Implementação Eficiente de Protocolos de Acordo de Chave em Dispositivos de Poder Computacional Restrito

Rafael Will Macedo de Araujo[1], **Routo Terada**[1]

[1]Instituto de Matemática e Estatística – Universidade de São Paulo (USP)
Rua do Matão, 1010 - Butantã – São Paulo – SP – Brazil

{rwill,rt}@ime.usp.br

Abstract. *Protocols for authentication and key establishment are fundamental parts in security implementations for electronic devices communication. In applications involving devices with limited computational power (such as smartphones and tablets) communicating with a server, the choice of efficient protocols that require a simpler infrastructure is essential. In this work we implement secure key agreement protocols in ID-based and Certificateless public key cryptography models on ARM processor platforms. We also compare running times, memory and network usage.*

Resumo. *Protocolos de autenticação e de estabelecimento de chaves são peças fundamentais em implementações de segurança para comunicação de dispositivos eletrônicos. Em aplicações que envolvam dispositivos com poder computacional restrito (tais como smartphones ou tablets) comunicando-se com um servidor, é primordial a escolha de protocolos eficientes e que necessitem de uma infraestrutura mais simples. Neste artigo implementamos protocolos de acordo de chave seguros nos modelos de criptografia de chave pública baseado em identidade (ID-based) e sem certificado (Certificateless) em plataformas com processadores ARM. Comparamos tempos de execução, utilização de memória e uso do canal de comunicação.*

1. Introdução

O gerenciamento de certificados digitais requer uma infraestrutura de suporte mais robusta, o que acarreta em custos maiores de implementação e manutenção. Além disso, há uma série de dificuldades nos processos de de recuperação, validação e revogação de certificados. De maneira geral, os sistemas convencionais utilizam o criptossistema RSA, baseado no problema de fatoração de números inteiros grandes. Neste tipo de sistema as chaves públicas são números escolhidos aleatoriamente, não possuindo qualquer vínculo com o seu dono.

Os modelos de criptografia de chave pública baseado em identidade e sem certificado são alternativas ao modelo convencional de criptografia assimétrica. Em ambos os modelos a autenticação da chave pública ocorre implicitamente, durante a execução dos protocolos, sem a necessidade de gerenciar e distribuir certificados digitais. Tais sistemas, aliados ao uso de curvas elípticas, possuem um custo operacional e computacional muito menor.

Fixado um cenário onde um dispositivo com poder baixo poder computacional (como celular ou *tablet*) precisa se comunicar com um servidor, com garantia de confidencialidade, integridade e autenticidade de usuários e de equipamentos, sem prévia

distribuição de segredos compartilhados, a solução deve girar em torno de tecnologias de criptografia de chave pública. Através de protocolos de acordo de chave com autenticação, é possível estabelecer chaves compartilhadas com garantia de autenticidade entre dois usuários remotos, para uso posterior com um algoritmo de criptografia simétrica. Assim, também é possível conseguir confidencialidade e integridade.

Organização do Trabalho

Na Seção 2, apresentamos os fundamentos matemáticos necessários para melhor compreensão dos protocolos e experimentos. Na Seção 3, descrevemos os conceitos elementares dos modelos de criptografia de chave pública no qual se baseiam os protocolos estudados. Na Seção 4, justificamos a escolha dos protocolos de acordo de chave analisados. Na Seção 5, apresentamos o ambiente utilizado para os experimentos, ilustramos possíveis otimizações e comparamos os resultados obtidos.

2. Fundamentos Matemáticos

Estabelecemos algumas notações, e revisamos algumas definições matemáticas importantes para a compreensão deste trabalho. Durante o texto utilizamos $\mathbb{G}_1$ e $\mathbb{G}_2$ para designar grupos (elípticos) finitos de ordem prima q. A notação $a \in_R \mathcal{D}$ designa que um valor a pertencente ao domínio $\mathcal{D}$ é escolhido de maneira aleatória, com igual probabilidade, entre todos os elementos pertencentes a $\mathcal{D}$.

2.1. Emparelhamento bilinear

Um emparelhamento bilinear é definido como um mapeamento $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, onde $\mathbb{G}_T$ é um grupo multiplicativo de ordem q, que possui as seguintes propriedades (considere $P, R \in \mathbb{G}_1$, $Q, S \in \mathbb{G}_2$ e $a, b \in \mathbb{Z}_q$):

1. $e(P + R, Q) = e(P, Q) \cdot e(R, Q)$
2. $e(P, Q + S) = e(P, Q) \cdot e(P, S)$
3. Das propriedades 1 e 2, temos que: $e(aP, bQ) = e(P, bQ)^a = e(aP, Q)^b = e(P, Q)^{ab}$

Quando $\mathbb{G}_1 = \mathbb{G}_2$ dizemos que o emparelhamento é simétrico.

2.2. Problemas computacionais

Seja g um gerador de $\mathbb{Z}_q^*$, p um gerador de g, o problema do logaritmo discreto sobre os inteiros consiste em encontrar r, dado $y = g^r \bmod p$. Sua variante para grupos elípticos (representados neste texto por $\mathbb{G}$) consiste em encontrar um inteiro s, dado $R = sP$, onde P é um gerador de $\mathbb{G}$. Acredita-se que em certos grupos(como no caso dos grupos elípticos), este problema seja mais difícil de ser resolvido do que o problema de fatoração de inteiros, no qual se baseia o RSA [Hoffstein et al. 2008].

A partir do problema do logaritmo discreto, surgiram outros problemas relacionados, conhecidos como problemas *Diffie-Hellman*. Definimos a seguir os problemas computacionais no qual se apoiam os protocolos que serão apresentados no decorrer deste texto:

- **DDH** (*Decision Diffie-Hellman Problem*): dados: $P, aP, bP, cP \in \mathbb{G}$; decidir: $abP = cP$.

- **CDH** (*Computacional Diffie-Hellman Problem*): dados: $P, aP, bP \in \mathbb{G}$; encontrar: abP.
- **GBDH** (*Gap-Bilinear Diffie-Hellman Problem*): dados: $P, aP, bP \in \mathbb{G}$; encontrar: abP, com ajuda de um oráculo de decisão (que dados $aP, bP, cP \in \mathbb{G}$, decide se $abP = cP$).
- **BDH** (*Bilinear Diffie-Hellman Problem*): dados: $P, aP, bP, cP \in \mathbb{G}$; encontrar: $e(P, P)^{abc}$.

