ANDRÉ GUSTAVO DEGRAF UCHÔA

DHA: UM ESQUEMA DE ACORDO DE CHAVES BASEADO EM MATRIZES PARA O PROTOCOLO DIFFIE-HELLMAN

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Curitiba
2007

ANDRÉ GUSTAVO DEGRAF UCHÔA

DHA: UM ESQUEMA DE ACORDO DE CHAVES BASEADO EM MATRIZES PARA O PROTOCOLO DIFFIE-HELLMAN

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Área de Concentração: Ciência da Computação

Orientador: Marcelo Eduardo Pellenz
Co-orientador: Altair Olivo Santin

Curitiba
2007

Uchôa, André Gustavo Degraf
DHA: UM ESQUEMA DE ACORDO DE CHAVES BASEADO EM MATRIZES PARA O PROTOCOLO DIFFIE-HELLMAN. Curitiba, 2007.

Dissertação - Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática.

1. Criptografia Assimétrica 2. Protocolo de Acordo de Chaves 3. Matrizes I.Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e Tecnologia. Programa de Pós-Graduação em Informática II - t

Dedico a minha dissertação de mestrado a Deus todo poderoso Pai Eterno e para os meus pais Sirlei e Raimundo, pela ajuda financeira, ajuda moral, ajuda física, entre outras.

Agradecimentos

Agradeço aos meus pais pela paciência, compreensão, respeito, dedicação, conselhos, ajuda financeira, ajuda moral e por fazer com que eu pudesse realizar este mestrado.

Agradeço aos meus colegas de mestrado Tiago, Luis Renato, Jaime, Teigão, Euclides, Edson, entre outros colegas.

Agradeço a PUC-PR por ter me concedido a bolsa Marcelino Champagnat que fez com que eu conseguisse fazer este mestrado.

Agradeço aos meus orientadores pela imensa ajuda na jornada do mestrado, pela paciência e compreensão nos momentos mais críticos da minha pesquisa. Também agradeço pelas referências bibliográficas que meus orientadores me passaram, as quais foram de suma importância na realização desta dissertação.

Agradeço ao professor doutor Carlos Alberto Maziero pelas aulas de Sistemas Operacionais e o uso da Linguagem C em ambiente UNIX-like e pela imensa ajuda no ensino do uso do editor de texto LaTex que foi utilizado para fazer esta dissertação.

Também agradeço a equipe de professores da academia Via Aventura por terem me ajudado a cuidar da saúde corporal através do esporte, http://www.viaaventura.com.br.

Agradeço em especial ao software livre LINUX, por ter tornado possível o desenvolvimento do DHA3 e das análises matemáticas. E também devo lembrar dos criadores do compilador para Linguagem C o GNU GCC e as bibliotecas GNU GMP e GNU MPFR. Com Relação a MPFR gostaria de agradecer a um dos criadores por ter me auxiliado durante minha pesquisa que é o PHd Vicent Lefevre.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

Γ	Gerador base do DHA3
υ	Vetor base do DHA3
P	Espaço de Texto Claro
C	Espaço de Texto Cifrado
K	Espaço da Chave
D	Espaço de Dados Decifrados
ε	Família de Funções Criptográficas
e	Chave Pública
$\in$	Pertence à
d	Chave Privada
c	Contante arbitrária
k	Contante arbitrária
p	Número primo
α	Gerador DH, ElGamal, MTI
Z_p^*	Numeros inteiros positivos não nulos - Campo Finito
x	Expoente arbitrário
y	Expoente arbitrário
a	Expoente arbitrário
k_i	Parâmetro CRTDH
D_p	Parâmetro CRTDH
gcd	Greatest Commom Divisor
GK	Chave do Grupo do CRTDH
U	U novo membro
pw	Senha Secreta no SPAKE
G	Grupo finito cíclico
M, N	Números aleatórios pertencentes a G
$h()$	Função de hash
S	Servidor Confiável

$pw1$	Senha no S-3PAKE
$pw2$	Senha no S-3PAKE
$X\|\|Y$	Concatenação entre X e Y
GL_n	Grupo de matrizes sobre campo finito
r	Expoente arbitrário
η	Magnitude do número primo
t	Bit mais significante
I_n	Matriz identidade
$A_{n\times n}$	Variável matricial
$\oplus$	Operação binária XOR

Lista de Abreviações

DLP	*Discrete Logarithm Problems*
DH	*Diffie-Hellman*
DES	*Data Encryption Standard*
NIST	*National Institute of Standards and Technology*
SSL	*Secure Socket Layer*
3DES	*Triplo DES*
CDC	*Cifra Decifra Cifra*
IDEA	*International Data Encryption Algorithm*
PGP	*Pretty Good Privacy*
NSA	*National Security Agency*
MIT	*Instituto Tecnológico de Massachusetts*
PKP	*Public Key Partners*
BSGS	*Baby-Step Giant-Step*
TTP	*Trusted Third Party*
SGC	*Secure Group of Communication*
CRT	*Chinese Remainder Theorem*
LCM	*Least Commom Multiple*
SPAKE	*Simple Password-based Encrypted Key Exchange Protocol*
S-3PAKE	*Simple Three-Party Key Exchange Protocol*
NFS	*Number Field Sieve*
IC	*Index Calculus*
AES	*Advanced Encryption Standard*

Resumo

Neste trabalho é proposto um protocolo sem autenticação das partes para acordo de chaves criptográficas de forma segura, baseado no protocolo Diffie-Hellman. Atualmente os principais protocolos para acordo de chaves como Diffie-Hellman e ElGamal trabalham com operações exponenciais modulares e a complexidade de tais protocolos está baseada na dificuldade em resolver problemas de logaritmos discretos DLPs. O protocolo proposto nesta dissertação se baseia no protocolo Diffie-Hellman, mas utiliza como gerador base uma matriz e um vetor, fazendo com que a complexidade de quebra de tal protocolo seja baseada na solução de um DLP e com complexidade de solução muito maior que a apresentada pelo Diffie-Hellman. Com base na análise de complexidade do protocolo DHA1 e do DHA2, melhorias foram propostas para aumentar a dificuldade de quebra do protocolo DHA1 gerando assim, uma terceira proposta chamada DHA3. O uso de matrizes, vetores e operações sobre $GL(n)$ é responsável por dificultar a criptoanálise. Devido ao uso de matrizes e vetores o método proposto pode trabalhar com números primos de menor magnitude nas operações modulares, diminuindo assim o custo computacional sem no entanto diminuir a complexidade de quebra. Um protótipo foi desenvolvido como prova de conceito para mostrar a viabilidade da proposta. Neste protótipo constatou-se que o DHA3 levou cinco vezes menos tempo de processamento que o DH clássico, para gerar a chave de sessão e ainda transmitiu pelo menos metade de informação que o DH.

Palavras-chave: Criptografia Assimétrica, Protocolo de Acordo de Chaves, Matrizes

Abstract

In this work is proposed an annonimous cryptographic key agreement protocol by secure way based on Diffie-Hellman protocol. Actually key agreement protocols like Diffie-Hellman, ElGamal and other work with modular exponentiation operations and the complexity of such protocols are based on apparent difficult to deal DLP (*Discrete Logarithm Problems*). The proposed protocol in this master degree thesis is based on Diffie-Hellman, but uses as base generator a matrix and a vector, doing with the complexity to solve such protocol be based on DLP solution, but much more complexity than was shown by Diffie-Hellman. Based on ccomplexity analysis of DHA1 and DHA2, improvements were proposed to difficult the criptanalisys of DHA1 generating thus, a third proposal called DHA3 Protocol. The usage of matrices, vectors and operations under $GL(n)$ are responsable by difficulting the criptanalisys. Due to using matrices and vectors the proposed method can work with prime numbers with small precision under modular operaions, decreesing thus the computational cost without decreesing the complexity to break. One prototype was developed as concept proof to show the proposal viability. In this prototype it was constacted that DHA3 taken about five less computer processing time than DH to generate the session key and also transmited at least half information than DH.

Keywords: Assymetric Cryptography, Key Agreement Protocols, Matrices

Capítulo 1

Introdução

As primeiras redes de computadores eram baseadas em cabeamento, de forma que dispositivos móveis como Handhelds e Notebooks, tinham que acessar tais redes ou por acesso discado usando a linha telefônica ou ainda em ambientes corporativos através dos "cabos de rede". Em poucos anos, tornou-se possível a comunicação dos computadores através de redes baseadas em tecnologias de comunicação sem fio que veio possibilitar uma maior mobilidade aos dispositivos móveis acessarem a rede sem precisar para isto estar fisicamente conectados a algum tipo de cabeamento. Contudo, com a implementação de redes sem fio, problemas de segurança antes não existentes nas redes cabeadas tradicionais tornaram-se possíveis em meios sem fio devido ao meio de acesso ser o ar e a forma como os dipositivos são distribuídos em redes sem fio.

1.1 Contextualização

Atualmente, as técnicas criptográficas assumem um papel importante ao proteger os dados sendo transmitidos entre os nós de uma rede sem fio. Existem diversas tecnologias que proporcionam a comunicação sem fio como: redes sem fio baseadas no padrão IEEE 802.11, redes de sensores, redes baseadas no padrão Bluetooth, entre outras. O grande desafio apresentado por essas redes é qual algoritmo criptográfico deverá ser utilizado e uma vez escolhido, como distribuir/compartilhar a chave criptográfica utilizada pelo algoritmo. Pois, devido ao meio sem fio utilizar o ar como meio de transporte das informações as chaves criptográficas compartilhadas/distribuídas podem ser capturadas por terceiros que podem ser elementos maliciosos à rede sem fio implantada.

Um modo específico de operação de redes sem fio, o modo *Ad-hoc*, são redes sem fio e sem infra-estrutura prévia, por exemplo: um grupo de computadores (nós da rede) comunicando-se via rede sem fio sem ter acesso a um servidor que realize o interfaceamento

entre a rede sem fio e algum tipo de provedor de acesso a Internet. Este modo de operação impõe alguns desafios para a área de seguraça como: realizar a autenticação dos nós sem a existência de um servidor centralizado; implementar-se um canal de comunicação seguro, sem ter a garantia de que os nós pertencentes à rede são confiáveis, como realizar a distribuição de chaves criptográficas em tais ambientes sem que terceiros maliciosos venham a capturar tais chaves; a utilização de algoritmos criptográficos que não causem alto custo computacional devido ao fato que geralmente os nós de uma rede no modo *Ad-hoc* possuem limitação de consumo de bateria.

Com o aumento do uso de dispositivos com conexão sem fio impôs-se um desafio adicional nos protocolos criptográficos, devido à facilidade para capturar pacotes transmitidos entre dois nós sobre um canal de rádio. Em redes sem fio ad-hoc um nó pode também se tornar um roteador para encaminhar pacotes de um nó para outro nó qualquer. Além disso, a implementação de algoritmos criptográficos baseados em chave pública, numa rede sem fio ad-hoc não é uma tarefa simples de ser executada, uma vez que tal abordagem necessita de um servidor centralizado para gerenciar as chaves públicas utilizadas pelos membros de tal rede [1].

Outra técnica para prover segurança em redes sem fio ad-hoc é através de algoritmos criptográficos baseados em chave simétrica, nos quais a chave secreta deverá ser compartilhada entre os nós participantes. O problema de tal técnica é a distribuição das chaves, porque um nó deve enviar a chave para outro, sem cifragem. Isto constitui uma fraqueza, uma vez que a chave pode ser facilmente interceptada num meio sem fio. Um modo de evitar tal problema é realizar a pré-distribuição da chave. Neste caso, os fabricantes devem previamente instalar as chaves secretas pré-definidas nos dispositivos de hardware [2]. A pré-distribuição de chaves restringe a possibilidade da escolha do tipo do algoritmo de criptografia a ser usado, como também o tamanho da chave. Adicionalmente, se um intruso tiver acesso físico ao dispositivo móvel, as chaves pré-definidas podem ser comprometidas.

1.2 Definição do Problema

Os protocolos criptográficos de acordo de chaves são baseados em sistemas simétricos e/ou assimétricos, sendo que existem ainda os que são “sem autenticação das partes” como é o caso do Diffie-Hellman-Merkle [3]. Tal protocolo é classificado como não tendo autenticação das partes devido ao fato de que não há qualquer tipo de autenticação das elementos participantes da comunicação, ele apenas garante que a chave compartilhada será entregue com segurança ponto a ponto. Além disso, utiliza operações exponenciais

modulares, que geralmente geram um custo computacional alto, mesmo que para isso sejam utilizados algoritmos que aumentem a velocidade para calcular as exponenciações [3].

1.3 Motivação

No Capítulo 3 desta dissertação serão discutidos alguns trabalhos relacionados. O Diffie - Hellman é um protocolo sem autenticação das partes para acordo de chaves que utiliza operações exponenciais modulares que necessita da geração de um número primo com uma alta precisão, em torno de 1024 a 2048 bits, para aumentar o tempo de criptoanálise por DLP. As operações de exponenciação que trabalham com números primos de grande magnitude, geram um alto custo computacional. O protocolo proposto trabalha com primos de magnitude menor do que a dos protocolos tradicionais [3] [4]. O MTI é uma variante do Diffie-Hellman que tenta aumentar a complexidade adicionando dois parâmetros extras (chaves públicas). No entanto, utiliza o mesmo gerador base α que permite a criptoanálise por DLP. Algumas variantes do MTI necessitam do cálculo da inversa multiplicativa o que acarreta um custo computacional alto. O protocolo $\gamma \equiv \alpha^a \beta^b$ [5] tentou aumentar a complexidade da criptoanálise definindo que seria necessário resolver dois DLPs um para cada gerador, α e β. No entanto tal proposição não era consistente como demonstrado por [6]. Além disso, o protocolo acarreta no envio e processamento do dobro de parâmetros comparado ao DH. O SPAKE [7] utiliza uma senha com um universo de busca limitado, pois são caracteres disponíveis no teclado de um computador e esta senha é previamente compartilhada entre os nós que desejam estabelecer uma chave de seção. Além de utilizar uma senha pré-estabelecida entre os nós, é também baseado em criptoanálise por DLP, pois utiliza o mesmo gerador base, sendo susceptível a ataque de "passphrase", não trazendo um aumento significativo para a complexidade de criptoanálise. O protocolo S-3PAKE [7] propõe uma melhoria no protocolo SPAKE ao dizer que não necessita de um servidor de chaves públicas, no entanto, tal protocolo requer um servidor confiável que autentique os nós participantes da comunicação. Também na proposta do S-3PAKE cada nó deve possuir uma senha compartilhada com o servidor confiável, assim pode-se dizer que a melhoria supostamente definida pelos autores não é de fato uma melhoria, pois simplesmente retirou-se a autoridade de um servidor de chaves públicas e passou para um servidor confiável. Além disso, este protocolo realiza muitas trocas, o que é indesejável em ambientes com limitações nos canais de comunicação ou mesmo limitação no consumo das baterias em dispositivos com baixo poder computacional. O protocolo Matrix Rings [8] foi o primeiro protocolo a propor o uso de matrizes como

gerador base para o aumentar a complexidade do DH, mas devido ao tipo da matriz geradora ser não singular (admite inversa) tal método pode ser resolvido e sua complexidade pode ser reduzida conforme descrito em [9].

