Acordo de Chave Seguro contra Autoridade Mal Intencionada

Denise Goya*, Dionathan Nakamura†, Routo Terada

[1] Departamento de Ciência da Computação – Universidade de São Paulo – Brasil

{dhgoya, nakamura, rt}@ime.usp.br

Abstract. *Certificateless key agreement protocols allow authenticated key establishment without the need of digital certificate distribution and with security level higher than the one reached by identity-based key agreement protocols. In this work, we introduce an enhanced security model that is resistant to malicious authority attacks, in which an authority is able to generate system parameters with shortcuts to session key recovery. We present a new protocol that is proved secure in this extended security model and has equivalent performance to previous ones.*

Resumo. *Protocolos de acordo de chaves no modelo de criptografia de chave pública sem certificado permitem o estabelecimento de chaves secretas com autenticação, sem a necessidade de distribuição de certificados digitais e com nível de segurança maior que o alcançado por protocolos baseados em identidade. Neste artigo, propomos a extensão do modelo de segurança, tornando-o resistente a ataques de uma autoridade mal intencionada, que gera parâmetros do sistema de forma a criar atalhos para recuperação de chaves de sessão. Apresentamos um protocolo que é mostrado seguro nesse modelo estendido, com eficiência equivalente a de protocolos anteriores.*

1. Introdução

Em 2003, Al-Riyami e Paterson propuseram um modelo alternativo de chave pública que dispensa a necessidade de certificados digitais e de uma infraestrutura de chaves públicas (ICP) [Al-Riyami e Paterson 2003]. Nesse modelo, conhecido como *certificateless*, cada usuário cria um par de chaves (pública e privada, tal qual no modelo convencional), porém a autoridade do sistema, chamada Centro de Geração de Chaves ou KGC (*Key Generation Center*), fornece a cada usuário registrado uma chave secreta parcial, calculada a partir da identidade do usuário e da chave secreta mestra do KGC. Essa chave secreta parcial é um componente da chave secreta e estabelece um vínculo entre o usuário e o sistema. A certificação da chave pública ocorre implicitamente com a execução dos protocolos. Comparado ao modelo convencional em que é requerida uma ICP para geração, distribuição, validação e revogação de certificados, o modelo de Al-Riyami e Paterson simplifica os processos, requer uma infraestrutura mais simples e potencialmente reduz custos operacionais. Por outro lado, os algoritmos sob esse modelo tendem a ser mais complexos, pois preveem a existência de um adversário capaz de substituir arbitrariamente as chaves públicas dos usuários.

*O autor recebeu apoio financeiro da Fapesp, 2008/06189-0.
†O autor recebeu apoio financeiro do CNPq.

No caso particular dos protocolos de acordo de chaves com autenticação no modelo de criptografia de chave pública sem certificados (CL-AKA), participantes em uma comunicação podem se autenticar mutuamente e estabelecer chaves de sessão sem verificar certificados de chave pública. Protocolos CL-AKA são mais seguros que os baseados em identidade [Chen et al. 2007], pois as consequências do comprometimento da chave mestra secreta são atenuadas e a segurança é menos dependente do nível de confiança que os usuários precisam depositar na autoridade do sistema. Um problema acerca dos protocolos CL-AKA é a carência de opções computacionalmente eficientes que são ao mesmo tempo seguras sob forte modelo de segurança.

Um grande número de protocolos CL-AKA pode ser encontrado na literatura. No entanto, um estudo realizado por Swanson e Jao mostra que todos os protocolos existentes até então são inseguros sob um modelo de segurança adequado ao caso sem certificados [Swanson e Jao 2009]. Lippold, Boyd e González Nieto (LBG) 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]. Um segundo protocolo CL-AKA de que temos conhecimento ter sido demonstrado seguro sob o mesmo modelo é o de Goya, Okida e Terada (GOT), uma versão otimizada do antecessor LBG [Goya et al. 2010]. Ambos protocolos, LBG e GOT, são seguros sob a hipótese de dificuldade do problema Diffie-Hellman Bilinear (BDH), porém são lentos e pouco viáveis para uso prático. Nos dois casos, os autores afirmam que versões simplificadas, cerca de 50% mais velozes, são seguras sob a hipótese de dificuldade do problema Diffie-Hellman Bilinear Lacunar (Gap-BDH).