A família de problemas lacunares (*gap*) foi apresentada por [Okamoto and Pointcheval 2001]. Já o problema BDH foi definido por [Boneh and Franklin 2003]. Do ponto de vista teórico, um protocolo que tenha sido demonstrado seguro sob a hipótese de dificuldade do problema BDH é, pelo menos, tão seguro quanto outro que tenha sido demonstrado seguro sob a hipótese de dificuldade do problema Gap-BDH, no mesmo modelo de segurança. Por esse motivo, é preferível utilizar protocolos que se apoiem na suposição de dificuldade do problema BDH, e o uso do Gap-BDH deve ser evitado sempre que possível [Goya 2011].

3. Modelos Alternativos de Criptografia de Chave Pública

Apresentamos em maiores detalhes os dois modelos de criptografia de chave pública nos quais se baseiam os protocolos analisados neste trabalho. Estes modelos necessitam uma infraestrutura de suporte mais simples que a ICP (Infraestrutura de Chaves Públicas). Ademais, fazem uso de curvas elípticas, o que possibilita a utilização de corpos finitos de menor ordem, propiciando ganho de eficiência computacional ao mesmo tempo que mantém um elevado nível de segurança.

3.1. Criptografia de chave pública baseado em identidade (*ID-Based*)

O modelo de criptografia de chave pública baseado em identidade foi proposto por [Shamir 1984]. Nele, a chave pública do usuário deixa de ser um valor gerado aleatoriamente, e passa a ser sua própria identidade no sistema (ou seja, dados pessoais como nome, número do CPF ou telefone, endereço IP, etc). A identificação neste caso é implícita, e dispensa o uso de certificados digitais.

Este modelo pressupõe a existência de uma autoridade de confiança, denominada PKG (*Public Key Generator*), responsável pela geração e distribuições das chaves das chaves secretas dos usuários. Inicialmente, o PKG calcula $s \in_R \mathbb{Z}_q^*$, que será a chave mestra secreta do sistema. A partir deste valor, a autoridade do sistema calcula a chave secreta de cada usuário U como sendo $S_U = s\mathcal{H}(ID_U)$, onde $\mathcal{H}$ é uma função de *hash* tal que $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{G}$ e ID_U é a identidade do usuário U.

É importante ressaltar que o nível de criticidade da chave mestra secreta é muito elevado, já que as chaves secretas de todos os usuários dependem dela. Seu comprometimento resultaria em uma quebra total do sistema. Outro problema inerente à este modelo é a propriedade de custódia de chaves: o PKG calcula, e portanto conhece todas as chaves secretas. Neste caso, a autoridade do sistema tem acesso à todas as comunicações sigilosas, e também poderia facilmente personificar qualquer usuário sem ser identificado. Portanto, este modelo é recomendado somente para uso interno em uma dada instituição.

Também vale salientar que em seu artigo original, [Shamir 1984] propôs apenas um esquema de assinatura. Somente em [Boneh and Franklin 2001] foi demonstrado um

esquema de encriptação prático, através do uso de emparelhamentos bilineares sobre curvas elípticas. Com base neste último trabalho surgiram outras variantes, como esquemas de acordo de chave baseados em identidade.

3.2. Criptografia de chave pública sem certificado (*Certificateless*)

Com o objetivo de eliminar a propriedade de custódia de chaves e reduzir a criticidade da chave mestra secreta, [Al-Riyami and Paterson 2003] introduziu o paradigma de criptografia de chave pública sem certificado. Tal modelo pressupõe a existência de uma autoridade de confiança, denominada KGC (*Key Generation Center*), que funciona de forma semelhante ao PKG do modelo baseado em identidade.

Cada usuário U calcula $x_U \in_R \mathbb{Z}_q^*$, que será seu segredo e fará parte de sua chave secreta. O KGC, por sua vez, calcula e envia $D_U = s\mathcal{H}(ID_U)$, que é a chave secreta parcial de U. Ao receber D_U, o usuário calcula sua chave secreta completa como sendo $S_U = x_U D_U$. Já a chave pública do usuário será $\langle x_U P, ID_U \rangle$, onde P é um parâmetro público e um gerador do grupo algébrico utilizado no sistema.

Nota-se que o KGC não é capaz de calcular a chave secreta de um usuário U, já que ele desconhece o valor x_U. Caso a chave mestra secreta seja comprometida, um atacante não conseguirá calcular a chave secreta completa de U pelo mesmo motivo. Uma desvantagem deste modelo é a necessidade de um repositório de chaves públicas, todavia não há necessidade de certificados pois a identidade de um usuário fica parcialmente relacionada à sua chave secreta.

4. Protocolos Analisados

A escolha dos protocolos a serem analisados levou em consideração aspectos importantes, tais como o modelo de segurança empregado, relevância do protocolo (quantidade de citações aos artigos em que são apresentados), ano de publicação (protocolos apresentados recentemente) e ausência de publicações científicas que comprovem falha de segurança ou quebra do protocolo. Outra exigência essencial foi a garantia de autenticidade dos usuários (protocolos de acordo de chave com autenticação, ou AKA - *Authenticated Key Agreement*).

Segundo [Krawczyk 2005] e [LaMacchia et al. 2007], protocolos de acordo de chave com autenticação devem dispor de certas propriedades de segurança, como: resistência a ataques de personificação básicos, segurança de chave conhecida, resistência a ataques de compartilhamento desconhecido de chave (*unknown key-share*), resistência a ataques de personificação pelo comprometimento de chave secreta (*key-compromise impersonation*), segurança futura perfeita (*perfect forward secrecy*), segurança futura perfeita-fraca (*weak perfect forward secrecy*) e resistência ao vazamento de segredos temporários. Para o caso especifico em que não há necessidade de certificados digitais para as chaves públicas, são desejáveis mais duas propriedades complementares: segurança no futuro perante o KGC (*KGC forward secrecy*), e resistência ao vazamento de segredos temporários para o KGC.

Um modelo de segurança para protocolos AKA é uma modelagem, através de uma máquina de Turing probabilística, das sessões entre os participantes e de um adversário $\mathcal{A}$. A diferença fundamental entre modelos está no poder do adversário. Quanto maior seu poder, mais seguro é o modelo.

Os protocolos escolhidos no modelo baseado em identidade foram demonstrados seguros sobre os modelos CK e eCK, definidos em [Canetti and Krawczyk 2001] e [LaMacchia et al. 2007] respectivamente. Estes modelos diferem-se basicamente no modo como tratam os vazamentos de segredos temporários. Enquanto no modelo CK todos os segredos temporários de uma sessão podem ser revelados de uma única vez, no modelo eCK os temporários podem ser comprometidos um a um, à escolha do adversário.