1.4 Objetivos

1.4.1 Objetivos Gerais

- Estudar os sistemas criptográficos simétricos e assimétricos;
- Estudar e analisar os protocolos de acordo de chaves criptográficas;
- Analisar os algoritmos baseados em problemas de logaritmos discretos (DLP);
- Estudar matrizes sobre campos finitos;
- Comparar e testar os algoritmos que realizam exponenciação;

1.4.2 Objetivos Específicos

- Criar um protocolo que não requer um servidor de chaves públicas para realizar as trocas;
- Produzir um protocolo que transmita menos informação que o Diffie-Hellman;
- Estabelecer um protocolo que diminua o processamento computacional para gerar os parâmetros pré-compartilhados;
- Desenvolver um protocolo que possua complexidade à criptoanalise em tempo exponencial;
- Implementar um protocolo que utilize matrizes nos seus procedimentos com o intuito de dificultar a criptoanálise;

1.5 Contribuição

Nesta dissertação, é proposto um protocolo sem autenticação das partes para o acordo de chaves criptográficas baseado no protocolo clássico DH (*Diffie-Hellman*), sendo que a complexidade para resolver tal protocolo está baseada na aparente intratabilidade para resolver Logarítmos Discretos (DLP). Foram implementadas duas propostas o DHA1

e o DHA2, mas ambas estas propostas podiam ser reduzidas ao caso do DH, com o objetivo de realmente criar um protocolo mais robusto que o DH. Foi então implementado o DHA3 que é a junção do DHA1 e do DHA2. A modificação realizada no DH feita pelo DHA3 é a utilização de uma matriz quadrada Γ e um vetor v como geradores base em vez do α utilizado pelo DH. Devido a utilização deste par de elementos: uma matriz e um vetor como geradores base, o protocolo DHA3 faz com que seja mais complexo o tempo de resolução comparado ao DH e permitindo apenas que os algoritmos para resolver DLPs com complexidade de resolução $O(\sqrt{p})$ possam encontrar uma solução. Outros métodos ainda necessitam um estudo mais profundo, além disso, o DHA3 leva 5 vezes menos tempo de processamento que o DH e transmite pelo menos 50% menos informação que o DH a cada troca de mensagens.

O protocolo apresentado nesta dissertação pode ser utilizado tanto em redes no modo *Ad-hoc* como em outras configurações de redes. Este protocolo visa aumentar a complexidade do DH, visa utilizar números primos menores, e visa transmitir menos informação que o DH, de forma a ser mais eficiente que outros protocolos de acordo de chaves até então criados e utilizados. Visa também garantir a entrega segura da chave compartilhada e produzir menos custo computacional para os dispositivos ao utilizá-lo.

1.6 Organização

Esta dissertação está organizada da seguinte forma: no Capítulo 2, são apresentados a fundamentação matemática e alguns algoritmos para resolver DLPs; no Capítulo 3, são apresentados os trabalhos relacionados ao DHA3; no Capítulo 4, é apresentado o protocolo proposto, o DHA3, bem como é também feita análise de quebra e a complexidade de algoritmos baseados em DLP que serviram de base para criar-se o DHA3, e por fim; o Capítulo 5, a conclusão desta dissertação.

Capítulo 2

Fundamentos de Criptografia

Neste capítulo serão abordados os principais fundamentos que serão requeridos para compreensão do protocolo proposto. Os tópicos serão: conceitos básicos sobre segurança e criptografia, DLPs, algoritmos para resolver DLPs, matrizes sobre GL_n.

2.1 Conceitos Básicos Sobre Segurança

Ao longo dos anos, vários métodos e protocolos foram desenvolvidos para proteger uma "informação", sendo que esta informação era documento físico como: papel, carta, contratos. Frequentemente os objetivos de segurança da informação não podem ser alcançados através de métodos matemáticos e protocolos apenas, mas requerem técnicas e procedimentos para poder alcançar o objetivo desejado. Com o avanço das tecnologias de transporte e envio de informação, alcançar segurança da informação tornou-se mais complexo devido ao meio utilizado; falsificar um documento como um contrato, é mais complexo do que simplesmente realizar uma cópia digital de um arquivo que apresentará as mesmas características do original, tornando mais complexo o serviço dos peritos na detecção da falsificação. Alcançar segurança da informação numa sociedade eletrônica requer um conjunto vasto de técnicas e habilidades legais. Nunca se tem uma garantia absoluta de que o sistema está 100% seguro, mas um meio técnico de prover segurança a um sistema é através da criptografia.

A palavra Criptografia etimologicamente vem da palavra grega $\kappa\rho\upsilon\pi\tau\varsigma$ (oculto) e graphos (escrita). Esta área de estudo mais abrangente engloba as disciplinas da Criptografia e da Criptoanálise. Criptologia é o estudo de Códigos e Cifras (não necessariamente secretos). Mensagens ocultas que não são nem codificadas nem cifradas, são simplesmente ocultas. Um código é um sistema pré-estabelecido de substituição de palavras ou de parágrafos. Um idioma estrangeiro, por exemplo, é como um código secreto onde cada

palavra possui uma equivalente na outra língua. Assim, “água” em português equivale a “wassar” em alemão ou “water” em inglês. A maioria dos códigos funciona como se fosse um dicionário, onde apresentam-se as equivalências entre os termos. Já a palavra cifra vem do hebraico *saphar*, que significa “dar número”. A maioria das cifragens são freqüentemente baseadas em técnicas de sistemas numéricos.

A Criptoanálise é o estudo de como burlar os algoritmos criptográficos existentes, a ponto de conseguir-se obter o texto claro. Uma das técnicas clássicas é o ataque por força bruta, onde a técnica é baseada em tentativas exaustivas de obter apenas o texto claro de posse apenas do texto cifrado. A criptografia é o estudo de técnicas matemáticas relacionadas aos aspectos de segurança da informação tais como confidencialidade, integridade dos dados, autenticação das entidades e autenticação dos dados.

Propriedades Criptográficas:

- Confidencialidade: é um serviço usado para manter o conteúdo da informação protegido de todos, autorizando o acesso apenas ao que possui o direito de acessar tal informação. Existe um vasto grupo de aplicações para prover confidencialidade, desde métodos físicos de proteção, a algoritmos matemáticos que fazem com que a informação torne-se ininteligível [3].

- Integridade: é o serviço que garante que uma alteração não autorizada seja realizada sobre uma dada informação. Para garantir a integridade, o método deverá ser capaz de detectar uma manipulação nos dados por partes não autorizadas. A manipulação dos dados inclui inserção, apagamento e substituição dos dados por outra informação qualquer [3].

- Autenticação: é um serviço relacionado à identificação. Duas entidades que inicializem uma comunicação deverão identificar-se um com o outro. Em suma, a autenticação é utilizada para identificar quem é quem numa comunicação segura, evitando assim que um terceiro faça o papel de outra entidade.

- Não Repudiação: é um método que previne uma entidade de negar acordos ou ações realizadas previamente. Por exemplo, uma entidade pode dizer que acessou um determinado site e fez algumas alterações no mesmo, e em seguida negar tais ações. Para isso que serve a não repudiação, garantir que uma entidade não possa negar que não fez determinadas atitudes ou ações [3].

O objetivo principal da criptografia é aplicar estas quatro propriedades fundamentais tanto na teoria como na prática. A criptografia é um meio pelo qual pode-se evitar e/ou detectar falsificação e outros tipos de atividade maliciosas.

2.1.1 Sistemas Criptográficos Clássicos

A noção de sistema criptográfico é formalmente definida a seguir. Um sistema criptográfico é uma quíntupla $(P, C, K, \varepsilon, D)$ tal que [10]:

P, C, K são conjuntos finitos, onde P é o espaço de texto claro; C é o espaço de texto cifrado; K é o espaço da chave. Elementos de P são referidos como texto claro, elementos de C são referidos como textos cifrados e elementos de D são referidos como espaço de dados decifrados. Uma mensagem é um conjunto de símbolos de texto claro.

$\varepsilon = \{E_k | k \in K\}$ é uma família de funções $E_k : P \rightarrow C$ que são usadas para cifragem, e $D = \{D_k \mid k \in K\}$ é a família de funções $D_k : C \rightarrow P$ que são usadas para decifragem. Para cada chave $(e \in K)$ existe uma chave $(d \in K)$ tal que para cada $p \in P : D_d(E_e(p)) = p$.

Um sistema criptográfico é chamado simétrico (ou de "chave privada") se $d = e$, ou se d pode ao menos "facilmente"ser obtida a partir de e. Um sistema criptográfico é chamado assimétrico (ou de "chave pública") se $d \neq e$, sendo na prática "impossível"obter d a partir de e. Assim, d é a chave privada, e é a chave pública. Às vezes, diferentes tamanhos de chaves são utilizados para cifragem e para decifragem, o que resulta numa leve modificação da definição feita acima.

2.1.2 Definições

Algoritmos de Criptografia em Blocos: Estes algoritmos dividem o texto plano em blocos de tamanho fixo e estes blocos são manipulados por operações matemáticas e/ou lógicas um por vez, até gerar o texto cifrado, e também são conhecidos por não possuírem memória.

Algoritmos de Criptografia em Fluxo (Stream Ciphers): São algoritmos que cifram os dados conforme estes são recebidos. Não contém uma fase preparatória de divisão do texto claro em "blocos", cada caractere é cifrado um por vez e o caractere cifrado pode ou não possuir correlação com o caractere anterior que já foi cifrado.

2.1.3 Criptoanálise

A criptoanálise para analisar a força de um sistema criptográfico depende de vários fatores: dificuldade de descobrir a chave (chaves criptográficas com tamanho maior, em bytes, dificultam mais na tentativa de quebra do sistema criptográfico, porém causarão o aumento do custo computacional para cifrar a informação ou texto plano), dificuldade de se quebrar o código mesmo já conhecendo a mensagem cifrada, entre outros. A seguir

alguns dos tipos de criptoanálises realizadas em sistemas criptográficos:

- Ataque do texto cifrado (*cyphertext-only*): Quando se tem à disposição uma grande quantidade de mensagens cifradas, mas desconhece-se o texto claro e as chaves utilizadas. O objetivo neste tipo de ataque é conseguir obter o texto claro (obter as chaves utilizadas para gerar o texto cifrado).

- Ataque do texto claro conhecido (*known-plaintext*): Quando se tem à disposição uma grande quantidade de mensagens cifradas e também textos claros correspondentes aos textos cifrados. Neste tipo de ataque, tenta-se obter as chaves utilizadas na cifragem dos textos claros, sendo assim, uma forma de analisar a robustez do algoritmo no quesito chaves.

- Ataque adaptativo do texto escolhido (*adaptative-choosen-plaintext*): Neste método se tem à disposição porções de textos claros e cifrados, e a cada passo analisamse as partes para tentar extrair alguma informação; após isto, pode se tentar obter novos textos claros e cifrados, permitindo que pequenas porções de textos cifrados e/ou claros sirvam para o ataque. O objetivo é deduzir as chaves utilizadas. Alguns sistemas criptográficos como o RSA [3] [4], são muito vulneráveis a este tipo de ataque.

- Ataque do texto cifrado escolhido (*choosen-cyphertext*): Neste tipo de ataque, não tem-se apenas uma gama de textos claros e seus equivalentes cifrados, mas pode-se produzir uma mensagem cifrada específica a ser decifrada, e compará-la com o texto claro inserido no algoritmo criptográfico em análise. É usado quando se tem uma "caixa-preta" que faz a decifragem automática. Sua tarefa é encontrar as chaves utilizadas.

- Ataque da chave escolhida (*choosen-key*): Neste tipo de ataque pode-se testar o sistema com diversas chaves diferentes, ou pode-se convencer diversos usuários legítimos do sistema a utilizarem chaves pré-determinadas. A finalidade imediata é decifrar as mensagens cifradas com essas chaves.

- Ataque de força bruta (*brute force attack*): Resume o que todos os outros ataques realizaram, é um ataque de caráter geral, mas é utilizado apenas para busca do texto claro.

2.1.4 Sistemas Criptográficos Simétricos

Sistema simétrico ou de chave secreta é um sistema criptográfico que utiliza uma mesma chave (ou segredo) para cifrar e decifrar a mensagem. Esta forma também é conhecida como Criptografia por Chave Secreta ou Chave Única ou Criptografia Simétrica. A chave pode ser uma palavra, uma frase ou uma seqüência aleatória de números e deverá ser conhecida por ambas as partes comunicantes (se uma mensagem é enviada a um determinado destinatário, este deverá possuir a chave para poder recuperar a informação por ele recebida). Seu tamanho é medido em bits e, via de regra, quanto maior a chave, mais segura será a informação cifrada. Sua geração, transmissão e armazenamento, são denominados Gerência de Chaves.

O principal desafio deste sistema é garantir que ninguém mais conheça a chave além das partes comunicantes. Para tanto, as partes comunicantes devem trocar pessoalmente ou possuir um sistema de transmissão confiável capaz de garantir a confiabilidade das chaves transmitidas. Pois, qualquer um que venha, de alguma forma, a ter conhecimento desta chave, pode mais tarde ler, modificar ou forjar mensagens cifradas ou autenticadas que utilizem aquela chave. Devido ao fato de ser necessário compartilhar a chave secreta, os sistemas de criptografia por chave simétrica apresentam dificuldades em garantir plena segurança, especialmente em ambientes distribuídos com um grande número de usuários.

Tais métodos são mais utilizados em ambientes com dispositivos limitados computacionalmente, onde o transmissor e o receptor se preparam antecipadamente para o uso da chave secreta. Outro fato é de que ambos conhecem a chave secreta, com isso pode-se permitir o repúdio, dizer que determinada mensagem não foi enviada pelo participante 'A' da comunicação, mesmo ele tendo enviado tal mensagem. A grande vantagem da criptografia de chave secreta é que ela é muito rápida, podendo ser facilmente implementada em hardware, principalmente em equipamentos de baixo poder computacional. Tal método pode ser bem utilizado em aplicações em que o próprio usuário cifra e decifra os dados, como por exemplo, o caso de cifrar os dados do disco rígido para que terceiros não tenham acesso às informações ali constantes. A Figura 2.1 mostra em diagrama de blocos o funcionamento de um sistema criptográfico simétrico.

A seguir são citados alguns dos algoritmos criptográficos simétricos mais conhecidos descritos em [3] [4]:

- Crypt: Sistema de cifragem do UNIX, que utiliza uma chave de tamanho variável. Alguns programas podem automaticamente decifrar arquivos cifrados sem necessariamente conhecer a chave e/ou mesmo o texto plano. O Crypt não é seguro. O sistema crypt é utilizado pelo UNIX para cifrar senhas.