Mais recentemente, Yang e Tan propuseram protocolo CL-AKA que não requer emparelhamento bilinear [Yang e Tan 2011]. Os autores refinam a descrição do modelo de segurança dado inicialmente em [Lippold et al. 2009] e apresentam demonstração sob a hipótese de dificuldade do problema Diffie-Hellman Computacional Lacunar (Gap-DH).

Alguns protocolos de assinatura e de cifragem no modelo sem certificado são vulneráveis ao ataque do KGC mal intencionado, que gera parâmetros do sistema de forma a criar atalhos para forjar assinaturas ou decifrar textos de usuários predeterminados [Au et al. 2007]. Não é de nosso conhecimento nenhum modelo ou protocolo para CL-AKA que tenham sido propostos para segurança contra KGC mal intencionado.

Neste trabalho, apresentamos as seguintes contribuições:

- estendemos o modelo de segurança para CL-AKA, tornando-o mais forte que o de [Lippold et al. 2009] para prevenir ataques contra KGC mal intencionado;
- apresentamos um novo protocolo e respectiva demonstração de segurança sob esse modelo estendido;
- apontamos que existem falhas na demonstração de segurança do protocolo de [Yang e Tan 2011], de modo que nada se pode afirmar sobre sua segurança.

Organização do Trabalho

Na Seção 2, apresentamos conceitos necessários para a compreensão dos protocolos citados. Na Seção 3, descrevemos uma extensão do modelo de segurança para captura de ataques de um KGC mal intencionado. Na Seção 4, propomos um novo protocolo que é demonstrado seguro no modelo estendido (no Apêndice A). Na Seção 5, apontamos falha na demonstração de segurança do protocolo de [Yang e Tan 2011]. Por fim, na Seção

6 realizamos comparações com o novo protocolo proposto, seguidas das conclusões na Seção 7.

2. Conceitos Preliminares

Os protocolos aqui discutidos fazem uso de um emparelhamento bilinear admissível $e : G \times G \rightarrow G_T$, com G e G_T grupos de ordem prima q [Boneh e Franklin 2003]. Seja $P \in G$ um gerador de G e valores aleatórios $a, b, c \in \mathbb{Z}_q$. 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.

2.1. Propriedades de Segurança para CL-AKA

Dentre as propriedades de segurança mais importantes e requeridas nos protocolos de acordo de chaves com autenticação estão a resistência a ataques de personificação básicos e a ataques de personificação pelo comprometimento de chave secreta (KCI, ou *Key-Compromise Impersonation*), a segurança de chave conhecida, a segurança no futuro completa-fraca (wPFS, ou *Weak Perfect Forward Secrecy*), a resistência a ataques de compartilhamento desconhecido de chave (UKS, ou *Unknown Key-Share*) e a resistência ao vazamento de segredos temporários ou do estado da sessão [LaMacchia et al. 2007, Krawczyk 2005]. No caso especial sem certificado, é desejado também segurança no futuro perante o KGC (*KGC Forward Secrecy*) e resistência ao vazamento de segredos temporários para o KGC. Essas duas últimas propriedades são tratadas no modelo formalmente descrito em [Lippold et al. 2009], que permite um adversário:

- substituir a chave pública de um dado usuário *ou* revelar o valor secreto correspondente à chave pública de um usuário;
- 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.

Esse modelo é uma variante do Canetti-Krawczyk estendido [LaMacchia et al. 2007]. Um protocolo CL-AKA demonstrado seguro no modelo de [Lippold et al. 2009] se mantém seguro ainda que o adversário corrompa no máximo dois dos três componentes de cada um dos participantes da sessão de Teste, que denotamos por A e B. Por exemplo, sobre o participante A, o adversário pode efetuar, no máximo, duas das três linhas abaixo:

- revelar o valor secreto x_A ou substituir a chave pública de A;
- revelar a chave secreta parcial d_A ou revelar a chave mestra secreta;
- revelar o valor temporário de sessão r_A.

Todos os demais usuários podem ser integralmente corrompidos pelo adversário. A principal diferença entre os modelos propostos em [Swanson e Jao 2009] e em

[Lippold et al. 2009] 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 sem certificado [Dent 2008]. O modelo *Strong* é estritamente mais forte que o *Weak*, isto é, os protocolos que são seguros no primeiro também o são no segundo. Outra diferença no modelo de Swanson e Jao é que o adversário que conhece a chave mestra secreta não pode substituir chaves públicas.