Já protocolos escolhidos no modelo sem certificado foram demonstrados seguros sobre os modelos LBG (definido em [Lippold et al. 2009]), SJ$^+$, Mal-LBG e Mal-SJ$^+$ (definidos em [Goya 2011]). Todos são extensões do modelo eCK que incluem adaptações para o caso sem certificado. O adversário $\mathcal{A}$ no modelo LBG possui maior poder do que no modelo SJ$^+$, sendo classificados por [Goya 2011] como modelos de segurança contra adversário forte e moderado, respectivamente. Já os modelos Mal-LBG e Mal-SJ$^+$ são extensões dos modelos anteriores, mas com proteção contra autoridade mal intencionada. Isto é, preveem também um atacante interno que conhece a chave mestra secreta.

A seguir são listados os protocolos de acordo de chave estudados, o problema computacional e o modelo de segurança em que se baseiam.

Protocolos no modelo baseado em identidade:

- **HC-BDH**: seguro no modelo eCK, [Huang and Cao 2009]
- **HLZ-GBDH**: seguro no modelo eCK, [Hu et al. 2009]
- **CC-BDH**: seguro no modelo CK, [Chow and Choo 2007]
- **NCLH-BDH** e **NCLH-GBDH**: seguro no modelo eCK, [Ni et al. 2011]
- **NCL-BDH**: seguro no modelo eCK, [Ni et al. 2012]

Protocolos no modelo sem certificado:

- **LBG-BDH** e **LBG-GBDH**: seguro no modelo LBG, [Lippold et al. 2009]
- **GOT-BDH** e **GOT-GBDH**: seguro no modelo LBG, [Goya et al. 2010]
- **GNT3-BDH**: seguro no modelo LBG, [Goya 2011]
- **GNT1-GBDH**: seguro no modelo Mal-LBG, [Goya et al. 2011]
- **GNT2-GBDH**: seguro no modelo Mal-SJ$^+$, [Goya 2011]
- **GNT4-BDH**: seguro no modelo SJ$^+$, [Goya 2011]

5. Experimentos

O cenário proposto para os experimentos consiste em acordar uma chave secreta entre um dispositivo de baixo poder computacional (cliente) e um servidor, que funciona como a autoridade do sistema (PKG ou KGC). Neste ambiente não há distribuição prévia de segredos compartilhados. A comunicação de ambos acontece por meio de uma rede TCP/IP, utilizando os padrões IEEE 802.11g (para comunicação sem fio) e 100BASE-TX (*Fast Ethernet*), a depender do dispositivo. A Figura 1 ilustra este cenário.

Foram escolhidos três dispositivos para a realização dos experimentos: um *smartphone*, um *tablet*, e um *single-board computer* voltado para sistemas embarcados, todos com processadores ARM. Tais dispositivos dispõem de sistema operacional Android ou Linux. A seguir são detalhadas as configurações dos equipamentos utilizados:

- **Servidor (PC)**: processador Intel Core 2 Duo T5800 de 2.0 GHz, memória RAM de 3 GB, e sistema operacional Ubuntu 12.04 LTS de 32 bits.

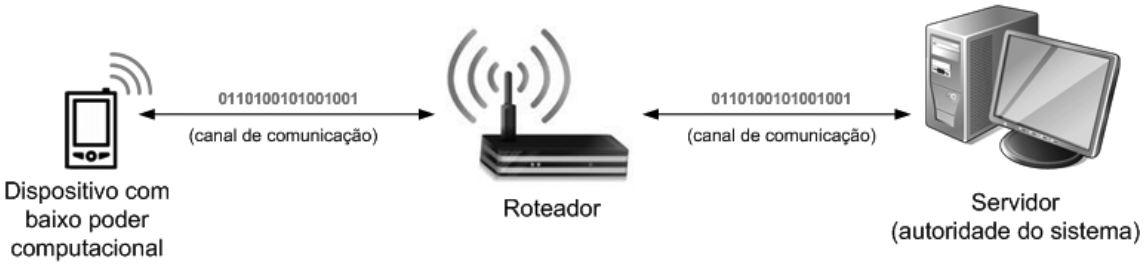

Figura 1. Cenário de comunicação entre o dispositivo e o servidor (autoridade do sistema).

- **Smartphone (MM1)**: Motorola Milestone 1, processador ARM Cortex-A8 (*single-core*) de 600 MHz, memória RAM de 256 MB, e sistema operacional Android 2.2.
- **Tablet (GN7)**: Asus Google Nexus 7, processador ARM Cortex-A9 (*quad-core*) de 1.2 GHz, memória RAM de 1 GB, e sistema operacional Android 4.2.
- **Single-board computer (RPi)**: Raspberry Pi, processador ARM1176JZF-S (*single-core*) de 700 MHz, memória RAM de 256 MB, e sistema operacional Debian "Wheezy" (armhf).

5.1. Ambiente de desenvolvimento

Para viabilizar a implementação dos protocolos, foi utilizada uma biblioteca criptográfica com suporte para operações sobre curvas elípticas e emparelhamentos bilineares. Escolhemos a *RELIC-toolkit* (versão 0.3.1) [Aranha and Gouvêa 2009], construída em linguagens C e Assembly, e com ênfase em eficiência. Tal biblioteca permite configurações flexíveis para níveis de segurança específicos e escolhas algorítmicas, além de suporte para diversas arquiteturas de processadores, tais como x86, ARM, AVR e MSP.

As implementações dos protocolos estudados foram escritas em linguagem C, integrando-se de forma transparente à biblioteca *RELIC-toolkit*. A comunicação entre os dispositivos e o servidor é realizada através de *sockets* em C. Dentre as vantagens da utilização de *sockets* estão a fácil integração dos códigos construídos com as funções de comunicação em rede, comunicação TCP/IP direta, e facilidade em transmitir alguns tipos de dados complexos da *RELIC-toolkit*, já que não existem funções nativas que convertam tais tipos para *strings* e vice-versa.

O processo de compilação do código construído para os dispositivos com sistema operacional Android se deu com auxilio do Android *Native Developement Kit* (NDK). Trata-se de um *framework* para desenvolvimento de software, que permite escrever e compilar código nativo em linguagens C e C++. Em geral, aplicativos para Android são construídos em linguagem de programação Java, e funcionam sobre a máquina virtual Dalvik (uma implementação da máquina virtual Java otimizada para Android). O código nativo produzido com auxílio do NDK pode ser executado a partir de aplicativos construídos em Java, através do *Java Native Interface* (JNI), como também pode ser executado de forma autônoma, utilizando-se um *shell* de comando.