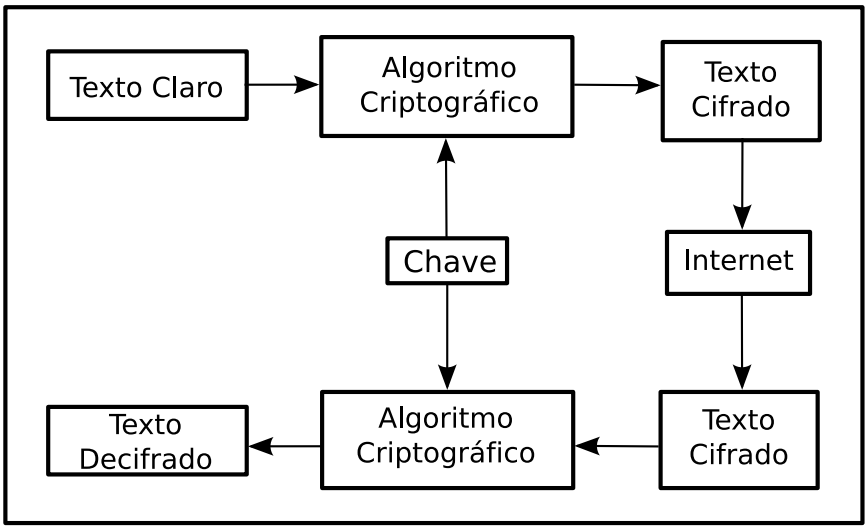

Figura 2.1: Sistema Criptográfico Simétrico.

- DES (*Data Encryption Standard*): É um padrão de cifragem de dados desenvolvido na década de 70, pelo National Bureau of Standards and Technology (atualmente conhecido como: NIST (*National Institute of Standards and Technology*)) e a IBM. O DES utiliza uma chave de 56 bits e opera em blocos de 64 bits. Foi projetado inicialmente para ser utilizado em componentes de hardware. Nos dias atuais é utilizado na Internet em conexões Web segura, o SSL (*Secure Socket Layer*) utiliza o DES dentre outros algoritmos criptográficos. Ele é um algoritmo seguro para uma grande quantidade de aplicações, entretanto, em aplicações altamente confidenciais, o DES não deve ser usado, pois existe o risco de violação. Existe ainda o 3DES (*Triplo DES*) que opera com três chaves no esquema CDC (*Cifra Decifra Cifra*), onde cada uma destas operações é feita com uma chave diferente. Todas estas chaves são de 56 bits e trabalham também com blocos de texto claro de 64 bits.

- IDEA (*International Data Encryption Algorithm*): foi desenvolvido, na década de 80 por Xuejia Lai e James Massey da ASCOM Tech AG da Suíça, em Zurique. Ele foi projetado para ser de fácil implementação, além de ser resistente à muitas formas de criptoanálise. O seu método baseia-se na utilização de uma chave de 128 bits, onde blocos de texto de 64 bits da mensagem de entrada são alterados em uma seqüência de interações, produzindo blocos cifrados de informação na saída do processo de cifragem. Sua chave de 128 bits é o suficiente para resistir à maioria dos ataques. IDEA é usado pelo popular programa PGP, para cifrar arquivos e e-mails. Entretanto, a vasta utilização do IDEA é impedida por uma série de patentes de software da ASCOM Tech. A ASCOM Tech concedeu apenas, o uso gratuito em implementações do PGP (*Pretty Good Privacy*) fora dos Estados Unidos da América,

mas sendo necessário verificar os termos e suas licenças.

- RC2 e RC4: Mais velozes do que os sistemas DES e 3DES, esses códigos podem se tornar mais seguros com o simples aumento do tamanho das chaves. O RC2 pode substituir perfeitamente o DES com a vantagem de ser 2 vezes mais rápido, em termos de desempenho. O algoritmo RC4 é 10 vezes mais rápido.

- RC5: Um sistema de cifragem desenvolvido por Ronald Rivest [4] e publicado em 1994. O RC5 permite definir o tamanho da chave, o tamanho dos blocos de dados, e o número de cifragens realizadas.

- Skipjack: Sistema criptográfico desenvolvido pela NSA (*National Security Agency*) utiliza chaves de 80 bits para cifrar blocos de 64 bits de informação. Até 1998 era classificado como secreto, seu projeto inicial foi concebido em meados de 1980. O Skipjack foi uma tentativa de substituir o DES devido às suas vulnerabilidades. Contudo um dia após o código do Skipjack ter sido aberto ao público (estudiosos, criptógrafos), foram encontrados métodos para realizar ataques nas tabelas de rotação 16 das 32 utilizadas pelo mesmo e em questões de meses foram encontradas 31 das 32 tabelas de rotação. Por esses e outros motivos realizar segurança através da obscuridade não traz benefício algum, uma vez que os estudiosos e criptógrafos podem contribuir de forma significativa para a melhoria dos algoritmos sendo repostos pelas grandes agências de segurança.

2.1.5 Sistemas Criptográficos de Chave Pública

No presente sistema, cada usuário possui um par de chaves, uma que é a Chave Pública e outra a Chave Privada. Enquanto a chave pública tem seu conhecimento divulgado, a chave privada necessita ficar armazenada seguramente. Assim, a necessidade das partes comunicantes de trocar informações sigilosas é eliminada porque todas as comunicações irão envolver somente a chave pública, não sendo preciso trocar as chaves privadas por nenhuma das partes. Paralelamente, este sistema não exige credibilidade dos meios de transmissão envolvidos. A única exigência deste sistema é que a chave pública esteja associada aos seus usuários após ser feita a autenticação da mesma. Qualquer um dos possuidores da chave pública pode usá-la para enviar uma mensagem. Contudo, a mesma mensagem só pode ser decifrada mediante o uso da chave privada a qual é de uso restrito de seu proprietário.

Em criptografia de chave pública as implementações mais conhecidas são o RSA e o sistema PGP. Em 1977, Rivest, Shamir e Adelman [3] desenvolveram o RSA e publicaram

o algoritmo de cifragem apesar da oposição do governo norte americano, que considera a criptografia um assunto de segurança nacional. Mais tarde a patente do RSA é dada ao MIT (*Instituto Tecnológico de Massachusetts*) que logo a cede a um grupo denominado PKP (*Public Key Partners*).

O programador Phil Zimmermann, em 1991, autoriza a publicação em boletins eletrônicos e grupos de notícias de um programa por ele desenvolvido e batizado como Pretty Good Privacy ou PGP [11]. O PGP que tem como base os algoritmos do RSA publicados em 1978. Quando Zimmermann publicou o PGP se viu em problemas com o Departamento de Estado Norte Americano que abriu uma investigação para determinar se ele havia violado as restrições de exportação de criptografia ao autorizar a divulgação do código fonte do PGP na Internet. Apesar do mesmo ter se comprometido em não divulgar seu desenvolvimento, diversos programadores em várias partes do mundo continuaram adiante, portando-o para várias plataformas e assegurando sua expansão.

No método de cifragem por chave pública resolve o problema de transmitir uma mensagem totalmente segura através de um canal inseguro. O receptor da mensagem cria duas chaves que são relacionadas entre si, uma pública e uma privada. A chave pública pode e deve ser distribuída livremente. Quem envia a mensagem tem que utilizar a chave publica do receptor para cifrá-la. Uma vez cifrada, esta mensagem só pode ser decifrada pela chave do receptor.

Apesar dos benefícios no quesito segurança, onde os sistemas de criptografia assimétrica permitem trafegar informações sobre um canal inseguro, tais sistemas apresentam uma desvantagem que é o fator velocidade de cifragem. Enquanto um algoritmo de cifragem simétrica pode processar milhões de bytes de texto claro, os algoritmos de cifragem assimétrica podem apenas cifrar na casa de milhares de bytes de texto claro o que é indesejável para aplicações em sistemas embutidos. A Figura 2.2 mostra em diagrama de blocos o funcionamento de um sistema criptográfico assimétrico.

A seguir são descritos alguns dos sistemas criptográficos assimétricos mais utilizados descritos em [3] [4]:

- ***Diffie-Hellman*** - Um protocolo para trocas de chaves criptográficas entre partes comunicantes, não é um método de cifragem e decifragem, mas um método desenvolvido para o compartilhamento de chaves de Sessão sobre um canal de comunicação público. Ao realizar o protocolo, as duas partes concordam em alguns valores numéricos *Domain Parameters*, e cada parte gerará ao final dos procedimentos a mesma chave. Cada parte pode então calcular uma outra chave de Sessão que não pode facilmente ser derivada por um ataque sem conhecer ambas as tro-

cas realizadas entre as partes. Existem várias versões deste protocolo, envolvendo um diferente número de partes e múltiplas transformações matemáticas. As chaves podem apresentar vários tamanhos dependendo do tipo de implementação. Chaves maiores geralmente são mais seguras, devido a dificuldade em resolver DLPs.

- ***RSA*** - Um sistema criptográfico desenvolvido por três pessoas, (Ronald Rivest, Shamir e Leonard Adleman do MIT), e por este motivo é chamado de RSA. O RSA pode ser usado para cifragem de informações e assinaturas digitais. As chaves podem apresentar vários tamanhos dependendo do tipo de implementação.

- ***ElGamal*** - Outro protocolo de acordo de chaves que se baseia em operações exponenciais e em funções modulares como o DH, é o ElGamal. Ele é usado principalmente para cifragem e assinaturas digitais de uma maneira similar ao algoritmo Diffie-Hellman.

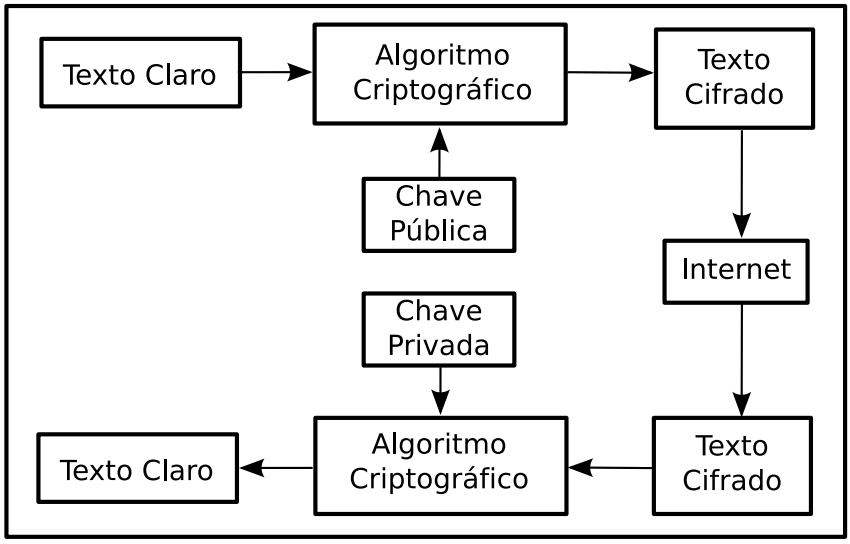

Figura 2.2: Sistema Criptográfico Assimétrico.

2.1.6 Funções de Hash Criptográficas

As funções de hash criptográficas são utilizadas para realizar uma espécie de "impressão digital" de um determinado documento de forma a garantir teoricamente que dois documentos distintos não resultem no mesmo *hash*. As funções de hash podem ser aplicadas para garantir a integridade dos dados bem como serem usadas para realizar a autenticação das mensagens utlizando-se para técnicas de assinaturas digitais. As funções de hash transformam uma dada entrada (em bits) de qualquer tamanho fixo numa saída de tamanho fixo idependentemente de qual seja esta entrada. A Figura 2.3 descreve o

funcionamento do simplificado de uma função de hash onde os resultados estão na base hexadecimal, e para este exemplo foi utilizado o hash MD5.

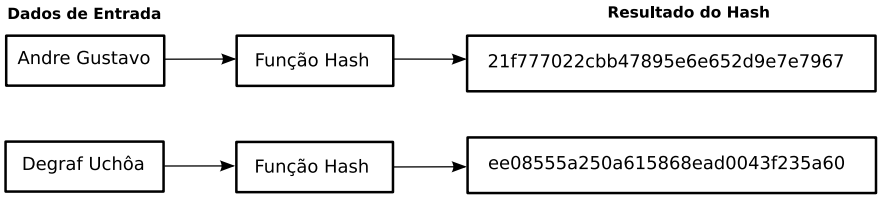

Figura 2.3: Funcionamento função de Hash

Algumas definições e propriedades de algumas funções de hash são descritas abaixo, para funções de hash h com entradas x, x' e saídas y, y' [3]:

1. *resistência a pré-imagem* - para essencialmente todas as saídas pré-especificadas, é computacionalmente inviável encontrar qualquer entrada dado os hashes, por exemplo, encontrar qualquer pré-imagem x' tal que $h(x') = y$ quando dado qualquer y para o qual a correspondente entrada não é conhecida.

2. *resistência a segunda pré-imagem* - é computacionalmente inviável encontrar qualquer segunda entrada que tenha a mesma saída com qualquer entrada especificada, por exemplo, dado x, encontrar uma segunda pré-imagem $x' \neq x$ tal que $h(x) = h(x')$.

3. *resistência a colisão* - é computacionalmente inviável encontrar qualquer duas entradas distintas x, x' que o hash tenha a mesma saída, por exemplo, tal que $h(x) = h(x')$.

Apesar das funções de hash primarem por essas propriedades já foi demostrado que a grande maioria delas apresentou alguns casos de colisão (duas entradas diferentes geram o mesmo hash). Para o escopo do protocolo proposto nesta dissertação tais problemas não afetariam a segurança do protocolo DHA3, uma vez que a utilização de tais funções de hash são atualizadas apenas para a derivação das chaves de sessão geradas pelo protocolo.

Principais aplicações das funções de hash [3]:

1. *Integridade dos dados* - garante que os dados não foram alterados, modificados ou mesmo substituídos.

2. *Derivação de chaves* - para computar sequências de novas chaves a *priori* calculadas, se a chave sendo utilizada for comprometida esta não comprometerá as chaves utilizadas nas transações anteriores.

3. *Gerador de números pseudo-aleatórios* - em alguns casos muito específicos pode se usar funções de hash para gerar números pseudo-aleatórios, mas não é considerado um método seguro, pois uma vez determinada a semente geradora dos hashes todo sistema estará comprometido.

A Tabela 2.1 descreve as funções de hash que são baseadas em cifradores de bloco (*block ciphers*) e serão discutidas a sua utilização no Capítulo 4 desta dissertação. Na Tabela 2.1 é descrito o tamanho da saída em bits e o número de operações de hash necessárias a serem realizadas para gerar uma colisão (dados teóricos uma vez que foram encontrados meios de causar colisões em menos operações).

Tabela 2.1: Funções de hash-especificações

Função hash	Saída em bits	Operações de hash
MD5	128	2^{64}
RIPEMD-128	128	2^{64}
SHA-1, RIPEMD-160	160	2^{80}

2.2 DLP

A segurança de muitas técnicas criptográficas depende da intratabilidade de problemas de logaritmo discreto (DLPs). Uma lista parcial de tais métodos inclui Diffie-Hellman e suas variantes, ElGamal e suas variantes. Um problema de logaritmo discreto pode ser definido como sendo: Dado um α um primo p e um $\beta = \alpha^a \mod p$ encontrar a; sendo $a, \alpha, \beta \in Z_p^*$ [3]. Supondo $p = 97$. Então Z_{97}^* é um grupo cíclico de ordem $n = 96$. Um gerador de Z_{97}^* pode ser $\alpha = 5$. Sendo, $5^{32} \equiv 35 \pmod{97}$, $\log_5 35 = 32$ em Z_{97}^*.