Outro modelo de segurança que sabemos ter sido desenvolvido para protocolos CL-AKA foi apresentado em [Zhang et al. 2010]. No entanto, trata-se de um modelo mais fraco que restringe os poderes 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 são resistentes a ataques do tipo KCI.

Lippold e González Nieto apresentaram uma versão mais fraca de modelo para mostrar a segurança no modelo padrão de uma construção geral de CL-AKA [Lippold e González Nieto 2010]. Nesse modelo mais fraco, o adversário tem as mesmas restrições que as de [Zhang et al. 2010], além de não poder revelar chaves de sessão para os casos em que um dos participantes tenha a chave pública substituída. Com essas limitações, o modelo fica mais fraco que o de Swanson e Jao.

3. Extensão do Modelo de Segurança

Nesta seção, descrevemos informalmente o modelo de segurança que previne ataques do KGC mal intencionado. Tomamos como ponto de partida o modelo de Lippold et al., mas alterações similares podem ser feitas sobre o modelo de [Swanson e Jao 2009].

O modelo de Lippold et al. 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 que pode revelar chaves parciais de entidades à sua escolha, pode substituir chaves públicas ou revelar segredos temporários de sessão. 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. 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. Também é preciso modificar o comportamento do oráculo que cria novos usuários (desonestos, isto é, sob total controle do adversário); nesse caso, o adversário informa o valor da chave secreta parcial, além da identidade e chave pública do usuário. Esse atacante interno mal intencionado pode substituir chaves públicas e revelar segredos temporários de sessão.

Duas sessões são consideradas com *matching* se: (1) envolvem os mesmos participantes, (2) se na primeira sessão um participante é o emissor e o outro o receptor, então na segunda sessão os participantes devem inverter os papéis e (3) as mensagens de saída de uma sessão são iguais às de entrada na outra e vice-versa. Uma sessão é considerada *fresh* se:

- ela se encerrou e o adversário não revelou a chave de sessão;
- nenhum de seus participantes teve mais que dois segredos corrompidos;
- não existe sessão com *matching*que tenha tido sua chave secreta revelada.

O adversário pode realizar uma única consulta ao oráculo de Teste, sobre uma sessão necessariamente *fresh*. Para responder esse oráculo, o simulador joga uma moeda não viciada para escolher se entrega a verdadeira chave da sessão de Teste ou um número aleatório. O adversário vence o jogo contra o simulador se puder advinhar qual foi o resultado da moeda jogada. A *vantagem* do adversário é definida como a distância entre $0{,}5$ e a probabilidade dele vencer o jogo.

Um protocolo CL-AKA é dito *seguro* se qualquer adversário, externo ou interno mal intencionado, tem vantagem negligenciável sob o parâmetro de segurança. A descrição formal desses conceitos e modelo segue [Bellare e Rogaway 1993a] e [LaMacchia et al. 2007].

4. Novo Protocolo CL-AKA Seguro no Modelo Estendido

Passamos a descrever um novo protocolo CL-AKA que pode ser demonstrado seguro no modelo estendido, descrito na Seção 3. Os parâmetros do sistema incluem um emparelhamento bilinear $e : G \times G \to G_T$, três funções de hash criptográficas $H : \{0,1\}^* \to \{0,1\}^k$ $H_1 : \{0,1\}^* \to G$ e $H_2 : \{0,1\}^* \to G$, além da chave mestra pública sP, calculada a partir da chave mestra secreta s do KGC. Um usuário identificado por sua identidade, digamos A, possui um valor público $(Q_{A_1} = H_1(A), Q_{A_2} = H_2(A))$, um par para chave secreta parcial $(d_{A_1} = sQ_{A_1}, d_{A_2} = sQ_{A_2})$, que é calculado e entregue pelo KGC de forma segura, um valor secreto x_A e a correspondente chave pública x_AP.