Os experimentos deste trabalho foram executados a partir do *shell* de comando. Todavia, as chaves compartilhadas resultantes da execução dos protocolos de acordo de

chave analisados podem ser facilmente convertidas para *strings* e enviadas para um aplicativo Android construído em Java, através de JNI. Desta forma, aplicativos do mundo real podem se beneficiar do código nativo gerado, que faz uso de uma das bibliotecas criptográficas mais eficientes disponíveis atualmente.

5.2. Curvas Elípticas e Custo das operações

A biblioteca *RELIC-toolkit* dispõe de um conjunto de curvas elípticas pré-definidas. Tais curvas se dividem em dois grupos: curvas elípticas sobre corpos binários e curvas elípticas sobre corpos primos. Nos interessam as curvas conhecidas como *pairing-friendly*, que permitem a construção de sistemas criptográficos que fazem uso de emparelhamentos bilineares [Freeman et al. 2010].

Para a realização dos experimentos, foram utilizadas curvas elípticas conhecidas como BN [Barreto and Naehrig 2006], definidas sobre corpos de ordem prima. Estas curvas possuem grau de mergulho 12, permitindo assim o uso de corpos algébricos menores, contudo mantendo elevado nível de segurança. Por esse mesmo motivo, tais curvas são mais eficientes quando comparadas com curvas elípticas sobre corpos binários.

Escolhemos três curvas com diferentes níveis de segurança:

- **BN158**: segurança de 78 bits.
- **BN256**: segurança de 128 bits.
- **BN638**: segurança de 192 bits.

Para mensurar o desempenho de cada um desses níveis de segurança, medimos o tempo de execução das principais operações envolvidas nos protocolos analisados. Curvas BN trabalham com emparelhamento assimétrico, portanto foi escolhido o emparelhamento *Optimal Ate*. Também faz-se necessário mensurar operações em $\mathbb{G}_1$ e em $\mathbb{G}_2$, já que $\mathbb{G}_1 \neq \mathbb{G}_2$. A Tabela 1 mostra o desempenho das operações em diferentes níveis de segurança, para cada um dos dispositivos analisados. Foram utilizadas 50 amostras de tempo para o cálculo de cada um dos valores apresentados.

Para obter melhores resultados, utilizamos a biblioteca GMP (GNU *Multiple Precision Library*) como *backend* aritmético da *RELIC-toolkit*. Desta forma, o tempo médio das operações diminuiu na razão de 3 a 8 vezes, dependendo do dispositivo e do nível de segurança. Pode-se observar pela Tabela 1 que as operações que demandam mais tempo são emparelhamento, exponenciação em $\mathbb{G}_T$ e multiplicação escalar nos grupos $\mathbb{G}_2$ e $\mathbb{G}_1$, nesta ordem. Utilizando-se da propriedade de bilinearidade em emparelhamentos, é possível converter operações de exponenciação no grupo $\mathbb{G}_T$ em multiplicações nos grupos $\mathbb{G}_1$ ou $\mathbb{G}_2$, resultando em um ganho de eficiência maior que 50% para este tipo de conversão.

Em nossos testes, também avaliamos o desempenho das operações em curvas binárias (caso com emparelhamento simétrico). Entretanto, os tempos obtidos foram muito mais altos, o que tornam essas curvas pouco atraentes para os tipos de dispositivos analisados.

Tabela 1. Intervalo de confiança (95%) dos tempos de execução (em milissegundos) das operações para diferentes níveis de segurança.

		Nível de Segurança da Curva Elíptica		
Operação	**Dispositivo**	**78 bits**	**128 bits**	**192 bits**
Emparelhamento *Optimal Ate*	PC	[5.47, 5.48]	[13.07, 13.08]	[90.06, 90.08]
	RPi	[42.22, 42.33]	[94.66, 94.83]	[624.75, 625.04]
	MM1	[49.34, 50.67]	[110.67, 112.57]	[684.97, 695.99]
	GN7	[31.37, 31.92]	[68.14, 68.19]	[408.1, 409.0]
Exponenciação em $\mathbb{G}_T$	PC	[5.27, 5.35]	[12.59, 12.74]	[88.07, 88.74]
	RPi	[40.88, 41.54]	[91.04, 92.17]	[612.32, 616.96]
	MM1	[48.15, 49.64]	[107.83, 110.25]	[673.39, 685.02]
	GN7	[30.49, 31.25]	[66.42, 67.27]	[401.86, 405.3]
Multiplicação em $\mathbb{G}_T$	PC	[0.04, 0.04]	[0.05, 0.05]	[0.15, 0.15]
	RPi	[0.27, 0.27]	[0.37, 0.38]	[1.02, 1.06]
	MM1	[0.27, 0.27]	[0.36, 0.64]	[0.98, 1.24]
	GN7	[0.18, 0.19]	[0.25, 0.26]	[0.66, 0.66]
Multiplicação em $\mathbb{G}_1$	PC	[0.56, 0.56]	[1.4, 1.4]	[10.01, 10.06]
	RPi	[4.5, 4.54]	[10.14, 10.2]	[70.32, 70.65]
	MM1	[4.91, 5.46]	[11.03, 11.75]	[77.9, 79.63]
	GN7	[3.5, 3.57]	[7.55, 7.6]	[45.6, 46.47]
Multiplicação em $\mathbb{G}_2$	PC	[2.35, 2.38]	[6.07, 6.13]	[47.31, 47.62]
	RPi	[16.98, 17.22]	[41.25, 41.7]	[316.97, 319.02]
	MM1	[19.78, 20.78]	[48.25, 51.73]	[348.66, 354.51]
	GN7	[12.77, 13.06]	[29.57, 29.89]	[206.17, 207.75]
Adição em $\mathbb{G}_1$	PC	[0.004, 0.004]	[0.01, 0.01]	[0.02, 0.02]
	RPi	[0.03, 0.03]	[0.04, 0.05]	[0.12, 0.12]
	MM1	[0.04, 0.05]	[0.05, 0.05]	[0.09, 0.32]
	GN7	[0.02, 0.02]	[0.03, 0.03]	[0.07, 0.08]
Adição em $\mathbb{G}_2$	PC	[0.01, 0.01]	[0.01, 0.01]	[0.04, 0.04]
	RPi	[0.07, 0.07]	[0.1, 0.1]	[0.26, 0.28]
	MM1	[0.07, 0.07]	[0.1, 0.11]	[0.23, 0.41]
	GN7	[0.05, 0.05]	[0.06, 0.06]	[0.17, 0.17]

5.3. Adaptação eficiente para emparelhamento assimétrico

Outra forma de implementar protocolos mais eficientes é distribuir os valores dos grupos $\mathbb{G}_1$ e $\mathbb{G}_2$ de forma que o dispositivo com menor poder computacional (cliente) realize mais cálculos no grupo $\mathbb{G}_1$, enquanto o servidor (geralmente com maior poder computacional) realiza mais cálculos no grupo $\mathbb{G}_2$. Como visto na Tabela 1, operações em $\mathbb{G}_2$ chegam a custar em média quatro vezes mais que operações em $\mathbb{G}_1$. Por isso devemos maximizar a quantidade de multiplicações em $\mathbb{G}_1$ no lado do cliente.