Resolver o DLP num grupo cíclico G de ordem n é em ensência calcular o isomorfismo entre G e Z_n. Sendo asssim qualquer dois grupos cíclicos de mesma ordem são isomórficos (que é, eles possuem a mesma estrutura contudo os elementos podem ser escritos em diferentes representações), um algoritmo eficiente para calcular um grupo não necessariamente implica ser eficiente para calcular em outro grupo. Para ver sito, considere que cada grupo cíclico de ordem n é isomórfico ao grupo cíclico aditivo Z_n, isto é, o conjunto de inteiros $\{0, 1, 2, \cdots, n-1\}$ onde a operação do grupo é adição módulo n. Além disso, o problema de logaritmo discreto neste grupo, nomeadamente, é o problema de encontrar um inteiro x tal que $ax \equiv b (\mod n)$ dado $a, b \in Z_n$. Primeiro notar que não existe uma solução x se $d = gcd(a, n)$ não divide b. Entretanto, se d divide b o algoritmo Euclidiano extendido [3] pode ser usado para encontrar inteiros s e t tais que $as + nt = d$. Multiplicando ambos os lados desta equação pelo inteiro b/d têm - se

$a(sb/d) + n(tb/d) = b$. Reduzindo esta equação módulo n têm - se $a(sb/d) \equiv b(\text{mod } n)$ e daí $x = (sb/d) \ (\text{mod } n)$ é a solução desejada [4].

Os algorítmos para resolver DLP podem ser categorizados desta forma [3]:

1. Algoritmos que trabalham em grupos arbitrários, por exemplo, busca exaustiva, baby - step giant - step, algoritmo Pollard's rho;

2. algoritmos que trabalham em grupos arbitrários mas são especialmente eficientes se a ordem do grupo possuir fatores primos pequenos, por exemplo, algoritmos Pohllig - Hellman; e

3. Os algoritmos index - calculus que são eficientes apenas em certos grupos.

Nas próximas seções serão apresentados alguns destes algoritmos para resolver DLPs.

2.3 Busca Exaustiva

O algoritmo mais óbvio para resolver um DLP é calcular sucessivamente α^0, α^1, α^2, $\cdots$ até β ser obtido. Este método levaria $O(2^n)$ multiplicações, onde n é a ordem de α, e é além do mais ineficiente se n é de uma magnitude grande (que é o caso de interesse da criptografia) [3].

2.4 Algoritmo baby-step giant-step

Sendo $m = \lceil \sqrt{n} \rceil$, onde n é a ordem de α. O algoritmo BSGS (*Baby-Step Giant-Step*) é um algoritmo de memória limitado ao método de busca exaustiva e é baseado na seguinte observação. Se $\beta = \alpha^x$, então pode-se dizer que $x = im + j$, onde $0 \ \leq i, j < m$. Assim, $\alpha^x = \alpha^{im}\alpha^j$, que implica em $\beta(\alpha^{-m})^i = \alpha^j$. Disto, obtêm-se o seguinte algoritmo para calcular x [3]:

Algoritmo BSGS Algoritmo Baby-Step Giant-Step

ENTRADA: gerador α de um grupo cíclico G de ordem n, e um elemento $\beta \in G$

SAÍDA: o logaritmo discreto $x = \log_\alpha \beta$

1. Definir $m \leftarrow \lceil \sqrt{n} \rceil$
2. Construir uma tabela com as entradas (j, α^j) para $0 \leq j < m$. Ordenar esta tabela pelo segundo componente.
3. Calcular α^{-m} e definir $\gamma \leftarrow \beta$.
4. Para i de 0 até $m-1$ faça o seguinte:

   4.1 Verificar se γ é o segundo componente de alguma entrada na tabela.

   4.2 Se $\gamma = \alpha^j$ então retornar $(x = im + j)$.

   4.3 Definir $\gamma \leftarrow \gamma \cdot \alpha^{-m}$.

O tempo de execução do algoritmo BSGS é de $O(\sqrt{n})$ grupos de multiplicações.

2.4.1 Exemplo do BSGS

Considerar o BSGS trabalhando em logaritmos sobre Z^*_{113}. Sendo $p = 113$. O elemento $\alpha = 3$ é o gerador e a ordem $n = 112$. considerar $\beta = 57$. então $\log_3 57$ é calculado como a seguir.

1. Defina $m \leftarrow \lceil \sqrt{112} \rceil = 11$.
2. Construa uma tabela que as entradas são $(j, \alpha^j \bmod p)$ para $0 \leq j < 11$

j	0	1	2	3	4	5	6	7	8	9	10
$3^j \bmod p$	1	3	9	27	81	17	51	40	7	21	63

ordenar a tabela pelo segundo componente:

j	0	1	8	2	5	9	3	7	6	10	4
$3^j \bmod p$	1	3	7	9	17	21	27	40	51	63	81

3. Calcular $\alpha^{-1} = 3^{-1} \bmod 113 = 38$ então calcular $\alpha^{-m} = 38^{11} \bmod 113 = 58$.
4. Assim, $\gamma = \beta\alpha^{-mi} \bmod 113$ para $i = 0, 1, 2, \cdots$ é calculado até um valor na segunda linha da tabela é obtido. Gerando:

i	0	1	2	3	4	5	6	7	8	9
$\gamma = 57 \cdot 58^i \bmod p$	57	29	100	37	112	55	26	39	2	3

Finalmente, dado $\beta\alpha^{-9m} = 3 = \alpha^1$, $\beta = \alpha^{100}$ e, além do mais, $\log_3 57 = 100$.

2.5 Algoritmo Pohlig-Hellman

Este algoritmo para calcular DLPs leva em consideração o fato da fatorização de ordem n do grupo G. Sendo, $n = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$ seja a fatorização dos primos de n. Se $x = \log_\alpha \beta$, então o objetivo é determinar $x_i = x \bmod p_i^{e_i}$ para $1 \leq i \leq r$, então usar o algoritmo de Gauss [3] para recuperar $x \bmod n$. Cada inteiro x_i é determinado pelo calculo dos digitos $l_0, l_1, \cdots, l_{e_{i-1}}$ em torno de sua representação $p_i - ary$: $x_i = l_0 + l_i p_i + \cdots + l_{e_{i-1}} p_i^{e_{i-1}}$, onde $0 \leq l_j \leq p_i - 1$.

Algoritmo Pohlig-Hellman

ENTRADA: gerador α de um grupo cíclico G de ordem n, e um elemento $\beta \in G$

SAÍDA: o logaritmo discreto $x = \log_\alpha \beta$

1. Encontrar a fatorização de n: $n = n = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$, onde $e_i \geq 1$.
2. Para i de 1 até r realizar o seguinte:

(Calcular $x_i = l_0 + l_1 p_i + \cdots + l_{e_{i-1}} p_i^{e_{i-1}}$, onde $x_i = x \bmod p_i^{e_i}$)

2.1 (Simplificar a notação) Definir $q \leftarrow p_i$ e $e \leftarrow e_i$

2.2 Definir $\gamma \leftarrow 1$ e $l_{-1} \leftarrow 0$.

2.3 Calcular $\overline{\alpha} \leftarrow \alpha^{\frac{n}{q}}$.

2.4 (Calcular o l_j) Para j de 0 até $e-1$ faça :

Calcular $\gamma \leftarrow \gamma \alpha^{l_{j-1} q^{j-1}}$ e $\overline{\beta} \leftarrow (\beta \gamma^{-1})^{\frac{n}{q^{j+1}}}$.

Calcular $l_j \leftarrow \log_{\overline{\alpha}} \overline{\beta}$.

2.5 Definir $x_i \leftarrow l_0 + l_1 q + \cdots + l_{e-1} q^{e-1}$.

3. Utilizar o algoritmo de Gauss para calcular o inteiro x, $0 \leq x \leq n-1$, tal que $x \equiv x_i (\bmod p_i^{e_i})$ para $1 \leq i \leq r$.
4. Retornar (x).

O exemplo a seguir do Algoritmo Pohlig-Hellman utiliza parâmetros de pequena magnitude para facilitar a compreensão.

2.5.1 Exemplo Pohlig-Hellman

Dado $p = 251$. O elemento $\alpha = 1$ é o gerador de Z_{251}^* de ordem $n = 250$. Considerar $\beta = 210$. então $x = \log_{71} 210$ é calculado a seguir [3]:

1. A fatorização em primos de n é $250 = 2 \cdot 5^3$.

2. (a) (Calcular $x_1 = x \bmod 2$)

Calcular $\overline{\alpha} = \alpha^{\frac{n}{2}} \bmod p = 250$ e $\overline{\beta} = \beta^{\frac{n}{2}} \bmod p = 250$. Então $x_1 = \log_{250} 250 = 1$.

(b) (Calcular $x_2 = x \bmod 5^3 = l_0 + l_1 5 + l_2 5^2$)

Calcular $\overline{\alpha} = \alpha^{\frac{n}{5}} \bmod p = 20$.

Calcular $\gamma = 1$ e $\overline{\beta} = (\beta\gamma^{-1})^{\frac{n}{5}} \bmod p = 149$. Usando busca exaustiva, calcular $l_0 = \log_{20} 149 = 2$.

Calcular $\gamma = \gamma\alpha^2 \bmod p = 21$ e $\overline{\beta} = (\beta\gamma^{-1})^{\frac{n}{25}} \bmod p = 113$. Usando busca exaustiva, calcular $l_1 = \log_{20} 113 = 4$.

Calcular $\gamma = \gamma\alpha^{4 \cdot 5} \bmod p = 115$ e $\overline{\beta} = (\beta\gamma^{-1})^{\frac{p-1}{125}} \bmod p = 149$. Usando busca exaustiva, calcular $l_2 = \log_{20} 149 = 2$.

Assim, $x_2 = 2 + 4 \cdot 5 + 2 \cdot 5^2 = 72$.

3. Finalmente, resolver o par de congruências $x \equiv 1 (\text{mod} 2)$, $x \equiv 72 (\text{mod} 125)$ para obter $x = \log_{71} 210 = 197$.

Devido a fatorização de n, o tempo de execução do Pohlig-Hellman é $O(\sum_{i=1}^{r} e_i (\lg n + \sqrt{p_i}))$ grupos de multiplicações. A eficiência de tal algoritmo depende apenas se cada primo divisor p_i de n é relativamente pequeno, pois se for de magnitude grande tal método não se torna mais eficiente [3].

2.6 Algoritmo Index-Calculus

O algoritmo Index-Calculus é o mais poderoso método conhecido para calcular logaritmos discretos. A técnica empregada não se aplica a todos os grupos, mas quando é aplicável, geralmente trabalha sobre tempo sub - exponencial. O algoritmo index-calculus requer a seleção de um pequeno subconjunto S de elementos de G, chamado de *factor base*, de tal modo que uma fração significante dos elementos de G possa ser efetivamente expressada como produtos dos elementos de S [3].

Algoritmo Index-Calculus

ENTRADA: gerador α de um grupo cíclico G de ordem n, e um elemento $\beta \in G$

SAÍDA: o logaritmo discreto $y = \log_\alpha \beta$

1. (Selecionar um fator base S) Escolher um sub - conjunto $S = \{p_1, p_2, \cdots, p_t\}$ de G tal que uma "porção significativa" dos elementos em G possam ser efitivamente expressadas como um produto de elementos em S.

2. (Coletar relações lineares envolvendo logaritmos dos elementos em S)

2.1 Selecionar um inteiro aleatório k, $0 \leq k \leq n-1$, e calcular α^k.

2.2 Tentar escrever α^k como um produto dos elementos em S:

$\alpha^k = \prod_{i=1}^{t} p_i^{c_i}, c_i \geq 0$. (ic.1)

Se obtiver sucesso, pegar os logaritmos de ambos os lados da equação (ic.1) para obter uma relação linear

$k \equiv \sum_{i=1}^{t} c_i \ \log_\alpha p_i \ (\mathrm{mod} n)$ (ic.2).

2.3 Repetir os passos 2.1 e 2.2 até $t + c$ relações da forma (ic.2) forem obtidas.

3. (Encontrar os logaritmos dos elementos em S) Trabalhando com módulo n, resolver o sistema linear de $t + c$ equações (em t incógnitas) da forma (ic.2) obtidas no passo 2 para obter os valores de $\log_\alpha p_i$, $1 \leq i \leq t$.

4.(Calcular y)

4.1 Selecionar um inteiro aleatório k, $0 \leq k \leq n-1$, e calcular $\beta \cdot \alpha^k$.

4.2 Tentar escrever $\beta \cdot \alpha^k$ como um produto dos elementos em S:

$\beta \cdot \alpha^k = \prod_{i=1}^{t} p_i^{d_i}$, $d_i \geq 0$. (ic.3)

Se a tentativa não der certo então repetir o passo 4.1. Senão, pegar os logaritmos de ambos os lados da equação (ic.3) que geraria $\log_\alpha \beta = (\sum_{i=1}^{t} d_i \log_\alpha p_i - k) \bmod n$; assim, calcular $y = (\sum_{i=1}^{t} d_i \log_\alpha p_i - k) \bmod n$ e retornar (y).

2.6.1 Exemplo: Index-Calculus em Z_p^*

Para o campo Z_p um primo, o fator base S pode ser escolhido como o primeiro número primo t. Uma relação (ic.1) é gerada pelo cálculo de $\alpha^k \bmod p$ e então usando uma divisão trivial para checar se este inteiro é o produto dos primos em S. A seguir um exemplo para o Index-Calculus [3].

Sendo $p = 229$. O elemento $\alpha = 6$ é a gerador de Z_{229}^* de ordem $n = 229$. Considerar $\beta = 13$. Então $\log_6 13$ é calculado como a seguir, usando a técnica do index-

calculus [3].