Para estabelecerem uma chave secreta em comum, dois usuários A e B escolhem seus valores secretos temporários, respectivamente $r_A, r_B \in \mathbb{Z}_q$, e calculam r_AP e r_BP. Então trocam as seguintes mensagens

$A \to B : E_A = (r_AP, x_AP)$ $\qquad$ $B \to A : E_B = (r_BP, x_BP)$

Ao receberem a mensagem do parceiro, verificam a pertinência a G^2 e:

A calcula:	B calcula:
$K = e(r_BP + Q_{B_1}, r_AsP + d_{A_1})$	$K = e(r_AP + Q_{A_1}, r_BsP + d_{B_1})$
$L = e(r_BP + Q_{B_2}, r_AsP + d_{A_2})$	$L = e(r_AP + Q_{A_2}, r_BsP + d_{B_2})$
$M = e(x_BP, d_{A_1}) \cdot e(Q_{B_1}, x_AsP)$	$M = e(x_AP, d_{B_1}) \cdot e(Q_{A_1}, x_BsP)$
$Z = (x_Ax_BP, x_Ar_BP, r_Ar_BP, r_Ax_BP)$	$Z = (x_Bx_AP, r_Bx_AP, r_Br_AP, x_Br_AP)$
$SK = H(A, B, E_A, E_B, K, L, M, Z)$	$SK = H(A, B, E_A, E_B, K, L, M, Z)$

4.1. Segurança do Novo Protocolo

Para a segurança do protocolo proposto, 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 proposto.

A redução indica que se houver um adversário com vantagem não negligenciável contra o protocolo, sob o modelo de adversário estendido da Seção 3, então existe algoritmo de tempo polinomial que resolve o Gap-BDH. A seguir, enunciamos o teorema que relaciona a vantagem do adversário com a do solucionador do Gap-BDH; no apêndice A, apresentamos sua demonstração sob o modelo de oráculo aleatório [Bellare e Rogaway 1993b].

Teorema 1 *Sob a hipótese de dificuldade do problema Gap-BDH, se as funções H, H_1 e H_2 são modeladas como oráculos aleatórios, então o protocolo CL-AKA Novo é seguro.*

5. Revisão do Protocolo de Yang e Tan

Yang e Tan propuseram pequenas modificações na especificação formal do modelo de segurança de [Lippold et al. 2009], porém sem alterar as propriedades essenciais, como permitir adversário ativo, isto é, que pode escolher arbitrariamente o valor do temporário secreto dos envolvidos em uma sessão de comunicação [Yang e Tan 2011]. Os autores propuseram um protocolo CL-AKA sem emparelhamento bilinear e apresentaram demonstração de segurança sob a hipótese de dificuldade do Gap-DH. O protocolo de Yang e Tan é menos eficiente no uso do canal de comunicação, porém tende a ser mais eficiente computacionalmente, dependendo do esquema de assinatura escolhido.

No entanto, na prova de segurança, os autores não tratam corretamente os casos em que o adversário revela chaves de sessões diferentes da de Teste, mas que envolvem seus participantes e que possuem *matching* com outra sessão. Nessas situações (por exemplo no caso $(1f)$ da demonstração), a simulação pode ser abortada e o simulador não poderá aproveitar a vantagem do adversário. Por esse motivo, não se pode afirmar que o protocolo é seguro no modelo prometido.

6. Comparações

O protocolo novo tem desempenho computacional similar aos anteriores, porém apresenta nível de segurança maior com relação a um adversário interno, pois foi mostrado seguro no modelo estendido apresentado na Seção 3. A Tabela 1 mostra os protocolos LBG [Lippold et al. 2009], GOT [Goya et al. 2010] e o novo (descrito na Seção 4), todos no nível de segurança para o problema Gap-BDH, em dois cenários diferentes. O cenário normal exibe os protocolos da maneira como eles são definidos, contando-se o tempo desde a escolha do valor secreto temporário rP até o cálculo da chave de sessão SK.

Tabela 1. Comparação dos protocolos seguros sob o problema Gap-BDH

		Normal			Pré-computação		
		LBG-Gap	GOT-Gap	Novo	LBG-Gap	GOT-Gap	Novo
Emparelhamentos		4	4	4	1	1	2
Exponenciações em G_T		2	0	0	1	0	0
Multiplicações em G_T		2	1	1	1	0	0
Multiplicações em G		5	7	6	4	5	5
Adições em G		0	2	4	0	2	4
Modelo de segurança		Lippold et al.		est.	Lippold et al.		est.
Tempo (s)	B-271	0,062	0,061	0,062	0,019	0,019	0,033
	B-1223	3,504	3,432	3,442	0,992	0,955	1,769

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.