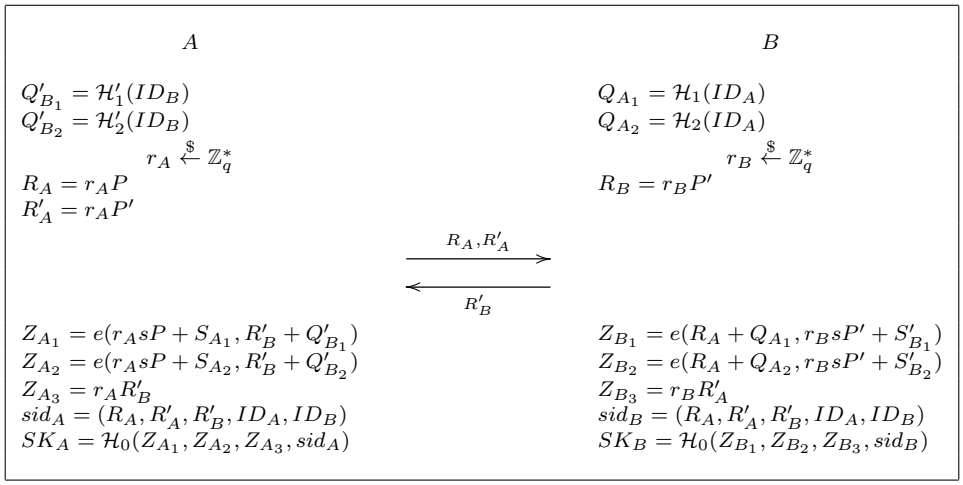

Figura 2. Protocolo [Huang and Cao 2009] assimétrico - caso 1

Apresentamos, como exemplo, duas adaptações do protocolo Huang-Cao [Huang and Cao 2009] para emparelhamento assimétrico. A Figura 2 ilustra o caso menos eficiente para o cliente (representado pelo usuário B), e a Figura 3 o caso mais eficiente. Considere $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ um emparelhamento assimétrico admissível, P um ponto gerador de $\mathbb{G}_1$, P' um ponto gerador de $\mathbb{G}_2$, $\langle sP, sP' \rangle$ chaves públicas da autoridade do sistema, e $\mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}'_1, \mathcal{H}'_2$ funções de *hash* tais que: $\mathcal{H}_0 : \{0,1\}^* \rightarrow \{0,1\}^k$, $\mathcal{H}_1, \mathcal{H}_2 : \{0,1\}^* \rightarrow \mathbb{G}_1$ e $\mathcal{H}'_1, \mathcal{H}'_2 : \{0,1\}^* \rightarrow \mathbb{G}_2$.

Note que no cenário da Figura 2, a partir da geração dos valores temporários, o servidor (representado pelo usuário A) terá que executar duas multiplicações em $\mathbb{G}_1$ e duas multiplicações em $\mathbb{G}_2$. Já o cliente, com menor poder computacional, executará três multiplicações em $\mathbb{G}_2$, que são mais custosas.

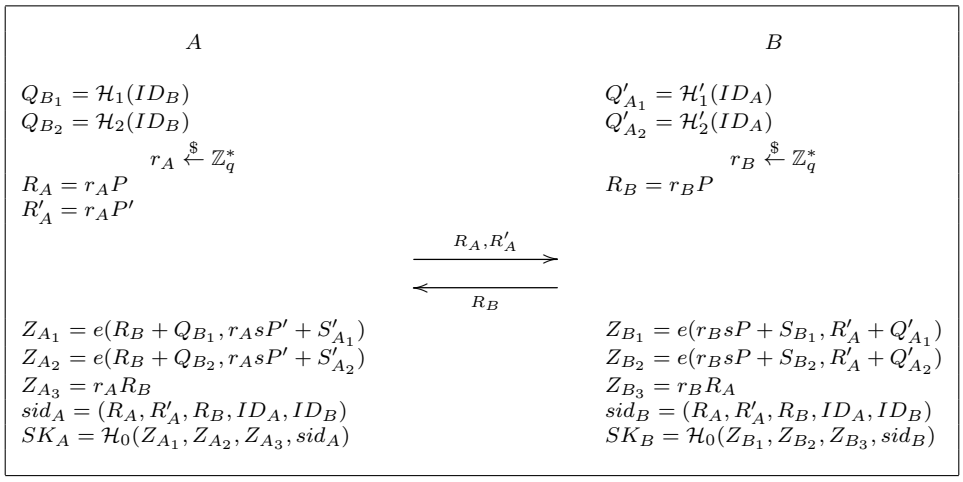

Figura 3. Protocolo [Huang and Cao 2009] assimétrico - caso 2

O cenário da Figura 3 é o mais vantajoso para o cliente, que necessita calcular apenas três multiplicações em $\mathbb{G}_1$. Já o servidor calcula duas multiplicações em $\mathbb{G}_1$ e duas multiplicações em $\mathbb{G}_2$. Adaptações similares foram produzidas para todos os protocolos estudados, com o objetivo de tornar o código utilizado pelos dispositivos (no papel de clientes) o mais eficiente possível.

5.4. Comparação de desempenho entre os protocolos analisados

Apresentamos nas Tabelas 2 e 3 comparações entre os tempos de execução dos protocolos estudados, nos três dispositivos analisados, em diferentes níveis de segurança. A contagem de tempo iniciou-se no momento da escolha dos valores temporários, e encerrou-se após o cálculo da chave de sessão.

Para cada protocolo, em cada dispositivo, foram obtidas um total de 100 amostras de tempo, sendo 50 para o caso sem pré-computação, e 50 para o caso com pré-computação. Cada amostra corresponde a uma sessão completa, ou seja, escolhem-se novos valores temporários e calcula-se uma nova chave de sessão, diferente das anteriores.

O cenário sem pré-computação, apresentado na Tabela 2, representa a execução dos protocolos da maneira como são definidos. Já o cenário com pré-computação, exibido

na Tabela 3, é considerado quando um usuário A se comunica frequentemente com um usuário B. Assim, alguns valores referentes a B não precisam ser recalculados a cada nova sessão.