1. O fator base é escolhido para ser os 5 primeiros primos: $S = \{2, 3, 5, 7, 11\}$.
2. As seguintes seis relações envolvendo elementos do fator base são obtidas:

   $6^{100} \bmod 229 = 180 = 2^2 \cdot 3^2 \cdot 5$

   $6^{18} \bmod 229 = 176 = 2^4 \cdot 11$

   $6^{12} \bmod 229 = 165 = 3 \cdot 5 \cdot 11$

   $6^{62} \bmod 229 = 154 = 3 \cdot 7 \cdot 11$

   $6^{143} \bmod 229 = 198 = 2 \cdot 3^2 \cdot 11$

   $6^{206} \bmod 229 = 210 = 2 \cdot 3 \cdot 5 \cdot 7$

   Estas relações geram as seguintes seis equações envolvendo os logaritmos dos elementos no fator base:

   $100 \equiv 2\log_6 2 + 2\log_6 3 + \log_6 5 (\text{mod} 228)$

   $18 \equiv 4\log_6 2 + \log_6 11 (\text{mod} 228)$

   $12 \equiv \log_6 3 + \log_6 5 + \log_6 11 (\text{mod} 228)$

   $62 \equiv \log_6 2 + \log_6 7 + \log_6 11 (\text{mod} 228)$

   $143 \equiv \log_6 2 + 2\log_6 3 + \log_6 11 (\text{mod} 228)$

   $206 \equiv \log_6 2 + \log_6 3 + \log_6 5 + \log_6 7 (\text{mod} 228)$

3. Resolvendo o sistema linear das seis equações com cinco incógnitas (o logaritmo $x_i = log_6 p_i$) gera a solução $log_6 2 = 21$, $log_6 3 = 208$, $log_6 5 = 98$, $log_6 7 = 107$ e $log_6 11 = 162$.
4. Supor que o inteiro $k = 77$ é selecionado. Sendo $\beta \cdot \alpha^k = 13 \cdot \alpha^{77} \bmod 229 = 147 = 3 \cdot 7^2$, vê - se que:

$\log_6 13 = (\log_6 3 + 2\log_6 7 - 77) \bmod 228 = 117.$

2.7 Discussão sobre DLPs

Como mostrado anteriormente, existem duas classes principais de algoritmos para resolver DLP, uma classe cujo tempo de execução estimado é de $O(\sqrt{p})$ e a classe de algoritmos baseados em $L_p = [k, c]$ com tempo de execução sub - exponencial. Os que são baseados na complexidade $O(\sqrt{p})$ são: Baby-Step Giant-Step, Pollard's rho, Pohlig-Hellman (somente se a ordem n for um primo) [3]. Quanto aos que são baseados na complexidade $L_p = [k, c]$ para resolver um DLP são: Index-Calculus, Coppersmith e o NFS que é uma variação do Index-Calculus sendo este o melhor algoritmo para resolver DLP [3]. Os algoritmos que solucionam um DLP em tempo de execução subexponencial como o NFS [12], a sua complexidade é descrita pela equação 2.1, onde p é primo sendo utilizado, c é uma constante e k é uma constante satisfazendo $0 \leq k \leq 1$. A Tabela 2.2

apresenta a complexidade que cada algoritmo leva para resolver DLP.

$$L_p[k,c] = O(e^{(c+O(1))\cdot((\ln\ p)^k)\cdot((\ln\ \ln\ p)^{1-k})}) \tag{2.1}$$

O gráfico apresentado na Figura 2.4 mostra a comparação entre os métodos baseados na complexidade $O(\sqrt{p})$ (BSGS) e o NFS [12]. O eixo horizontal mostra a variação da magnitude do número primo p em bits e no eixo vertical é mostrada em bits a complexidade para resolver um DLP. Até primos com menos de 80 bits de magnitude, nos métodos baseados em $O(\sqrt{p})$ encontram a solução em menor tempo de execução. Contudo, com primos maiores que 80 bits de magnitude torna-se mais vantajoso o uso do NFS, uma vez que para primos com 1024 bits o NFS requer apenas $O(2^{132})$ execuções enquanto que os métodos baseados em $O(\sqrt{p})$ requerem $O(2^{512})$, conforme [3].

Tabela 2.2: Algoritmos clássicos para resolver DLP

Algoritmo	**Complexidade**
BSGS	$O(\sqrt{p})$
Pollard's rho	$O(\sqrt{p})$
Pohlig-Hellman	$O(\sum_{i=1}^{r} e_i(\lg n + \sqrt{p_i}))$
Index-Calculus	$L_p = [\frac{1}{2}, c]$
Coppersmith	$L_p = [\frac{1}{3}, 1.587]$
NFS	$L_p = [\frac{1}{3}, 1.923]$

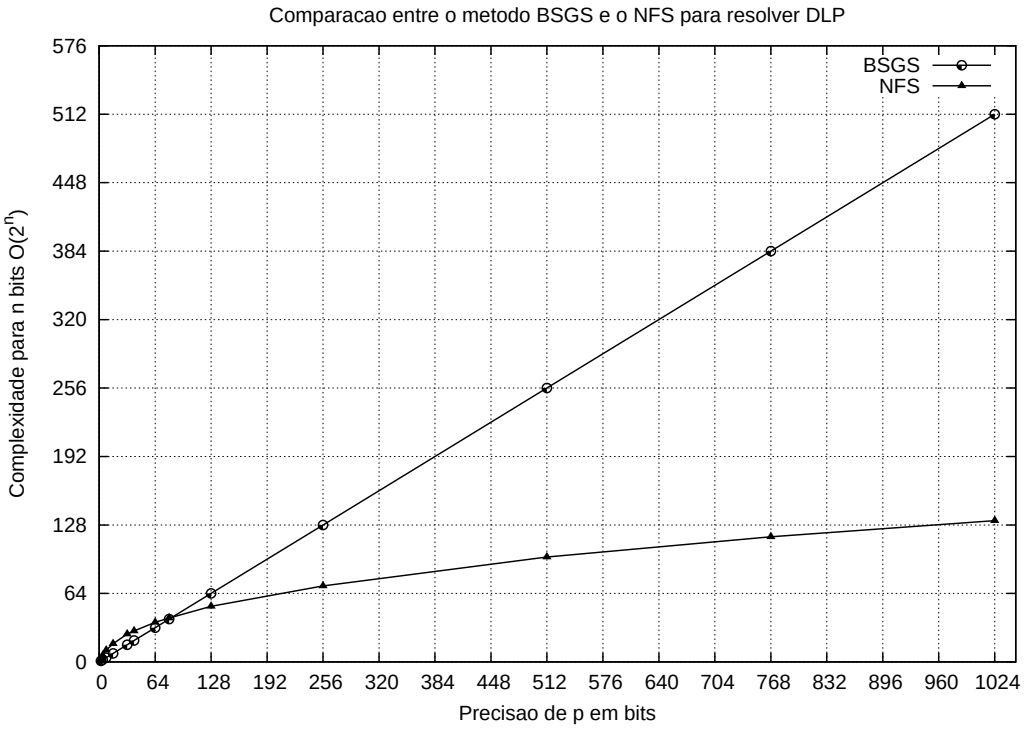

Figura 2.4: Comparação entre os Métodos BSGS e NFS

A próxima seção irá discutir sobre grupos de matrizes sobre campos finitos, devido ao fato do protocolo proposto nesta dissertação utilizar de tais grupos de matrizes e também para facilitar a compreensão das operações matemáticas apresentadas no Capítulo 4.

2.8 Matrizes Sobre Z_p^*

Um campo é a menor estrutura matemática no qual podem-se realizar todas as operações aritméticas soma, subtração, multiplicação e divisão (sendo esta por elementos não nulos), em particular todo elemento não nulo deve possuir sua inversa multiplicativa [13]. A definição de um campo finito é:

1. Um *campo* é uma tripla $(F, +, \cdot)$ tal que $(F, +)$ é um grupo abeliano (chamado sua identidade 0) e $(F - \{0\}, \cdot)$ é também um grupo abeliano, e a seguinte lei *distributiva* diz: $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$, para todo $a, b, c \in F$ [13].

2. Para qualquer campo F tem $F^* = F - \{0\}$ [13].

Para cada $n \in Z^+$ deixe $GL_n(F)$ ser o conjunto de todas $n \times n$ matrizes que as entradas vêm de F e cujo determinante é não nulo, isto é, $GL_n(F) = \{A | A$ é uma $n \times n$ matriz com entradas de F e $det(A) \neq 0\}$, onde o determinante de qualquer matriz A com entradas de F pode ser calculado pela mesmas fórmulas usadas em $F = R$ (Campo no domínio dos reais). Para matrizes arbitrárias $n \times n$ A e B deixe ser AB como o produto destas matrizes como calculado pelas mesmas regras como quando $F = R$. Este produto é associativo. Também, desde que $det(AB) = det(A) \cdot det(B)$, segue que se $det(A) \neq 0$ e $det(B) \neq 0$, então $det(AB) \neq 0$, logo $GL_n(F)$ está fechado sobre multiplicação de matrizes. Além do mais, $det(A) \neq 0$ se e somente se A possui uma matriz inversa, logo cada $A \in GL_n(F)$ possui uma inversa, A^{-1}, em $GL_n(F)$: $AA^{-1} = A^{-}1A = I$, onde I é uma matriz identidade $n \times n$. Assim $GL_n(F)$ é um grupo sobre multiplicação de matrizes, chamado também de *grupo linear geral de ordem n* [13].

2.9 Conclusão

Este capítulo apresentou uma visão geral sobre criptografia tradicional bem como os algoritmos que se utilizam de criptografia assimétrica e/ou simétrica, uma análise sobre DLPs (Problemas de Logaritmo Discreto), os algoritmos utilizados para tentar resolver um DLP em menor tempo do que por busca exaustiva, uma análise comparativa entre

os algoritmos que resolvem um DLP e suas complexidades, além de uma breve descrição sobre matrizes sobre campos finitos. No próximo capítulo serão apresentados os principais protocolos estudados para a análise e o desenvolvimento do DHA3, que são: Algorítmo Criptográfico Diffie-Hellman, El-Gamal, MTI, SPAKE, S-3PAKE, Needham-Schroeder, $\gamma = \alpha^a \beta^b$ e Matrix rings. Sendo que ao final de cada seção algumas análises sobre segurança de tais protocolos serão feitas.

Capítulo 3

Protocolos de Acordo de Chave

Neste Capítulo, serão apresentados os trabalhos relacionados ao DHA3. São protocolos de acordo de chave criptográfica que possuem sua complexidade baseada nos (DLPs) problemas de logaritmo discreto. Os protocolos são os seguintes: Diffie-Hellman, ElGamal, MTI, CRTDH, Needham-Schoreder, SPAKE, S-3PAKE, $\gamma = \alpha^a \beta^b$ e Matrix Rings.

3.1 Diffie-Hellman

O DH é o protocolo de acordo de chaves criptográficas mais utilizado nas aplicações e implementações. Os protocolos de acordo de chave possuem um importante papel no estabelecimento de um canal seguro de comunicação entre partes comunicantes. O DH também pode ser chamado de um protocolo baseado em chave exponencial. A complexidade para resolver tal algoritmo depende da aparente intratabilidade dos DLP's. O DH é um protocolo de acordo de chave sem autenticação das partes baseado em DLP. O DH [3] usa duas trocas para gerar a chave compartilhada K entre os nós.

O DH requer um número primo p e um gerador $\alpha \in Z_p^*$, onde $2 \leq \alpha \leq p-1$, de forma que estes valores sejam pré compartilhados entre os nós (usando, por exemplo, um repositório para armazenar e gerenciar tais valores), como mostrado no Passo 1 da Figura 3.1. Depois destes procedimentos o nó A realiza então as operações apresentadas no Passo 2 e a mesma operação é realizada pelo nó B, como mostrado na Figura 3.1. No passo 3 ambos os nós enviam o resultado das operações realizadas no Passo 2. Ao final no Passo 4 cada nó irá calcular a chave compartilhada através das mensagens recebidas no Passo 2. Este protocolo opera com uma troca, com duas mensagens entre os nós. A chave gerada pelo DH terá a mesma magnitude do primo p utilizado.

Para melhor explanação do Diffie-Hellman, a seguir é apresentado um exemplo numérico de tal protocolo. Considerando o número primo $p = 7$, o gerador $\alpha = 3$, $x = 4$

Passo	Nó A	Trocas	Nó B
1	Os nós concordam num primo grande p e num gerador α.		
2	Escolhe um número aleatório grande $1 \leq x \leq p-2$ e calcula $z_1 = \alpha^x \mod p$		Escolhe um número aleatório grande $1 \leq y \leq p-2$ e calcula $z_2 = \alpha^y \mod p$
3		$z_1 \Rightarrow$ $\Leftarrow z_2$	
4	Calcula sua chave $K = (z_2)^x \mod p = (z_1)^y \mod p$		Calcula sua chave $K = (z_1)^y \mod p = (z_2)^x \mod p$

Figura 3.1: Protocolo de acordo de chave Diffie-Hellman.

e $y = 5$, assim as mensagens trocadas entre os nós A e B são as equações (3.1) e (3.2), sendo (3.3) a chave compartilhada gerada por cada nó:

$$z_1 : 3^4 \mod 7 = 4 \tag{3.1}$$

$$z_2 : 3^5 \mod 7 = 5 \tag{3.2}$$

$$K = (z_1)^5 \mod 7 = (z_2)^4 \; mod \; 7 = 2 \tag{3.3}$$

O exemplo apresentado é muito fácil de ser quebrado. Para que isso não ocorra o número primo utilizado deverá ter uma precisão de 1024 bits a 2048 bits, que gera um número de 300 a 500 digitos decimais. Tal sistema provê proteção contra ataques passivos (*eavesdroppers*), mas não contra ataques ativos onde o adversário pode interceptar, modificar, ou mesmo injetar mensagens falsas. Nenhuma parte tem certeza da identidade do outro e nem mesmo da identidade da outra parte a qual está se gerando a chave, sem autenticação qualquer.

De forma genérica o DLP pode ser definido como sendo: dado um primo p, um gerador α sobre Z_p^* [3], e $\alpha^a \; mod \; p$, encontre a.

3.2 ElGamal

O Protocolo ElGamal também é baseado na complexidade de resolver o DLP. A Figura 3.2 apresenta o funcionamento do ElGamal.

O protocolo de acordo de chave ElGamal [3] é uma variante do Diffie-Hellman provendo um protocolo de apenas um passo (troca de apenas uma mensagem) com a

Passo	**Nó A**	**Trocas**	**Nó B**
1	Os nós concordam num primo grande p e num gerador α.		
2			Escolhe um número aleatório grande $1 \leq y \leq p-2$ e calcula $z_2 = \alpha^y \mod p$ e torna z_2 público através de certificado.
3	Escolhe um número aleatório grande $1 \leq x \leq p-2$ e calcula $z_1 = \alpha^x \mod p$		
4		$z_1 \Rightarrow$	
5	Calcula sua chave $K = (z_2)^x \mod p = (z_1)^y \mod p$		Calcula sua chave $K = (z_1)^y \mod p = (z_2)^x \mod p$

Figura 3.2: Protocolo de acordo de chave El Gamal.

autenticação de chave unilateral (Do receptor para o originador da troca), provida através da chave do receptor que é conhecida a *priori* pelo originador, sendo esta, por exemplo embutida num certificado podendo assim verificar a sua autenticidade. Ambos os nós selecionam o mesmo primo p e um gerador $\alpha \in Z_p^*$, passo 1 da Figura 3.2. O nó B seleciona um número inteiro y, e realiza o passo 2 da Figura 3.2. Após estes procedimentos, o nó A pega uma cópia da chave pública de B (p, α, z_2), para então o nó A realizar o passo 3. No passo 4, envia z_1 para o nó B. No passo 5, cada nó computa sua chave, sendo que a chave gerada em cada nó será a mesma.

Um exemplo prático de tal protocolo é mostrado a seguir. Tomando como base os seguintes valores $p = 7, \alpha = 3, x = 4, y = 5$, a chave pública de B em (3.4), o nó A envia para B (3.5), assim cada nó irá gerar a mesma chave K em (3.6).

$$(p, \alpha, z_2) = (7, 3, 5) \tag{3.4}$$

$$z_1 : 3^4 \mod 7 = 4 \tag{3.5}$$

$$K = (z_1)^5 \mod 7 = (z_2)^4 \mod 7 = 2 \tag{3.6}$$

O protocolo ElGamal gera apenas uma mensagem devido ao fato do nó B previamente colocar os seus dados públicos em um servidor de chaves públicas (por exemplo, embutido num certificado), de outra maneira este protocolo seria como o Diffie-Hellman,

e somente existe a autenticidade dos dados do nó B para o nó A devido ao fato novamente do Servidor de chaves públicas ser requerido por este protocolo. Em termos de complexidade para resolver o ElGamal, enquadra-se no mesmo caso do DH que é resolver DLP [3].