Os protocolos foram codificados com a biblioteca criptográfica Relic (versão 0.2.3) [Aranha e Gouvêa], que é escrita em linguagem de programação C. A Tabela 1 apresenta dois tempos que se referem às execuções empregando as curvas binárias B-271 e B-1223 presentes na biblioteca. Essas curvas apresentam nível de segurança mínimo de 70 bits e 128 bits respectivamente. 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) e sistema operacional Ubuntu 11.04. Apesar do processador ter 2 núcleos, os programas são executados em apenas uma única *thread*. Os programas são compilados e executados em 64 bits. Os tempos da Tabela 1 foram obtidos com essa configuração.

Com base nos resultados apresentados na Tabela 1, observa-se que em uso normal os protocolos possuem aproximadamente os mesmos tempos de execução. Já com pré-computação, o novo protocolo não obtém vantagem de tempo. Na tabela não foram incluídos os protocolos que são seguros sob outros modelos, pois apresentam nível de segurança inferior.

7. Conclusões

Estendemos o modelo de segurança mais forte dentre os existentes para protocolos de acordo de chave no modelo de chave pública sem certificado, tornando-o resistente a ataques de uma autoridade mal intencionada. Apresentamos um novo protocolo que é seguro nesse modelo estendido, sob a hipótese de dificuldade do problema Gap-BDH, no modelo de oráculos aleatórios. O protocolo proposto tem desempenho equivalente a outros que foram mostrados seguros em condições similares, mas ocupa um patamar de segurança mais elevado. Apontamos que o protocolo de Yang e Tan, que tem potencial para implementações eficientes, apresenta falhas na demonstração de segurança. Estamos trabalhando em duas variações sobre o protocolo aqui proposto, uma para garantir nível de segurança maior com o problema computacional BDH (melhor que Gap-BDH) e outra para maior eficiência com modelo de segurança melhorado de Swanson e Jao. Como trabalho futuro, sugerimos a busca por protocolos que sejam mostrados seguros no modelo padrão, sem oráculos aleatórios.

Referências

Al-Riyami, S. S. e Paterson, K. G. (2003). Certificateless public key cryptography. In *ASIACRYPT 2003*, volume 2894 of *LNCS*. Springer.

Aranha, D. F. e Gouvêa, C. P. L. RELIC is an Efficient LIbrary for Cryptography. `http://code.google.com/p/relic-toolkit/`.

Au, M. H., Mu, Y., Chen, J., Wong, D. S., Liu, J. K. e Yang, G. (2007). Malicious kgc attacks in certificateless cryptography. In *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, ASIACCS '07, pages 302–311, New York, NY, USA. ACM.

Bellare, M. e Rogaway, P. (1993a). Entity authentication and key distribution. In *LNCS - CRYPTO'93*, pages 232–249. Springer Berlin. v.773.

Bellare, M. e Rogaway, P. (1993b). Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA. ACM.

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

Chen, L., Cheng, Z. e Smart, N. P. (2007). Identity-based key agreement protocols from pairings. *Int. J. Inf. Secur.*, 6(4):213–241.

Dent, A. W. (2008). A survey of certificateless encryption schemes and security models. *Int. J. Inf. Secur.*, 7(5):349–377.

Goya, D., Okida, C. e Terada, R. (2010). A two-party certificateless authenticated key agreement protocol. In *SBSeg 2010 X Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*. Sociedade Brasileira de Computação.

Krawczyk, H. (2005). Hmqv: a high-performance secure diffie-hellman protocol. In *Advances in Cryptology, CRYPTO 2005, LNCS 3621*, page 546566.

LaMacchia, B., Lauter, K. e Mityagin, A. (2007). Stronger security of authenticated key exchange. In *ProvSec'07: Proceedings of the 1st international conference on Provable security*, volume 4784 of *LNCS*, pages 1–16, Berlin, Heidelberg. Springer-Verlag.

Lippold, G., Boyd, C. e González Nieto, J. (2009). Strongly secure certificateless key agreement. In *Pairing '09: Proceedings of the 3rd International Conference Palo Alto on Pairing-Based Cryptography*, volume 5671 of *LNCS*, pages 206–230, Berlin, Heidelberg. Springer-Verlag.