Tabela 2. Intervalo de confiança (95%) dos tempos de execução (em milissegundos) dos protocolos analisados, para o cenário sem pré-computação em diferentes níveis de segurança.

		Nível de Segurança da Curva Elíptica		
Protocolos *ID-Based*	**Dispositivo**	**78 bits**	**128 bits**	**192 bits**
HC-BDH	RPi	[103.6, 104.6]	[218.4, 218.9]	[1430.9, 1431.9]
	MM1	[111.8, 113.0]	[227.4, 229.0]	[1422.1, 1426.9]
	GN7	[48.0, 50.0]	[93.0, 94.9]	[526.7, 531.2]
HLZ-GBDH	RPi	[94.7, 94.9]	[206.2, 206.6]	[1318.8, 1319.8]
	MM1	[104.2, 107.0]	[213.9, 216.0]	[1339.1, 1372.7]
	GN7	[44.6, 46.9]	[86.5, 88.7]	[482.5, 485.6]
CC-BDH	RPi	[80.7, 81.1]	[175.1, 175.8]	[1013.7, 1016.0]
	MM1	[88.5, 91.4]	[184.0, 186.1]	[1060.9, 1131.9]
	GN7	[36.4, 38.8]	[74.9, 76.1]	[381.2, 382.9]
NCLH-BDH	RPi	[344.4, 346.6]	[780.9, 784.1]	[5037.5, 5051.7]
	MM1	[374.5, 379.4]	[782.0, 788.0]	[4992.5, 5011.8]
	GN7	[144.3, 146.5]	[319.4, 321.7]	[1819.0, 1825.2]
NCLH-GBDH	RPi	[177.1, 178.3]	[402.4, 404.1]	[2583.2, 2590.8]
	MM1	[194.7, 198.7]	[403.8, 406.0]	[2552.5, 2563.9]
	GN7	[75.2, 77.3]	[161.5, 165.3]	[934.1, 940.3]
NCL-BDH	RPi	[94.8, 95.0]	[217.2, 217.6]	[1364.3, 1364.9]
	MM1	[105.2, 107.8]	[214.8, 216.3]	[1346.0, 1352.4]
	GN7	[44.8, 48.2]	[86.0, 86.9]	[496.4, 499.6]
Protocolos *Certificateless*	**Dispositivo**	**78 bits**	**128 bits**	**192 bits**
LBG-BDH	RPi	[464.2, 465.7]	[1034.0, 1034.9]	[6815.3, 6818.9]
	MM1	[489.0, 501.6]	[1065.0, 1072.7]	[6558.5, 6573.1]
	GN7	[198.9, 200.0]	[421.6, 424.2]	[2419.3, 2424.3]
LBG-GBDH	RPi	[243.1, 243.6]	[563.0, 563.5]	[3634.6, 3637.7]
	MM1	[265.5, 268.9]	[577.1, 584.0]	[3522.9, 3539.3]
	GN7	[110.1, 113.8]	[230.4, 233.7]	[1299.8, 1303.1]
GOT-BDH	RPi	[396.6, 397.2]	[837.5, 838.1]	[5635.5, 5638.6]
	MM1	[397.2, 399.8]	[873.8, 886.6]	[5403.3, 5496.4]
	GN7	[162.9, 164.4]	[347.3, 349.9]	[1972.8, 1978.2]
GOT-GBDH	RPi	[207.6, 208.0]	[468.6, 469.1]	[2975.1, 2977.4]
	MM1	[222.3, 227.4]	[475.5, 482.8]	[2913.6, 2938.1]
	GN7	[92.4, 95.4]	[187.3, 188.3]	[1080.7, 1084.4]
GNT3-BDH	RPi	[284.3, 284.7]	[640.2, 640.9]	[4306.0, 4309.2]
	MM1	[310.9, 316.4]	[673.7, 676.8]	[4131.5, 4145.2]
	GN7	[126.6, 128.6]	[270.2, 272.9]	[1523.4, 1528.7]
GNT1-GBDH	RPi	[215.2, 215.6]	[451.2, 451.8]	[3009.1, 3012.6]
	MM1	[222.6, 224.9]	[475.7, 480.3]	[2912.8, 2928.3]
	GN7	[91.9, 93.4]	[187.9, 189.1]	[1079.6, 1085.8]
GNT2-GBDH	RPi	[160.7, 161.1]	[357.0, 357.6]	[2366.7, 2368.8]
	MM1	[179.7, 189.4]	[378.2, 381.2]	[2307.3, 2315.1]
	GN7	[76.8, 78.9]	[149.5, 151.1]	[858.4, 862.9]
GNT4-BDH	RPi	[209.5, 210.0]	[460.1, 460.7]	[3009.3, 3012.1]
	MM1	[224.1, 230.2]	[477.0, 483.5]	[2918.2, 2924.3]
	GN7	[93.1, 95.4]	[186.5, 189.2]	[1080.8, 1086.9]

Fica evidente na Tabela 2 que os protocolos cuja segurança se baseia na dificuldade do problema BDH (mais seguro) consomem mais tempo do que os que se baseiam no problema GBDH (menos seguro). Protocolos seguros sob modelos de segurança mais fracos também possuem desempenho melhor. No caso do modelo baseado em identidade, os protocolos HLZ-GBDH e CC-BDH alcançaram desempenho médio 6,5% e 22,2% melhor que o protocolo HC-BDH, respectivamente. Note que o primeiro está no mesmo modelo de segurança que HC-BDH (eCK), todavia com problema computacional mais fraco. Já o segundo possui o mesmo problema computacional, mas demonstrado seguro

sob o modelo de segurança CK (mais fraco que eCK).

Os protocolos NCLH (BDH e GBDH) e NCL-BDH são uma subclasse à parte de protocolos de acordo de chave, pois possuem uma propriedade conhecida como modo *escrow*, e por isto são recomendados apenas em cenários muito específicos. O protocolo NCL-BDH apresentou desempenho médio 72,5% e 46,7% melhor que os protocolos NCLH-BDH e NCLH-GBDH, nesta ordem. Consequentemente, seu uso é recomendado, por ser seguro sob um problema computacional equivalente ou mais difícil.