3.3 Protocolo MTI

O Protocolo MTI também é baseado na complexidade para resolver o DLP. O MTI [3] é um protocolo de acordo de chave realizado em uma troca com duas mensagens entre os nós participantes. O MTI/A0 é uma variante do DH, com a vantagem de garantir a autenticidade das chaves públicas de cada nó participante do acordo de chave, devido ao fato de estas chaves públicas estarem embutidas em certificados, sendo assim possível verificar a autenticidade das mesmas. De maneira semelhante aos dois protocolos discutidos anteriormente, são escolhidos um primo p e um gerador $\alpha \in Z_p^*$, $1 \leq \alpha \leq p-2$. Estes dados são armazenados num servidor como mostrado no passo 1 da Figura 3.3.

Passo	**Nó A**	**Trocas**	**Nó B**
1	Os nós concordam num primo grande p e num gerador α, armazenados em servidor de chaves públicas.		
2	Escolhe um número aleatório grande $1 \leq a \leq p-2$ e calcula uma chave pública $z_A = \alpha^a \mod p$		Escolhe um número aleatório grande $1 \leq b \leq p-2$ e calcula uma chave pública $z_B = \alpha^b \mod p$
3	Escolhe $1 \leq x \leq p-2$, calcula $z_1 = \alpha^x \mod p$		Escolhe $1 \leq y \leq p-2$, calcula $z_2 = \alpha^y \mod p$
4		$z_1 \Rightarrow$ $\Leftarrow z_2$	
5	Calcula sua chave $K = (z_2)^a (Z_B)^x \mod p$		Calcula sua chave $K = (z_1)^b (Z_A)^y \mod p$
6	Os nós geram a mesma chave $K = \alpha^{bx+ay}$.		

Figura 3.3: Protocolo de acordo de chave MTI/A0.

O nó A seleciona uma chave privada que é um inteiro aleatório a, passo 2, Figura 3.3. De forma análoga, passo 2, B gera b e Z_B. Os nós A e B possuem acesso aos certificados de cada um dos nós, sendo assim possível obter as chaves públicas autenticadas de cada um dos nós. O nó A escolhe um segredo aleatório x e realiza o passo 3, tal que $1 \leq x \leq p-2$. Nó B escolhe um segredo aleatório y e realiza as operações descritas no

passo 3 da Figura 3.3. No passo 4, os nós enviam os resultados das operações realizadas no passo 3. Os nós A e B computam suas respectivas chaves como mostrado no passo 5. No passo 6, ambos os nós acabam por compartilhar a mesma chave.

Considere um exemplo numérico do protocolo MTI/A0 que servirá de base para a análise e compreensão do seu funcionamento. Sendo primo $p = 19$, $\alpha = 3$, $a = 5$, $b = 6$, $x = 7$ e $y = 9$. Assim, resulta-se que $z_A = \alpha^a \mod p = 15$ e $z_B = \alpha^b \mod p = 7$. Dessa forma o nó A envia para o nó B equação (3.7), em seguida o nó B envia pra nó A (3.8). Após estes procedimentos cada nó gera sua própria chave que acaba sendo a mesma chave compartilhada nó A (3.9) e nó B (3.10), mostrando que as chaves geradas são as mesmas.

$$z_1 = \alpha^x \mod p = 2 \tag{3.7}$$

$$z_2 = \alpha^y \mod p = 18 \tag{3.8}$$

$$K = (18)^5(7)^7 \mod 19 = 12 \tag{3.9}$$

$$K = (2)^6(15)^9 \mod 19 = 12 \tag{3.10}$$

O MTI possui quatro variantes, a apresentada acima é a MTI/A0. As demais se diferenciam pelas informações que são trocadas entre os nós e os expoentes escolhidos para as chaves públicas. A Tabela 3.1 apresenta estas variações, levando em consideração que $z_A = \alpha^a \mod p$ e que $z_B = \alpha^b \mod p$. Onde K_A e K_B denotam as chaves geradas em cada nó que representam ao final dos procedimentos a mesma chave K.

Tabela 3.1: Variantes do MTI

Protocolo	z_1	z_2	K_A	K_B	Chave K
MTI/A0	α^x	α^y	$\alpha^{ya} \cdot Z_B^x$	$\alpha^{xb} \cdot Z_A^y$	α^{bx+ay}
MTI/B0	z_B^x	z_A^y	$(z_A^y)^{a^{-1}} \cdot \alpha^x$	$(z_B^x)^{b^{-1}} \cdot \alpha^y$	α^{x+y}
MTI/C0	z_B^x	z_A^y	$(z_A^y)^{a^{-1}x}$	$(z_B^x)^{b^{-1}y}$	α^{xy}
MTI/C1	z_B^{xa}	z_A^{yb}	$(z_A^{yb})^x$	$(z_B^{xa})^y$	α^{abxy}

Apesar dos protocolos MTIs implementarem a autenticação das chaves públicas (z_A e z_B) estes protocolos são suceptíveis a ataques ativos, logo devem ser utilizados em contextos onde apenas ataques passivos sejam possíveis [3]. Mas mesmo assim tal protocolo ainda trabalha com um mesmo gerador α o que permite a solução por DLP. Com relação a complexidade computacional dos protocolos MTI têm-se : Os protocolos $A0$ e $B0$ requerem 3 exponenciações por cada nó, enquanto que o $C0$ e o $C1$ requerem apenas 2 exponenciações (A complexidade para resolver uma exponenciação é $O((log\ p)^3)$). O $C1$ apresenta uma vantagem em relação ao $B0$ e o $C0$ que é o fato de não requerer o cálculo da inversa multiplicativa (A complexidade para calcular a inversa é $(O((log\ p)^2))$ [3].

3.4 Protocolo Needham-Schroeder

O protoclo Needham-Schroeder [3] é baseado em criptografia por chave pública e pode ser usado para um acordo de chave criptográfica. Dois nós desejando estabelecer uma chave de sessão irão compartilhar duas chaves secretas simétricas, e ao final do protocolo cada nó irá calcular o *hash* das chaves compartilhadas para então gerar a chave da sessão. Notação: $P_X(Y)$ representa a cifragem com algoritmo de chave pública do dado Y usando a chave pública do nó X; $P_X(Y_1, Y_2)$ representa a cifragem da concatenação dos dados Y_1 e Y_2 ; k_1 e k_2 são as chaves secretas simétricas pertencentes aos nós A e B, respectivamente.

Neste protocolo assume-se *a priori* que cada nó possui a chave pública autêntica do outro nó através de certificados e/ou outra forma de garantir a autenticidade das chaves públicas utilizadas. Caso não possuam as chaves públicas autenticadas passos adicionais serão necessários neste protocolo para a obtenção das chaves. A Figura 3.4 apresenta o funcionamento deste protocolo.

Passo	**Nó A**	**Trocas**	**Nó B**
1	Utiliza a chave pública de B para gerar $z_1 = P_B(k_1)$		
2		$z_1 \Rightarrow$	
3			Utiliza a chave pública de A para gerar $z_2 = P_A(k_1, k_2)$
4		$\Leftarrow z_2$	
5	Verifica os dados recebidos e se estiver correto gera $z_3 = P_B(k_1, k_2)$		
6		$z_3 \Rightarrow$	
7	Caso os dados estejam em conformidade gerará a chave de sessão $K = h(k_1, k_2)$ através de uma função de *hash*.		Verifica os dados recebidos e calcula o *hash* das chaves para gerar a chave de sessão $K = h(k_1, k_2)$.

Figura 3.4: Funcionamento do protocolo Needham-Schroeder.

O protocolo é realizado com três trocas de mensagens: O nó A realiza o passo 1 e no passo 2 envia para o nó B sua chave simétrica cifrada. O nó B ao receber z_1 obtém k_1, e com isso no passo 3 calcula z_2 e no passo 4 envia z_2 para A. O nó A ao receber z_2 verifica se a chave k_1 recuperada, irá coincidir com a chave enviada no passo 2. Caso coincida, ele gera z_3 no passo 5 e envia para B no passo 6, caso não coincida ele termina

o protocolo e não estabelece comunicação com B. O nó B ao receber z_3 verifica se a chave k_2 coincide com a chave enviada no passo 4, caso coincida este realizará o passo 7 que é um *hash* das chaves k_1 e k_2 para obter a chave de sessão; o mesmo procedimento será realizado por A para gerar a chave de sessão K.

Contudo, este protocolo requer uma Autoridade Certificadora para garantir que as chaves públicas utilizadas no protocolo sejam autênticas, e apesar de realizar o procedimento para o acordo da chave de sessão em apenas três trocas, este protocolo poderá vir a requisitar passos adicionais caso os nós não possuam as chaves públicas um do outro. O protocolo de acordo de chave proposto nesta dissertação (DHA3) não autentica as entidades comunicantes sem que haja a intervenção de um TTP (*Trusted Third Party*). O mesmo ocorre com o protocolo do Needham - Schroeder, pois caso os nós apenas troquem as chaves públicas não haverá como garantir a autenticidade das mesmas.

3.5 CRTDH e Grupos Seguros de Comunicação

Nesta seção será fornecido um embasamento sobre Grupos Seguros de Comunicação antes de discutir sobre o protocolo CRTDH. Em redes cabeadas, normalmente os serviços de segurança como autenticação, gerenciamento de chaves e autorização são realizados por um TTP servidor centralizado. Em ambientes *ad-hoc* tais tipos de serviços como uma CA não são geralmente disponíveis, de forma que os próprios membros pertencentes a tal rede deverão prover tais serviços de segurança por eles mesmos. Considerando-se um cenário onde um número de nós móveis forma uma rede *ad-hoc* sem qualquer conhecimento prévio, pode-se definir o que é um SGC (*Secure Group of Communication*) ou Grupo Seguro de Comunicação.

Um SGC é definido como o processo pelo qual membros de um grupo podem seguramente comunicar-se entre si e a informação sendo transmitida é inacessível a qualquer elemento que não pertença ao grupo [14]. No cenário citado acima, uma chave de grupo é compartilhada pelos membros deste grupo e esta chave é utilizada para cifrar todas as mensagens trafegadas neste grupo. Um SGC deve evitar a serialização dos membros, onde a informação é transmitida de membro a membro numa seqüência pré-definida para então ser criada a chave do grupo. Devido ao fato da mobilidade e o dinamismo de redes *ad-hoc* a serialização deve ser evitada. Um SGC deve possuir um bom mecanismo para o compartilhamento da chave do grupo.

O protocolo de acordo de chaves utilizado por SGC deve ser eficiente na questão de economizar energia nos cálculos computacionais e no uso do canal de comunicação. Os nós de uma rede ad-hoc geralmente possuem limitações no consumo da bateria. Também

num SGC deve-se levar em conta que usuários móveis entrarão e sairão do grupo com um dinamismo alto. Assim um bom SGC deve cuidar destes aspectos para controlar corretamente a taxa de atualização das chaves de sessão ao ponto de garantir a segurança da comunicação e ao mesmo tempo não inferir num alto consumo computacional dos novos membros que irão participar do grupo.

Em [14] é proposto um protocolo para acordo de chaves para SGC, que utiliza o DH e o Teorema do Resto Chinês CRT (*Chinese Remainder Theorem*), sendo que a proposta é denominada pelos autores como CRTDH. Cada membro do grupo deverá executar 7 passos, sendo n o número de membros e $i = 1, \cdots, n$ e i representa o i-ésimo membro, sendo $\oplus$ a operação binária **XOR**. Estes passos são descritos a seguir:

Passo 1 : Cada nó seleciona os seus parâmetros privados do DH os x_i e cada nó calcula os seus parâmetros públicos (3.11).Os parâmetros α e p são o gerador e primo usados nas operações do DH. Estas informações são públicas e se os nós não as compartilham, então um *broadcast* de tais informações se faz necessário.

Passo 2 : Realizar o *broadcast* dos parâmetros públicos y_i de cada nós entre todos os membros do grupo.

Passo 3 : Receber os parâmetros públicos de todos os outros membros do grupo e calcular a chave DH compartilhada com cada um deles (3.12), onde $j = 1, \cdots, i-1, i+1, \cdots, n$.

Passo 4 : Encontrar o Mínimo Múltiplo Comum LCM (*Least Commom Multiple*) de todas as chaves DH calculadas no Passo 3, denominado de lcm_i.

Passo 5 : Selecionar um valor aleatório k_i, tal que $k_i < min(m_{ij}, \forall j)$, que será compartilhado com o grupo. Também selecionar um número arbitrário D tal que $D \neq k_i$ e outro número D_p tal que $gcd(D_p, lcm_i) = 1$, onde gcd é o máximo divisor comum.

Passo 6 : Resolver os CRTs (3.14), realizar um *broadcast* destes valores obtidos para o grupo.

Passo 7 : Receber os resultados dos cálculos dos CRT's de todos os outros membros do grupo e calcular (3.15) para todos os $j \neq i$. Por fim, calcular a GK chave do grupo que será utilizado pelo grupo para estabelecer um canal seguro de comunicação (3.16).

$$y_i = \alpha^{x_i} \mod p \tag{3.11}$$

$$m_{ij} = y_j^{x_i} \mod p \tag{3.12}$$

$$gcd(D_p, lcm_i) = 1 \tag{3.13}$$

$$crt_i \equiv k_i \mod lcm_i,\ crt_i \equiv D \mod D_p \tag{3.14}$$

$$k_j = crt_j \mod m_{ij} \tag{3.15}$$

$$GK = k_1 \oplus k_2 \oplus \cdots \oplus k_n \tag{3.16}$$

Caso um novo membro U decida participar do SGC este deverá realizar os passos de 1 a 6, sendo que no sétimo passo o novo membro irá receber um *hash* da chave do grupo para então realizar o XOR com o seu k_U, gerando (3.17).

$$GK_{novo} = h(GK) \oplus k_U \tag{3.17}$$

Se um um dos membros do grupo desejar sair do grupo, o grupo deverá escolher um dos membros restantes (membro 1, por exemplo) e elegê-lo para realizar os passos do acordo de chave (de 1 a 7) novamente e assim calcular a nova chave do grupo (3.18).

$$GK_{novo} = GK \oplus k_1 \tag{3.18}$$

Considerando somente a parte do acordo de chave e da questão do SGC, vê-se que um SGC nada mais é do que um servidor de autenticação visto sob outra ótica. O SGC passa a responsabilidade da autenticação e da integridade para o grupo que em outros modelos de acordos de chave a responsabilidade da autenticação é do servidor de autenticação. Logo, o protocolo DHA3 que é um protocolo sem autenticação das partes, usado para o acordo da chave de sessão, permite que a responsabilidade da autenticação e da integridade seja feita pelos nós que desejarem estabelecer o acordo de chave, ou utilizar um servidor de autenticação para realizar a autenticação onde cada nó disponibilizará seus parâmetros públicos neste servidor.

3.6 SPAKE

SPAKE (*Simple Password-based Encrypted Key Exchange Protocol*) [7] também é considerado como uma pequena variação do protocolo DH, porém nesta variação os nós compartilham uma senha secreta chamada de pw. Esta senha servirá de base para gerar a chave de sessão que será utilizada por ambos os nós. Este protocolo utiliza alguns parâmetros públicos que são: G é um grupo finito cíclico, α é o gerador sobre p onde p é o

número primo utilizado; M, N são números aleatórios pertencentes a G, e $h()$ representa uma função de *hash*. Dessa forma o protocolo é realizado com duas trocas. A Figura 3.5 mostra o funcionamento deste protocolo.

