3-540-75669-9. Citado na pág. 4, 51, 54, 113

Lee e Kim(2002) Byoungcheon Lee e Kwangjo Kim. Self-certified signatures. Em *INDOCRYPT '02: Proceedings of the Third International Conference on Cryptology*, páginas 199–214, London, UK. Springer-Verlag. ISBN 3-540-00263-4. Citado na pág. 22

Li *et al.*(2007) Jiguo Li, Xinyi Huang, Yi Mu, Willy Susilo e Qianhong Wu. Certificate-based signature: Security model and efficient construction. Em *EuroPKI*, volume 4582 of *Lecture Notes in Computer Science*, páginas 110–125. Springer. Citado na pág. 35

Libert e Quisquater(2006) Benoit Libert e Jean-Jacques Quisquater. On constructing certificateless cryptosystems from identity based encryption. Em *Public Key Cryptography 2006 (PKC'06)*, volume 3958 of *Lecture Notes in Computer Science*, páginas 474–490, New York, NY, USA. Springer-Verlag. Citado na pág. 27, 28, 36

Lim(2006) Hoon Wei Lim. *On the Application of Identity-Based Cryptography In Grid Security*. Doutorado, University of London. Citado na pág. 17

Lim e Paterson(2005) Hoon Wei Lim e Kenneth G. Paterson. Identity-based cryptography for grid security. Em *E-SCIENCE '05: Proceedings of the First International Conference on e-Science and Grid Computing*, páginas 395–404, Washington, DC, USA. IEEE Computer Society. ISBN 0-7695-2448-6. doi: http://dx.doi.org/10.1109/E-SCIENCE.2005.52. Citado na pág. 15

Lippold(2010) Georg Lippold. *Encryption Schemes and Key Exchange Protocols in the Certificateless Setting*. Tese de doutorado, Queensland University of Technology. Disponível em http://eprints.qut.edu.au/41697/. Citado na pág. 79

Lippold e González Nieto(2010) Georg Lippold e Juan González Nieto. Certificateless key agreement in the standard model. Em *Proceedings of the Eighth Australasian Conference on Information Security - Volume 105*, AISC '10, páginas 75–85, Darlinghurst, Australia, Australia. Australian Computer Society, Inc. ISBN 978-1-920682-86-6. Citado na pág. 3, 4, 53, 57, 79, 82

Lippold *et al.***(2009)** Georg Lippold, Colin Boyd e Juan González Nieto. Strongly secure certificateless key agreement. Em *Pairing '09: Proceedings of the 3rd International Conference Palo Alto on Pairing-Based Cryptography*, volume 5671 of *Lecture Notes in Computer Science*, páginas 206–230, Berlin, Heidelberg. Springer-Verlag. ISBN 978-3-642-03297-4. doi: http://dx.doi.org/10.1007/978-3-642-03298-1_14. Disponível em Cryptology ePrint Archive, Report 2009/219. Citado na pág. ix, xi, xvii, 3, 4, 50, 51, 53, 54, 56, 59, 67, 68, 70, 71, 75, 79, 81, 82, 99, 113, 114, 115

Lippold *et al.***(2010)** Georg Lippold, Colin Boyd e Juan Manuel González Nieto. Efficient certificateless kem in the standard model. Em *Proceedings of the 12th international conference on Information security and cryptology*, ICISC'09, páginas 34–46, Berlin, Heidelberg. Springer-Verlag. ISBN 3-642-14422-5, 978-3-642-14422-6. Citado na pág. 79

Liu e Zhou(2008) Joseph K. Liu e Jianying Zhou. Efficient certificate-based encryption in the standard model. Em *SCN '08: Proceedings of the 6th international conference on Security and Cryptography for Networks*, páginas 144–155, Berlin, Heidelberg. Springer-Verlag. ISBN 978-3-540-85854-6. doi: http://dx.doi.org/10.1007/978-3-540-85855-3_10. Citado na pág. 34

Liu *et al.***(2007)** Joseph K. Liu, Man Ho Au e Willy Susilo. Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model. Em *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*, páginas 273–283, New York, NY, USA. ACM. ISBN 1-59593-574-6. doi: http://doi.acm.org/10.1145/1229285.1266994. Citado na pág. 24, 27, 28, 47