No modelo sem certificado, de acordo com a Tabela 2, o protocolo GNT3-BDH apresentou desempenho médio 23,2% melhor que GOT-BDH. Já este último teve desempenho médio 17,6% melhor que LBG-BDH. Tal resultado era previsto, uma vez que GNT3-BDH é uma melhoria do protocolo GOT-BDH, que por sua vez é uma melhoria do protocolo LBG-BDH (sendo todos eles demonstrados seguros no modelo de segurança LBG). Já o protocolo GNT4-BDH mostrou ser 28,8% mais eficiente, em média, que o protocolo GNT3-BDH, o que também era esperado visto que GNT4-BDH é demonstrado seguro sob o modelo de segurança SJ^+ (mais fraco).

Ainda no modelo sem certificado, o protocolo GOT-GBDH apresentou desempenho médio 16,8% melhor que LGB-GBDH, ambos seguros no modelo de segurança LBG. Entretanto, o protocolo GNT1-GBDH foi 17% mais eficiente que LBG-GBDH e ficou tecnicamente empatado com GOT-GBDH. Assim, recomenda-se o uso do protocolo GNT1-GBDH, uma vez que é seguro sob o modelo de segurança Mal-LBG (mais forte que LBG). A única exceção é o protocolo GNT2-GBDH, seguro no modelo Mal-SJ^+ (mais fraco que Mal-LBG), e por este motivo mostrou-se, em média, 20,5% mais eficiente que o protocolo GNT1-GBDH.

Na Tabela 3, foram omitidos os protocolos HC-BDH, HLZ-GBDH, CC-BDH e NCL-BDH, todos do modelo baseado em identidade, uma vez que eles não possuem valores que possam ser armazenados entre diferentes sessões. Isto ocorre porque esses protocolos são mais simples, e em geral, todos os valores dependem dos segredos temporários, que mudam a cada nova sessão. Dentre os protocolos baseados em identidade estudados, os únicos que se beneficiaram do cenário com pré-computação são NCLH-BDH e NCLH-GBDH. Mesmo assim, o protocolo NCL-BDH continua mais vantajoso, e seu uso é recomendado (caso se opte por um protocolo com modo *escrow*).

Já os protocolos no modelo sem certificado analisados sofreram uma melhora significativa em desempenho no cenário com pré-computação. Em geral, há um ganho médio de desempenho entre 45% à 70%, quando comparamos os tempos dos mesmos protocolos com os da Tabela 2. Vale ressaltar que o protocolo GOT-BDH teve desempenho praticamente igual ao do GNT3-BDH, sendo este último mais recomendado pelo menor uso de memória RAM, de acordo com a Tabela 4. O protocolo GNT4-BDH também teve desempenho praticamente igual aos protocolos GNT3-BDH e GOT-BDH, e seu uso neste cenário não é recomendado, visto que é demonstrado seguro sob modelo de segurança mais fraco que os outros dois.

Ainda de acordo com a Tabela 3, o protocolo GOT-GBDH foi, em média 37% mais eficiente que GNT1-GBDH, uma diferença considerável quando comparamos com o cenário sem pré-computação. Neste caso, recomenda-se o uso de GOT-GBDH quando não houver preocupação com ataques originados pela autoridade do sistema.

Tabela 3. Intervalo de confiança (95%) dos tempos de execução (em milissegundos) dos protocolos analisados, para o cenário com pré-computação em diferentes níveis de segurança.

		Nível de Segurança da Curva Elíptica		
Protocolos *ID-Based*	**Dispositivo**	**78 bits**	**128 bits**	**192 bits**
NCLH-BDH	RPi	[174.4, 176.4]	[396.8, 400.2]	[2568.9, 2578.9]
	MM1	[194.6, 197.3]	[400.2, 405.8]	[2556.8, 2588.5]
	GN7	[76.9, 78.0]	[162.1, 163.7]	[940.3, 947.2]
NCLH-GBDH	RPi	[92.3, 93.3]	[209.1, 210.8]	[1343.7, 1348.9]
	MM1	[105.1, 106.6]	[212.1, 217.6]	[1432.7, 1455.7]
	GN7	[44.5, 46.1]	[88.0, 89.3]	[501.5, 506.2]
Protocolos *Certificateless*	**Dispositivo**	**78 bits**	**128 bits**	**192 bits**
LBG-BDH	RPi	[197.1, 197.5]	[437.2, 438.0]	[2870.3, 2871.9]
	MM1	[211.1, 231.9]	[456.6, 460.8]	[2779.9, 2802.3]
	GN7	[88.7, 89.5]	[179.2, 180.1]	[1029.3, 1032.5]
LBG-GBDH	RPi	[107.7, 108.1]	[247.9, 248.5]	[1597.3, 1598.9]
	MM1	[126.6, 132.8]	[263.2, 266.1]	[1559.2, 1568.0]
	GN7	[54.6, 56.6]	[102.4, 103.0]	[583.3, 585.8]
GOT-BDH	RPi	[115.5, 115.8]	[244.1, 244.6]	[1629.7, 1631.9]
	MM1	[126.4, 128.3]	[267.2, 285.2]	[1559.7, 1565.4]
	GN7	[53.8, 55.1]	[104.3, 105.5]	[583.4, 586.3]
GOT-GBDH	RPi	[67.5, 67.9]	[151.1, 151.6]	[958.3, 959.8]
	MM1	[81.2, 83.3]	[162.7, 165.2]	[972.0, 1009.1]
	GN7	[38.2, 40.4]	[65.7, 66.5]	[362.6, 364.4]
GNT3-BDH	RPi	[108.2, 108.5]	[242.2, 242.8]	[1612.5, 1614.4]
	MM1	[124.4, 126.7]	[261.2, 265.4]	[1564.5, 1573.7]
	GN7	[53.8, 55.2]	[103.8, 104.6]	[585.7, 588.7]
GNT1-GBDH	RPi	[115.6, 116.1]	[241.7, 242.2]	[1602.1, 1603.8]
	MM1	[127.0, 129.5]	[260.1, 262.8]	[1560.3, 1565.6]
	GN7	[55.0, 56.7]	[103.8, 105.1]	[586.1, 588.4]
GNT2-GBDH	RPi	[66.5, 66.9]	[146.7, 147.7]	[964.1, 965.7]
	MM1	[81.6, 83.1]	[160.8, 163.3]	[977.4, 1011.7]
	GN7	[37.5, 39.5]	[66.0, 67.2]	[361.3, 363.8]
GNT4-BDH	RPi	[112.3, 112.7]	[245.4, 245.9]	[1600.8, 1602.4]
	MM1	[127.6, 130.2]	[260.1, 263.4]	[1561.9, 1569.7]
	GN7	[56.1, 58.5]	[103.7, 104.9]	[586.9, 589.2]

Nas Tabelas 4 e 5 apresentamos a quantidade de memória utilizada por cada protocolo e o comprimento das mensagens trocadas, respectivamente. A primeira é baseada na quantidade mínima de variáveis para implementar o protocolo e o tamanho do tipo de dado de cada uma delas. A segunda se baseia no comprimento dos tipos de dados utilizados para troca de mensagens.