Passo	**Nó A**	**Trocas**	**Nó B**
1	Os nós concordam num primo grande p e num gerador α e numa senha pw.		
2	Escolhe um número aleatório grande $1 \leq x \leq p-2$ para então calcular $z_1 = \alpha^x \cdot M^{pw}\ mod\ p$.		Escolhe um número aleatório grande $1 \leq y \leq p-2$ para então calcular $z_2 = \alpha^y \cdot N^{pw}\ mod\ p$.
3		$z_1 \Rightarrow$	
4		$\Leftarrow\ z_2$	
5	Calcula $K_A = (\frac{z_2}{N^{pw}})^x mod\ p$, $SK_A = h(A, B, z_1, z_2, K_A)$		Calcula $K_B = (\frac{z_1}{M^{pw}})^y mod\ p$, $SK_B = h(A, B, z_1, z_2, K_B)$
6	Ambos os nós possuem a mesma chave $SK_A = SK_B$.		

Figura 3.5: Funcionamento do protocolo SPAKE.

Primeiramente o nó A realiza o passo 2, e calcula z_1 e então o envia para o nó B no passo 3. O nó B por sua vez, similarmente ao nó A gera z_2 para então enviar para o nó A no passo 4. No passo 5, cada nó calcula a chave de sessão gerada e ainda a aplicam numa função de hash as informações trocadas, mais os dados gerados das trocas realizadas. Ao final, no passo 6 os nós possuirão a mesma chave de Sessão.

Entretanto este protocolo é susceptível a ataque de senha (*password*), devido ao fato que uma senha é limitada a alguns caracteres e símbolos e não a todo o universo que um byte pode armazenar (2^8 possíveis valores). Existe neste protocolo uma autenticação mútua, devido ao compartilhamento de um segredo o pw, assim ambos devem conhecer as qualificações um do outro antes de estabelecer um canal seguro de informação.

3.7 S-3PAKE

No protocolo S-3PAKE (*Simple Three-Party Key Exchange Protocol*) os autores [7] sugerem um método para troca autenticada de uma chave de sessão sem a presença de um servidor de chaves públicas. No entanto é necessário um servidor confiável para autenticar as partes. Algumas definições que são utlizadas por esse protocolo: G:Grupo

finito cíclico; α: gerador sobre p; p: primo para campo finito; M, N: números aleatórios sobre o grupo G; S: servidor confiável; $pw1$: senha compartilhada entre o nó A e S; $pw2$: senha compartilhada entre o nó B e S; H, H': funções de hash; $X||Y$: concatenação das informações X e Y.

No S-3PAKE, considerar que dois nós A e B desejam entrar em acordo numa chave de sessão. Entretanto, como nenhum dos nós possui qualquer conhecimento *a priori* do outro, não podem autenticar-se mutuamente e faz-se necessário um servidor confiável aos quais ambos os nós possuem uma senha compartilhada com tal servidor. A Figura 3.6 descreve o funcionamento do S-3PAKE.

Passo	**Nó A**	**Nó B**	**Servidor S**
1	Envia $X = \alpha^x \cdot M^{pw1}$ para nó A		
2		$Y = \alpha^y \cdot N^{pw2}$	
3		Envia $X\|\|Y$ para S	
4			$\alpha^x = \frac{\alpha^x}{M^{pw1}}$; $\alpha^y = \frac{Y}{N^{pw2}}$
5			$\alpha^{xz} = (\alpha^x)^z$; $\alpha^{yz} = (\alpha^y)^z$
6			$X' = \alpha^{yz} \cdot H(A, S, \alpha^x)^{pw1}$ $Y' = \alpha^{xz} \cdot H(B, S, \alpha^y)^{pw2}$
7			Envia para nó B $X'\|\|Y'$
8		$\alpha^{xz} = \frac{Y'}{H(B,S,\alpha^y)^{pw2}}$ $\alpha^{xyz} = (\alpha^{xz})^y$ $\gamma = H(A, B, \alpha^{xyz})$	
9		Envia $X'\|\|\gamma$ para nó A	
10	$\alpha^{yz} = \frac{X'}{H(A,S,\alpha^x)^{pw1}}$ $\alpha^{xyz} = (\alpha^{yz})^x$ $SK_A = H'(A, B, \alpha^{xyz})$		
11	Envia para o nó B: $\beta = H(B, A, \alpha^{xyz})$		
12		$SK_B = H'(A, B, \alpha^{xyz})$	

Figura 3.6: Funcionamento do protocolo S-3PAKE.

No passo 1, o nó A escolhe um número aleatório $x \in Z_p$, e envia X para B. B no passo 2, escolhe um número aleatório $y \in Z_p$ e computa Y, para então enviá-lo para o servidor S no passo 3. Ao receber os dados do nó B, o servidor usa as senhas $pw1$ e

$pw2$ para calcular os dados do passo 4. Assim, S escolhe outro número aleatório $z \in Z_p$ no passo 5, e realiza o passo 6. Finalmente, S envia para B $X'||Y'$ no passo 7. Quando B recebe $X'||Y'$ este usa $pw2$ para realizar o passo 8. Por fim, B encaminha $X'||\gamma$ para A através do passo 9. No passo 10, A recebe $X'||\gamma$, então A checa se o γ recebido é o mesmo do calculado. Se não for o mesmo A termina o protocolo e desiste da comunicação com B. Caso γ seja o mesmo A é convencido de que α^{xyz} é válido. E neste caso, ele pode computar a chave de sessão SK_A apresentada no passo 10 e enviar β para B para que seja verificado através do passo 11. B ao receber β verifica se β é verdadeiro. Se for verdadeiro, B calcula a chave de sessão SK_B através do passo 12, caso não seja, desiste da comunicação com A. Ao final do protocolo ambos os nós terão a mesma chave de sessão (3.19).

$$SK_A = SK_B = H'(A, B, \alpha^{xyz}) \tag{3.19}$$

3.8 Sistema Criptográfico baseado em DLP $\gamma = \alpha^a \beta^b$

Neste protocolo criptográfico os autores [5] propõem o uso de dois geradores distintos α e β pertencentes a Z_p^*, onde os autores consideram que tal protocolo exige que sejam resolvidos dois DLPs um para cada gerador. No entanto, tal afirmação não é consistente e será explanada após a explicação do funcionamento de tal protocolo.

Este protocolo requer um número primo p e dois geradores α e β, onde $2 \leq \alpha, \beta \leq p - 1$, de forma que estes valores sejam pré compartilhados entre os nós (usando, por exemplo, um repositório para armazenar e gerenciar tais valores), como mostrado no Passo 1 da Figura 3.7. Depois destes procedimentos o nó A realiza então as operações apresentadas no Passo 2 e a mesma operação é realizada pelo nó B, como mostrado na Figura 3.7. No passo 3, ambos os nós enviam o resultado das operações realizadas no Passo 2. Ao final no Passo 4, cada nó irá calcular a chave compartilhada através das mensagens recebidas no Passo 2. O Passo 5 mostra a chave equivalente gerada em cada nó. Este protocolo opera com duas trocas entre os nós, mas é realizado o envio de dois parâmetros para cada troca.

Segundo [6] apesar dos autores deste protocolo justificarem que é necessário realizar dois DLPs tal afirmação não é verdadeira devido ao fato de que o gerador β está sobre Z_p^* de forma que pode ser considerado como sendo um fator de α, $\beta = \alpha^m$, logo $\gamma = \alpha^a \beta^b$ pode ser reescrito da seguinte maneira $\gamma = \alpha^a \cdot (\alpha^m)^b = \alpha^a \cdot (\alpha^{mb}) = \alpha^{a+mb}$. Assim bastará resolver apenas um DLP para α^{a+mb}. Além do protocolo não aumentar a complexidade, pois pode ser resolvido através de apenas um DLP ainda transmite o dobro de informações que o protocolo DH transmite. Sendo assim, tal protocolo não apresenta

vantagens criptográficas e nem de desempenho na implementação real dele.

Passo	**Nó A**	**Trocas**	**Nó B**
1	Os nós concordam num primo grande p e nos geradores α e β.		
2	Escolhe dois números aleatórios grandes $1 \leq a, b \leq p-2$ e calcula $z_1 = \alpha^a \ mod \ p$ e $z_2 = \beta^b \ mod \ p$		Escolhe dois números aleatórios grandes $1 \leq c, d \leq p-2$ e calcula $z_3 = \alpha^c \ mod \ p$ e $z_4 = \beta^d \ mod \ p$
3		$z_1 \vert z_2 \Rightarrow$ $\Leftarrow z_3 \vert z_4$	
4	Calcula sua chave $K = (z_3)^a \cdot (z_4)^b \ mod \ p$		Calcula sua chave $K = (z_1)^c \cdot (z_2)^d \ mod \ p$
5	Os nós obtêm a mesma chave $K = \alpha^{ac} \cdot \beta^{bd} \ mod \ p$.		

Figura 3.7: Protocolo de acordo de chave baseado em DLP $\gamma = \alpha^a \beta^b$.

3.9 Protocolo Matrix Rings

Em [8] é proposto o primeiro protocolo que utiliza uma matrix como gerador base para o DH numa tentativa de aumentar a complexidade da criptoanálise. O protocolo proposto requer que ambos os nós compartilhem um número primo grande p e uma matriz geradora $A \in GL_n$, tal que seja inversível e ainda admita um expoente r que, ao realizar A^r gere-se a matriz identidade $A^r = I \ (mod \ p)$. O parâmetro r deve também ser compartilhado entre os nós. Considerando a comunicação entre o nó A e o nó B, o nó A escolhe o segredo $1 \leq x < r$ e o nó B escolhe $1 \leq y < r$. Assim, cada nó realiza os passos apresentados na Figura 3.8, que é exatamente o procedimento realizado no Diffie-Hellman, mas com a diferença de que o gerador base é uma matriz de ordem r. No entanto a matriz A requer alguns procedimentos extras para ser criada e além disso é necessário determinar a magnitude do parâmetro r conforme a seguir:

Dado um polinômio irredutível $f(x)$ de grau m

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{m-1} x^{m-1} \quad a_i \in GF(p), \tag{3.20}$$

uma matriz B é construída com os parâmetros do polinômio $f(x)$ conforme

$$B = \begin{bmatrix} 0 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & 1 & \cdots & \cdots & 0 \\ \vdots & & & & & \\ -a_{m-1} & -a_{m-2} & -a_{m-3} & \cdots & \cdots & -a_0 \end{bmatrix}, \tag{3.21}$$

sendo que a ordem da matriz B é $p^m - 1$, e logo $B^{p^m-1} = I$. Devem ser escolhidas outras matrizes B_i sobre GL_n, com ordens $m_1, m_2, \cdots, m_i$:

$$B = \begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_i \end{bmatrix}. \tag{3.22}$$

O expoente r é:

$$LCM\{(p^{m_1} - 1), (p^{m_2} - 1), (p^{m_3} - 1), \cdots, (p^{m_i} - 1)\}. \tag{3.23}$$

A matriz A pode ser obtida através da conjugação com uma matriz Y:

$$A = Y \cdot B \cdot Y^{-1}, \tag{3.24}$$

sendo que todas as matrizes utilizadas neste método devem ser não singulares, o que significa matrizes inversíveis.

Passo	**Nó A**	**Trocas**	**Nó B**
1	Os nós concordam num primo p, matriz A e num expoente r.		
2	Escolhe um número aleatório grande $1 \leq x < r$ e calcula $z_1 = A^x \ mod \ p$		Escolhe um número aleatório grande $1 \leq y < r$ e calcula $z_2 = r^y \ mod \ p$
3		$z_1 \Rightarrow$ $\Leftarrow z_2$	
4	Calcula sua chave $K = (z_2)^x \ mod \ p = (z_1)^y \ mod \ p$		Calcula sua chave $K = (z_1)^y \ mod \ p = (z_2)^x \ mod \ p$
1	ambos os nós obtêm a mesma chave $K = z_2^x = z_1^y = A^{xy}$.		

Figura 3.8: Protocolo Matrix Rings.

Em [9] são apresentados métodos para diminuir a complexidade do Matrix Rings. O primeiro método é o Baby-Step Giant-Step [3] que pode ser aplicável devido as matrizes utilizadas serem inversíveis e em $O(\sqrt{r})$ no pior caso encontrar-se-ia uma solução para o método Matrix Rings. Outra forma de diminuir a complexidade de tal método é quando o expoente r não é um número primo, podendo-se utilizar o CRT para encontrar uma solução, conforme descrito em [9]. Mas além disso, este método apresenta uma desvantagem que é a dimensão da matriz geradora A que pode ser $n \times n$ e como cada elemento de tal matriz está sobre a precisão do primo p, a quantidade de informação transmitida por cada nó em cada troca é de aproximadamente $n^2 \cdot p$. Considere uma matriz A de dimensão 5×5 e um número primo de 1024 bits de precisão. Cada nó enviaria $5^2 \cdot 1024 = 25600$ bits de informação, 25 vezes mais informação que o DH. Sendo que em certos cenários onde os dispositivos (nós) comunicantes possuem limitação de bateria (a transmissão consome muito mais que o processamento) não seria vantajoso utilizar tal método.

3.10 Protocolo de Acordo de Chaves Baseado em Matrizes Inversas

Passo	**Nó A**	**Trocas**	**Nó B**
1	Escolhe aleatoriamente uma matriz A de dimensão $k \times n$ e calcula sua inversa A^{-1}, calcula $z_1 = A \cdot A^{-1}$		
2		$z_1 \Rightarrow$	
3			Escolhe aleatoriamente uma matriz B de dimensão $m \times m$ e calcula sua inversa B^{-1}, calcula $z_2 = z_1 \cdot B$ e $z_3 = z_1 \cdot B \cdot B^{-1}$
4		$\Leftarrow z_2, z_3$	
5	Calcula $z_4 = A \cdot A^{-1} \cdot A \cdot B \cdot B^{-1} = A \cdot B \cdot B^{-1}$		
6		$z_4 \Rightarrow$	
7	Calcula a chave K como sendo $K = A \cdot A^{-1} \cdot A \cdot B = A \cdot B$.		Calcula a chave K como sendo $K = A \cdot B \cdot B^{-1} \cdot B = A \cdot B$.

Figura 3.9: Acordo de Chaves com Matrizes Inversas.

Este protocolo [15] se utiliza de matrizes não singulares para realizar um acordo chaves sem autenticação das partes com o intuito de dificultar a criptoanálise do DH. O objetivo dos autores [15] é utilizar matrizes com valores binários 1 *ou* 0. A dificuldade de se resolver tal método segundo os autores estaria na resolução de um sistema de equações lineares onde ter-se-ia mais incógnitas do que equações. As matrizes geradas neste protocolo estão sobre os campo finito Z_2^*. O funcionamento do protocolo pode ser visto na Figura 3.9.