Liu *et al.***(2008)** Joseph K. Liu, Joonsang Baek, Willy Susilo e Jianying Zhou. Certificate-based signature schemes without pairings or random oracles. Em *ISC '08: Proceedings of the 11th international conference on Information Security*, volume 5222 of *Lecture Notes in Computer Science*, páginas 285–297, Berlin, Heidelberg. Springer-Verlag. ISBN 978-3-540-85884-3. doi: http://dx.doi.org/10.1007/978-3-540-85886-7_20. Citado na pág. 35

Liu *et al.***(2009)** Joseph K. Liu, Joonsang Baek e Jianying Zhou. Certificate-based sequential aggregate signature. Em *WiSec '09: Proceedings of the second ACM conference on Wireless network security*, páginas 21–28, New York, NY, USA. ACM. ISBN 978-1-60558-460-7. doi: http://doi.acm.org/10.1145/1514274.1514278. Citado na pág. 35

Lu e Li(2008) Yang Lu e Jiguo Li. A general and secure certification-based encryption construction. Em *ChinaGrid'08*, páginas 182–189, Los Alamitos, CA. IEEE Computer Society. Citado na pág. 35

Lu *et al.***(2009)** Yang Lu, Jiguo Li e Junmo Xiao. Constructing efficient certificate-based encryption with paring. *Journal of Computers*, 4(1):19–26. Citado na pág. 35

Menezes *et al.*(**1995**) A.J. Menezes, M. Qu e S. A. Vanstone. Some key agreement protocols providing implicit authentication. Em *2nd Workshop Selected Areas in Cryptography, SAC'95*, páginas 22–32. Citado na pág. xii, 49

Menezes *et al.*(**1996**) Alfred J. Menezes, Scott A. Vanstone e Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA. Disponível em http://www.cacr.math.uwaterloo.ca/hac/. Citado na pág. 49

Monteiro *et al.*(**2012**) Fabio Monteiro, Denise Goya e Routo Terada. Aprimoramento de protocolo de identificação baseado no problema MQ. artigo submetido. Citado na pág. 119

Naccache(**2007**) David Naccache. Secure and practical identity-based encryption. *IET Information Security*, 1(2):59–64. Disponível em Cryptology ePrint Report 2005/369. Citado na pág. 16

Okamoto e Pointcheval(**2001**) Tatsuaki Okamoto e David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. Em *PKC '01: Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography*, páginas 104–118, London, UK. Springer-Verlag. ISBN 3-540-41658-7. Citado na pág. 41, 115

Okida *et al.*(**2012**) C. Okida, D. Goya e R. Terada. Java cryptographic library for smartphones. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, 10(1):1377 –1384. ISSN 1548-0992. doi: 10.1109/TLA.2012.6142487. Citado na pág. 6, 119

Okida *et al.*(**2011**) Cleber Okida, Denise Goya e Routo Terada. Biblioteca criptográfica em java para smartphones. Em *I2TS 2011, 10th International Information and Telecommunication Technologies Symposium*, páginas 288–295. Citado na pág. 119

Oliveira *et al.*(**2011**) Leonardo B. Oliveira, Diego F. Aranha, Conrado Porto Lopes Gouvêa, Michael Scott, Danilo F. Câmara, Julio López e Ricardo Dahab. TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. *Computer Communications*, páginas 485–493. Citado na pág. 15

Petersen *et al.*(**1997**) Holger Petersen, Patrick Horster e Delta Patrick Horster. Self-certified keys – concepts and applications. Em *In Proc. Communications and Multimedia Security'97*, páginas 102–116. Chapman & Hall. Citado na pág. 20, 21

Pointcheval(**2005**) David Pointcheval. *Provable Security for Public Key Schemes*, páginas 133–189. Birkhauser Publishers. Capítulo de Advanced Course on Contemporary Cryptology. Citado na pág. 40

Saeednia(**2003**) Shahrokh Saeednia. A note on Girault's self-certified model. *Inf. Process. Lett.*, 86(6):323–327. ISSN 0020-0190. doi: http://dx.doi.org/10.1016/S0020-0190(03)00203-5. Citado na pág. 19, 20