Tabela 4. Uso de memória (em *bytes*) dos protocolos analisados em diferentes níveis de segurança.

	Nível de Segurança da Curva Elíptica		
Protocolos *ID-Based*	**78 bits**	**128 bits**	**192 bits**
HC-BDH	1624	2536	6184
HLZ-GBDH	1376	2132	5156
CC-BDH	1460	2228	5300
NCLH-BDH	2812	4444	10972
NCLH-GBDH	1904	2996	7364
NCL-BDH	1496	2336	5696
Protocolos *Certificateless*	**78 bits**	**128 bits**	**192 bits**
LBG-BDH	3324	5232	12864
LBG-GBDH	2416	3784	9256
GOT-BDH	3448	5428	13348
GOT-GBDH	2540	3980	9740
GNT1-GBDH	2728	4276	10468
GNT3-BDH	2968	4660	11428
GNT2-GBDH	2060	3212	7820
GNT4-BDH	2488	3892	9508

Tabela 5. Comprimento das mensagens trocadas (em *bytes*) dos protocolos analisados em diferentes níveis de segurança.

	Nível de Segurança da Curva Elíptica		
Protocolos *ID-Based*	**78 bits**	**128 bits**	**192 bits**
HC-BDH	252	396	972
HLZ-GBDH	188	296	728
CC-BDH	316	496	1216
NCLH-BDH	252	396	972
NCLH-GBDH	252	396	972
NCL-BDH	188	296	728
Protocolos *Certificateless*	**78 bits**	**128 bits**	**192 bits**
LBG-BDH	504	792	1944
LBG-GBDH	504	792	1944
GOT-BDH	504	792	1944
GOT-GBDH	504	792	1944
GNT1-GBDH	504	792	1944
GNT3-BDH	504	792	1944
GNT2-GBDH	504	792	1944
GNT4-BDH	504	792	1944

6. Conclusão

Neste trabalho apresentamos implementações eficientes de protocolos de acordo de chave em dispositivos de poder computacional restrito nos modelos de criptografia baseado em identidade (*ID-Based*) e sem certificado (*Certificateless*). Mostramos que é viável utilizar protocolos relativamente complexos, com níveis elevados de segurança, e tempos de execução praticáveis. Ademais, nossos testes simularam situações do mundo real, incluindo comunicação em uma rede TCP/IP, e uso de bibliotecas criptográficas e matemáticas otimizadas, construídas principalmente em linguagens C e Assembly.

Os resultados expostos apontam sugestões para possíveis melhorias de tempo. Dentre elas, sugerimos a substituição de operações de exponenciação no grupo $\mathbb{G}_T$ por multiplicações no grupo $\mathbb{G}$. Outra recomendação é a redistribuição das operações de multiplicação nos grupos $\mathbb{G}_1$ e $\mathbb{G}_2$, quando utilizado emparelhamento assimétrico, de modo que o dispositivo com menor poder computacional efetue o máximo de multiplicações no grupo $\mathbb{G}_1$, para ganho de eficiência.

Referências

Al-Riyami, S. S. and Paterson, K. G. (2003). Certificateless public key cryptography. In *Advances in Cryptology - ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, Taipei, Taiwan. Springer.

Aranha, D. F. and Gouvêa, C. P. L. (2009). RELIC is an Efficient LIbrary for Cryptography. `http://code.google.com/p/relic-toolkit/`.

Barreto, P. S. and Naehrig, M. (2006). Pairing-friendly elliptic curves of prime order. In *Selected areas in cryptography*, pages 319–331. Springer.

Boneh, D. and Franklin, M. K. (2001). Identity-based encryption from the weil pairing. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO 2001, pages 213–229, London, UK. Springer-Verlag.

Boneh, D. and Franklin, M. K. (2003). Identity-based encryption from the Weil pairing. *SIAM J. of Computing*, 32(3):586–615. extended abstract in Crypto'01.

Canetti, R. and Krawczyk, H. (2001). Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology-EUROCRYPT 2001*, pages 453–474. Springer.

Chow, S. S. and Choo, K.-K. R. (2007). Strongly-secure identity-based key agreement and anonymous extension. In *Information Security*, pages 203–220. Springer.

Freeman, D., Scott, M., and Teske, E. (2010). A taxonomy of pairing-friendly elliptic curves. *Journal of Cryptology*, 23(2):224–280.

Goya, D., Nakamura, D., and Terada, R. (2011). Acordo de chave seguro contra autoridade mal intencionada. *SBSeg 2011, XI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 265–278.

Goya, D., Okida, C., and Terada, R. (2010). A two-party certificateless authenticated key agreement protocol. *SBSeg 2010, X Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 443–446.

Goya, D. H. (2011). *Criptografia de chave pública sem certificado*. PhD thesis, Instituto de Matemática e Estatística da Universidade de São Paulo.

Hoffstein, J., Pipher, J., and Silverman, J. (2008). *An Introduction to Mathematical Cryptography*. Springer Publishing Company, Incorporated, 1 edition.

Hu, X., Liu, W., and Zhang, J. (2009). An efficient id-based authenticated key exchange protocol. In *Information Engineering, 2009. ICIE'09. WASE International Conference on*, volume 2, pages 229–233. IEEE.

Huang, H. and Cao, Z. (2009). An id-based authenticated key exchange protocol based on bilinear diffie-hellman problem. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 333–342. ACM.

Krawczyk, H. (2005). Hmqv: A high-performance secure diffie-hellman protocol. In *Advances in Cryptology–CRYPTO 2005*, pages 546–566. Springer.

LaMacchia, B., Lauter, K., and Mityagin, A. (2007). Stronger security of authenticated key exchange. In *Provable Security*, pages 1–16. Springer.

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

Ni, L., Chen, G., and Li, J. (2012). Escrowable identity-based authenticated key agreement protocol with strong security. *Computers & Mathematics with Applications*.

Ni, L., Chen, G., Li, J., and Hao, Y. (2011). Strongly secure identity-based authenticated key agreement protocols. *Computers & Electrical Engineering*, 37(2):205–217.

Okamoto, T. and Pointcheval, D. (2001). The gap-problems: A new class of problems for the security of cryptographic schemes. In *Proceedings of PKC 2001, volume 1992 of LNCS*, pages 104–118. Springer-Verlag.

Shamir, A. (1984). Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, volume LNCS 196, pages 47–53. Springer-Verlag New York, Inc.