Lippold, G. e González Nieto, J. (2010). Certificateless key agreement in the standard model. In *Proceedings of the Eighth Australasian Conference on Information Security – volume 105*, AISC '10, pages 75–85. Australian Computer Society, Inc.

Swanson, C. e Jao, D. (2009). A study of two-party certificateless authenticated key-agreement protocols. In *INDOCRYPT '09: Proceedings of the 10th International Conference on Cryptology in India*, volume 5922 of *LNCS*, pages 57–71, Berlin, Heidelberg. Springer-Verlag.

Yang, G. e Tan, C.-H. (2011). Strongly secure certificateless key exchange without pairing. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '11, pages 71–79, New York, NY, USA. ACM.

Zhang, L., Zhang, F., Wu, Q. e Domingo-Ferrer, J. (2010). Simulatable certificateless two-party authenticated key agreement protocol. *Inf. Sci.*, 180:1020–1030.

A. Demonstração do Teorema 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 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

Tabela 2. 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

$\mathcal{A}$, nos moldes do jogo descrito no modelo de segurança de [Lippold et al. 2009] com as ampliações descritas na Seção 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 $answer$.

Antes do início do jogo entre o simulador $\mathcal{S}$ e o adversário $\mathcal{A}$, $\mathcal{S}$ tenta advinhar 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 2, 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 advinhar 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 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 no Apêndice A.1. 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 *onwner* ID_i e *peer* 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 no Apêndice A.1. 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 no Apêndice A.1.

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 A.1. Então $\mathcal{S}$ resolve o problema Gap-BDH em tempo polinomial em k, o que contradiz a hipótese de dificuldade do Gap-BDH. □

A.1. 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 $(ID_i, ID_j, E_i, E_j, \ldots, SK)$, preechendo 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

Tabela 3. 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

(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 3, 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 3 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 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 4. 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, $x_AP = aP$, $\mathcal{S}$ 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

Tabela 4. Nomenclatura das variáveis

$$K = \underbrace{e(r_B P, d_{A_1})}_{K_1} \cdot \underbrace{e(Q_{B_1}, r_A s P)}_{K_2} \cdot \underbrace{e(Q_{B_1}, d_{A_1})}_{K_3} \cdot \underbrace{e(r_B P, r_A s P)}_{K_4}$$

$$L = \underbrace{e(r_B P, d_{A_2})}_{L_1} \cdot \underbrace{e(Q_{B_2}, r_A s P)}_{L_2} \cdot \underbrace{e(Q_{B_2}, d_{A_2})}_{L_3} \cdot \underbrace{e(r_B P, r_A s P)}_{L_4 (= K_4)}$$

$$M = \underbrace{e(x_B P, d_{A_1})}_{M_1} \cdot \underbrace{e(Q_{B_1}, x_A s P)}_{M_2} \quad Z = (\underbrace{x_A x_B P}_{Z_1}, \underbrace{x_A r_B P}_{Z_2}, \underbrace{r_A r_B P}_{Z_3}, \underbrace{r_A x_B P}_{Z_4})$$

9, $\mathcal{S}$ desconhece a chave parcial secreta $d_{A_1} = abP$, pois não sabe calcular ab. Na Tabela 3, 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, no Apêndice A.2.

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 $\langle aP, r_cP, yP, K_1\rangle$ e $\langle aP, x_cP, yP, M_1\rangle$ 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 $\langle aP, bP, cP, r_AP, r_BP, K\rangle$ ao oráculo de decisão-BDH$^+$ (ver Apêndice A.2); se o oráculo responder positivamente, $\mathcal{S}$ calcula $K' = \frac{K}{K_2 \cdot K_4}$ $L' = \frac{L}{L_2 \cdot L_4}$ $U_1 = \left[e(\frac{y_A}{z}(r_BP + y_BP) - y_AcP - y_BbP, aP)\right]^{\frac{-1}{1+z}}$ $U_2 = [L']^{\frac{-1}{z}}$ $K_3 = U_1 \cdot \left[\frac{K'}{U_2}\right]^{\frac{1}{1+z}}$ e $answerBDH \leftarrow K_3$.
Se $\mathcal{A}$ consulta RevealSessionKey(A, B, s), $\mathcal{S}$ testa K como acima e fornece $\langle aP, bP, cP, x_AP, x_BP, M\rangle$ 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.

A.2. Variantes do problema Gap-BDH

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.