Sakai *et al.*(**2000**) R. Sakai, K. Ohgishi e M. Kasahara. Cryptosystems based on pairing. Em *Symposium on Cryptography and Information Security (SCIS2000)*, páginas 26–28, Okinawa, Japan. Inst. of Electronics, Information and Communication Engineers. Citado na pág. 5, 15

Sakai e Kasahara(**2003**) Ryuichi Sakai e Masao Kasahara. ID-based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054, 2003. URL http://eprint.iacr.org/. Citado na pág. 35, 69

Scott *et al.*(**2006**) Michael Scott, Neil Costigan e Wesam Abdulwahab. Implementing cryptographic pairings on smartcards. Em *CHES*, volume 4249 of *Lecture Notes in Computer Science*, páginas 134–147. Springer. Citado na pág. 17

Shamir(1984) Adi Shamir. Identity-based cryptosystems and signature schemes. Em *Proceedings of CRYPTO 84 on Advances in cryptology*, volume 196/1985 of *Lecture Notes in Computer Science*, páginas 47–53, New York, NY, USA. Springer-Verlag New York, Inc. ISBN 0-387-15658-5. Citado na pág. 2, 3, 11, 14

Shao(2007) Zuhua Shao. Self-certified signatures based on discrete logarithms. Em *WAIFI '07: Proceedings of the 1st international workshop on Arithmetic of Finite Fields*, páginas 252–263, Berlin, Heidelberg. Springer-Verlag. ISBN 978-3-540-73073-6. doi: http://dx.doi.org/10.1007/978-3-540-73074-3_19. Citado na pág. 22

Shao(2008) Zuhua Shao. Certificate-based verifiably encrypted signatures from pairings. *Information Sciences*, 178(10):2360–2373. ISSN 0020-0255. doi: http://dx.doi.org/10.1016/j.ins.2008.01.010. Citado na pág. 35

Shim(2003) Kyungah Shim. Efficient ID-based authenticated key agreement protocol based on Weil pairing. *Electronics Letters*, 39(8):653 – 654. Citado na pág. 68

Souza *et al.***(2011)** Wellington B. Souza, Denise Goya e Routo Terada. Medidas de contorno para ataques ao cartão mifare classic. artigo preliminar submetido. Citado na pág. 120

Sun *et al.***(2007)** Yinxia Sun, Futai Zhang e Joonsang Baek. Strongly secure certificateless public key encryption without pairing. Em *CANS*, volume 4856 of *Lecture Notes in Computer Science*, páginas 194–208. Springer. Citado na pág. 27, 47

Swanson(2008) C. M. Swanson. Security in key agreement: Two-party certificateless schemes. Dissertação de mestrado, University of Waterloo. Disponível em http://hdl.handle.net/10012/4156. Citado na pág. 50, 58, 67, 113

Swanson e Jao(2009) Colleen Swanson e David Jao. A study of two-party certificateless authenticated key-agreement protocols. Em *INDOCRYPT '09: Proceedings of the 10th International Conference on Cryptology in India*, volume 5922 of *Lecture Notes in Computer Science*, páginas 57–71, Berlin, Heidelberg. Springer-Verlag. ISBN 978-3-642-10627-9. doi: http://dx.doi.org/10.1007/978-3-642-10628-6_4. Citado na pág. xii, 2, 4, 50, 51, 53, 56, 57, 58, 59, 68, 69, 81, 105

Szczechowiak *et al.***(2008)** Piotr Szczechowiak, Leonardo B. Oliveira, Michael Scott, Martin Collier e Ricardo Dahab. Nanoecc: Testing the limits of elliptic curve cryptography in sensor networks. Em *European conference on Wireless Sensor Networks, EWSN08*, volume 4913 of *Lecture Notes in Computer Science*, páginas 305–320. Citado na pág. 15

Terada e Goya(2006) Routo Terada e Denise H. Goya. An improved certificateless public key encryption. Em *Symposium on Cryptography and Information Security 2006*, volume 2A2, páginas 2.1–2.6, Tokyo, Japan. IEICE Institute of Electronics, Information, and Communication Engineers. Citado na pág. 119

Terada e Goya(2007a) Routo Terada e Denise H. Goya. A certificateless signature scheme based on bilinear pairing functions. Em *Symposium on Cryptography and Information Security 2007*, volume 2C4-5, páginas 1–7, Tokyo, Japan. IEICE Institute of Electronics, Information, and Communication Engineers. Citado na pág. 120

Terada e Goya(2007b) Routo Terada e Denise H. Goya. A signature scheme based on asymmetric bilinear. Em *IV Congreso Iberoamericano de Seguridad Informática, CIBSI 2007*, volume 4, páginas MT1.2.1–MT1.2.14. Citado na pág. 120

Wang *et al.***(2007)** Lihua Wang, Jun Shao, Zhenfu Cao, Masahiro Mambo e Akihiro Yamamura. A certificate-based proxy cryptosystem with revocable proxy decryption power. Em *Progress in Cryptology - INDOCRYPT 2007, 8th International Conference on Cryptology in India, Chennai,*

India, December 9-13, 2007, Proceedings, volume 4859 of *Lecture Notes in Computer Science*, páginas 297–311. Springer. ISBN 978-3-540-77025-1. Citado na pág. 36

Waters(2005) Brent R. Waters. Efficient identity-based encryption without random oracles. Em *EUROCRYPT'05*, volume 3494 of *Lecture Notes in Computer Science*, páginas 114–127. Springer. Disponível em Cryptology ePrint Report 2004/180. Citado na pág. 16, 35

Yang e Tan(2011a) Guomin Yang e Chik-How Tan. Strongly secure certificateless key exchange without pairing. Em *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '11, páginas 71–79, New York, NY, USA. ACM. ISBN 978-1-4503-0564-8. doi: http://doi.acm.org/10.1145/1966913.1966924. Citado na pág. 3, 4, 57, 74, 75, 79, 82, 120

Yang e Tan(2011b) Guomin Yang e Chik How Tan. Certificateless public key encryption: A new generic construction and two pairing-free schemes. *Theor. Comput. Sci.*, 412:662–674. ISSN 0304-3975. doi: http://dx.doi.org/10.1016/j.tcs.2010.10.025. Citado na pág. 113

Yang e Tan(2011c) Guomin Yang e Chik How Tan. Certificateless cryptography with KGC trust level 3. *Theor. Comput. Sci.*, 412(39):5446–5457. Citado na pág. 26

Yao *et al.***(2004)** Danfeng Yao, Nelly Fazio, Yevgeniy Dodis e Anna Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. Em *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, páginas 354–363, New York - NY - USA. ACM Press. ISBN 1-58113-961-6. doi: http://doi.acm.org/10.1145/1030083.1030130. Citado na pág. 15

Yum e Lee(2004) Dae Hyun Yum e Pil Joong Lee. Identity-based cryptography in public key management. Em *EuroPKI 2004*, volume 3093 of *Lecture Notes in Computer Science*, páginas 71–84, Samos Island, Greece. Springer-Verlag. Citado na pág. 34, 36

Zhang e Wang(2008) Guoyan Zhang e Shaohui Wang. A certificateless signature and group signature schemes against malicious PKG. Em *22nd International Conference on Advanced Information Networking and Applications, AINA 2008*, páginas 334–341. IEEE Computer Society. Citado na pág. 28

Zhang *et al.***(2010)** Lei Zhang, Futai Zhang, Qianhong Wu e Josep Domingo-Ferrer. Simulatable certificateless two-party authenticated key agreement protocol. *Inf. Sci.*, 180:1020–1030. ISSN 0020-0255. doi: http://dx.doi.org/10.1016/j.ins.2009.11.036. Citado na pág. 3, 4, 58

Zhang *et al.***(2006)** Z. Zhang, D. S. Wong, J. XU e D. FENG. Certificateless public key signature: Security model and efficient construction. Em *4th. International Conference on Applied Cryptography and Network Security, ACNS'06*, volume 3989 of *Lecture Notes in Computer Science*, Singapore. Springer. Citado na pág. 28