Tabela 3.2: Comparação dos Protocolos

Protocolo	Utiliza Parâmetros Pré - Compartilhados	Quantidade de Parâmetros Pré - Compartilhados	Número de Trocas	Realiza Autenticação	Precisão da Chave
Diffie-Hellman	X	2	2		Primo p
ElGamal	X	3	1		Primo p
MTI	X	4	2		Primo p
Needham-Schroeder	X	2	3	X	Hash
CRTDH (n nós)	X	$3 \cdot n + 2$	$3 \cdot n$		-
SPAKE	X	6	2		Hash
S-3PAKE	X	7	5	X	Hash
DLP $\gamma \equiv \alpha^a \beta^b$	X	7	2		Primo p
Matrix Rings	X	Matriz, p e r	2		Primo p
Matriz Inversa	X	p	3		Primo p

No entanto segundo [16] pode-se reduzir significativamente a complexidade de tal protocolo devido ao fato do mesmo apresentar três trocas, se forem capturadas as mensagens z_1, z_2 e z_3, pode-se implementar um sistema linear onde seria facilmente encontrada a matriz B utilizada pelo nó B, ou mesmo poderia ser feito se fossem utilizadas as mensagens z_2, z_3 e z_4 afim de montar um sistema linear de equações para descobrir a matriz A utilizada pelo nó A, além disso é necessário multiplicar as trocas dos nós por outras matrizes auxiliares para facilitar na quebra do protocolo conforme mostrado em [16]. Dessa

maneira tal protocolo não dificultou a criptoanálise, e com isso não pode ser considerado como um protocolo substituto para o DH.

É apresentada uma tabela comparativa dos protocolos de acordo de chaves discutidos ao longos deste capítulo. A Tabela 3.2 apresenta o nome do protocolo, os parâmetros pré-compartilhados e a quantidade, o número de trocas realizadas por cada protocolo, a autenticação realizada ou não pelo protocolo e a precisão da chave de sessão gerada.

3.11 Conclusão

Este capítulo apresentou os principais protocolos que serviram de base para o estudo e implementação do DHA3 e mostrou as vantagens e desvantagens de protocolos com ou sem autenticação e as análises das complexidades para resolver tais protocolos. No próximo capítulo será apresentado o protocolo proposto o DHA3, onde será mostrado o tipo de cifrador utilizado, o funcionamento do protocolo, exemplo numérico, análise de segurança, aspectos criptográficos inerentes ao protocolo e aspectos da implementação prática do protocolo.

Capítulo 4

Protocolo de Acordo de Chaves Criptográficas DHA

Neste capítulo será apresentado o protocolo de acordo de chaves DHA e suas modificações que resultam na proposta final chamado de DHA3. Serão feitas análises de implementação, análises de segurança e análise do protótipo implementado.

4.1 Contextualização

Protocolos de acordo de chaves criptográficas possuem um importante papel no estabelecimento de canais seguros de comunicação entre nós comunicantes. A grande maioria destes protocolos utiliza-se de criptoanálise em funções exponenciais que recaem em resolver problemas de logaritmos discretos DLPs. No entanto, tais mecanismos necessitam trabalhar com números primos com uma grande magnitude, na ordem de 1024 a 2048 bits de precisão para que os algoritmos que resolvem problemas de DLP levem um tempo significativo (em termos computacionais) para encontrar uma solução.

Protocolos que são baseados em funções exponenciais por muitas vezes são chamados de protocolos de acordo de chaves exponenciais. Dentre eles podemos citar o DH, ElGamal, MTI e $\gamma \equiv \alpha^a \beta^b$ e outros. A complexidade para resolver tais protocolos está na aparente intratabilidade dos DLPs. Estes protocolos baseados em funções exponenciais podem ser resolvidos através de busca exaustiva (testar cada um dos possíveis valores que o expoente pode assumir), sendo que este ficará na magnitude do primo p. Em termos de implementações reais se trabalha com a magnitude de 1024 bits. Tal modo de encontrar a solução não pode ser considerado eficiente, por isso existem os algoritmos para solucionar o problema de maneira mais rápida. O melhor algoritmo para solucionar DLP é o NFS (*Number Field Sieve*) [12]. Algoritmos que solucionam DLP estão sobre a função de complexidade em tempo subexponencial $L_p[k, c]$, descrita pela equação 4.1, onde p é

primo sendo utilizado, c é uma constante e k é uma constante satisfazendo $0 \leq k \leq 1$. O algoritmo NFS possui uma complexidade $L_p[\frac{1}{3}, 1.923]$ [12].

$$L_p[k, c] = O(e^{(c+O(1))\cdot((\ln\ p)^k)\cdot((\ln\ \ln\ p)^{1-k})}) \tag{4.1}$$

4.2 Protocolo DHA1

No primeiro momento o protocolo DHA1 propôs substituir o gerador α do DH por uma matrix quadrada, Γ, onde os elementos estão sobre Z_p^* e cujo determinante é nulo, $det(\Gamma) = 0$. Assim, o procedimento para dois nós estabelecerem uma chave com o DHA1 são: Cada nó compartilha um primo grande p e uma matriz geradora Γ em (4.2), satisfazendo $3 \leq \alpha_{i,j} \leq p-2/i, j = 1, \cdots, n$ e $\alpha_{i,j} \neq 2^n/n \geq 1$, como mostrado pela Figura 4.1. Cada nó gera um valor secreto $1 \leq x \leq p-2$ (x para o nó A), e $1 \leq y \leq p-2$ (y para o nó B). Depois da geração destes valores secretos, os nós A e B realizarão o Passo 2.

$$\Gamma = \begin{bmatrix} \alpha_{1,1} & \cdots & \alpha_{1,n} \\ \vdots & \ddots & \vdots \\ \alpha_{n,1} & \cdots & \alpha_{n,n} \end{bmatrix} \tag{4.2}$$

No passo 3, o nó B recebe $z_1 = (\Gamma)^x$ e computa a chave compartilhada através do passo 4. A chave, $K_{n\times n}$, é uma matriz quadrada resultante, onde os elementos estão sob a magnitude de p. O nó A recebe $z_2 = (\Gamma)^y$ no passo 3 e computa a chave compartilhada através do passo 4. Então, cada nó ao final obterá a mesma chave como mostrado no passo 5, da Figura 4.1.

4.3 Exemplo Numérico DHA1

Para facilitar a compreensão do funcionamento do DHA1 serão definidos alguns valores para os parâmetros. Sendo $x = 5$, $y = 3$, $p = 257$ e matriz geradora com dimensão 2×2 dada por (4.3). Assim, o nó A realiza o cálculo de z_1 (4.4) e o nó B realiza o cálculo de z_2 (4.5), realizados no Passo 2 da Figura 4.1. Após estes procedimentos os nós realizam o Passo 3 da Figura 4.1. Ao receberem os parâmetros públicos cada nó irá calcular a chave compartilhada $K_{2\times2}$, o nó A calcula (4.6) e o nó B calcula (4.7) conforme o Passo 4 da Figura 4.1. Ao final ambos os nós obterão a mesma chave compartilhada, sendo que a chave gerada em cada nó é a mesma matriz, Passo 5 da Figura 4.1.

Passo	**Nó A**	**Trocas**	**Nó B**
1	Os nós concordam num primo p e matriz geradora Γ.		
2	Escolhe um número aleatório grande $1 \leq x \leq p-2$ e calcula $z_1 = (\Gamma)^x \ mod \ p$		Escolhe um número aleatório grande $1 \leq y \leq p-2$ e calcula $z_2 = (\Gamma)^y \ mod \ p$
3		$z_1 \Rightarrow$ $\Leftarrow z_2$	
4	Calcula sua chave $K_{n\times n} = (z_2)^x \ mod \ p = (z_1)^y \ mod \ p$		Calcula sua chave $K_{n\times n} = (z_1)^y \ mod \ p = (z_2)^x \ mod \ p$
5	Ambos os nós obtêm a mesma chave $K_{n\times n} = (\Gamma)^{xy} mod \ p$.		

Figura 4.1: Protocolo de acordo de chave DHA1.

$$\Gamma = \begin{bmatrix} 3 & 5 \\ 3 & 5 \end{bmatrix} \tag{4.3}$$

$$z_1 = \begin{bmatrix} 3 & 5 \\ 3 & 5 \end{bmatrix}^5 \ mod \ 257 = \begin{bmatrix} 209 & 177 \\ 209 & 177 \end{bmatrix} \tag{4.4}$$

$$z_2 = \begin{bmatrix} 3 & 5 \\ 3 & 5 \end{bmatrix}^3 \ mod \ 257 = \begin{bmatrix} 192 & 63 \\ 192 & 63 \end{bmatrix} \tag{4.5}$$

$$K_A = (z_2)^x \ mod \ 257 = \begin{bmatrix} 245 & 237 \\ 245 & 237 \end{bmatrix} \tag{4.6}$$

$$K_B = (z_1)^y \ mod \ 257 = \begin{bmatrix} 245 & 237 \\ 245 & 237 \end{bmatrix} \tag{4.7}$$

4.3.1 Análise de Segurança

Nas definições do protocolo DHA1, os elementos da matriz geradora devem ser maiores que 3 e não serem potências 2, tal condição foi estabelecida para evitar que os elementos da matriz geradora possam ser tratados ou visualizados como uma série. Por exemplo, quando todos os elementos são 2 encontra-se o n-ésima potência através da série $2^{2 \cdot n-1}$, sendo possível existir outras condições semelhantes a essas. Assim sendo o DHA1 seria a realização de 4 vezes o DH e não teria vantagens sobre o DH. Para que o DHA1 possua vantagens sobre o DH é preciso que o determinante seja nulo, e que os elementos

sejam diferentes entre si e que não sejam potências de 2.

Apesar de todas as considerações feitas anteriormente, tal proposta pode ser reduzida ao caso do DH, podendo assim ser resolvido como um DLP comum. Para isto, basta utilizar o exemplo numérico apresentado na seção anterior, onde as linhas da matriz são iguais, mas as colunas são diferentes. Vale ressaltar que para que se possa gerar matrizes com determinantes nulos independentemente do expoente sendo utilizado, é necessário que as linhas ou as colunas da matriz geradora sejam iguais. No contexto apresentado no exemplo numérico as linhas são iguais. Para realizar a redução basta somar os elementos diferentes, por exemplo, de z_1 gerando um $\beta = (\alpha_{1,1} + \alpha_{1,2}) \bmod p$, o resultado seria $\beta = (209+177) \bmod 257 = 129$, logo se fosse realizado a soma dos elementos diferentes da matriz geradora $(\alpha_{1,1} + \alpha_{1,2})$ e elevá-los ao expoente utilizado $x = 5$ obter-se-ia o mesmo resultado feito através da soma do elementos distintos de z_1, dado $(3+5)^8 \bmod 257 = 129$. Tal análise foi obtida através da observação de que os resultados obtidos em z_1 e z_2 são nada mais do que "produtos notáveis", e por esse motivo tal proposta não dificultou a criptoanálise do DH. No entanto se fosse possível trabalhar de forma que independentemente do expoente (segredo de cada nó) a matriz gerada para a troca (z_1 e z_2) obtivesse 4 valores distintos de forma a gerar uma matriz com determinante nulo, dessa forma tal proposta dificultaria a criptoanálise do DH. Mas, através dos estudos realizados até o presente momento não foi possível encontrar uma matriz geradora com quatro elementos diferentes que sempre gerasse, após a exponenciação matricial sobre módulo, matrizes com determinantes nulos.

4.4 DHA2

Com o objetivo de diminuir a quantidade de informação que está sendo transmitida entre os nós no acordo de chave e diminuir também o processamento, sem no entanto diminuir a complexidade para resolver o DHA1, foi feita uma modificação no DHA1 para que operasse com uma matriz Γ com dimensão finita 1×2, logo, apenas dois elementos são trasmitidos a cada troca realizada entre os nós. Esta proposta foi definida como DHA2, o 2 é devido utilizar apenas dois *domain parameters*. Assim, os procedimentos para dois nós estabelecerem uma chave com o DHA2 são: Cada nó compartilha um primo grande p e uma matriz geradora Γ em (4.8), satisfazendo $1 \leq \alpha_{i,j} \leq p-2$ para todo $i, j = 1, \cdots, n$, como mostrado pela Figura 4.2. Cada nó gera um valor secreto $1 \leq x \leq p-2$ (x para o nó A), e $1 \leq y \leq p-2$ (y para o nó B). Depois da geração destes valores secretos, os nós A e B realizarão o passo 2.

$$\Gamma = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} \end{bmatrix} \tag{4.8}$$

No passo 3, o nó B recebe $z_1 = (\Gamma)^x$ e calcula a chave compartilhada através do passo 4. A chave, $K_{1\times2}$, onde os elementos estão sobre a precisão de p. O nó A recebe $z_2 = (\Gamma)^y$ no passo 3 e calcula a chave compartilhada através do passo 4. Então, cada nó ao final obterá a mesma chave como mostrado no passo 5 da Figura 4.2.

No entanto, para o DHA2 a exponenciação da matriz geradora não segue a mesma regra que é feita pelo DHA1, onde é realizada a multiplicação sucessiva da matriz geradora base Γ. No DHA2 é realizado um mapeamento para executar a exponenciação. Para simplificar a explanação deste mapeamento será considerado o caso em que a matriz geradora seja elevada à potência de 2, Γ^2. Os elementos da matriz geradora serão renomeados para $a = \alpha_{1,1}$ e $b = \alpha_{1,2}$, sendo que este mapeamento permite um conjunto variado de possibilidades. Serão apresentados apenas *quatro* tipos de mapeamento, sendo que tais mapeamentos foram baseados nos "protutos notáveis", por exemplo: produto das somas, produto da soma pela diferença, produto das diferenças, entre outros, conforme mostrado abaixo:

1. $[a, b]^2 \ mod \ p \Longrightarrow [a^2 + a \cdot b, \ a \cdot b + b^2] \ mod \ p$
2. $[a, b]^2 \ mod \ p \Longrightarrow [a^2 + b^2, a \cdot b + a \cdot b] \ mod \ p$
3. $[a, -b]^2 \ mod \ p \Longrightarrow [a^2 - a \cdot b, \ -a \cdot b + b^2] \ mod \ p$
4. $[a, -b]^2 \ mod \ p \Longrightarrow [a^2 + b^2, \ -a \cdot b - a \cdot b] \ mod \ p$

Passo	**Nó A**	**Trocas**	**Nó B**
1	Os nós concordam num primo p e matriz geradora Γ.		
2	Escolhe um número aleatório grande $1 \leq x \leq p-2$ e calcula $z_1 = (\Gamma)^x \ mod \ p$		Escolhe um número aleatório grande $1 \leq y \leq p-2$ e calcula $z_2 = (\Gamma)^y \ mod \ p$
3		$z_1 \Rightarrow \Leftarrow z_2$	
4	Calcula sua chave $K_{1\times2} = (z_2)^x \ mod \ p = (z_1)^y \ mod \ p$		Calcula sua chave $K_{1\times2} = (z_1)^y \ mod \ p = (z_2)^x \ mod \ p$
5	Ambos os nós obtêm a mesma chave $K_{1\times2} = (\Gamma)^{xy} mod \ p$.		

Figura 4.2: Protocolo de acordo de chave DHA2.

4.5 Exemplo Numérico DHA2

Para facilitar a compreensão do funcionamento do DHA2 serão definidos alguns valores para os parâmetros. Sendo $x = 5$, $y = 4$, $p = 257$, os elementos da matriz geradora sendo $a = 2$ e $b = 3$, e matriz geradora com dimensão 1×2 dada por (4.9). Assim, o nó A realiza o cálculo de z_1 (4.10) e o nó B realiza o cálculo de z_2 (4.11), realizados no passo 2 da Figura 4.2. Após estes procedimentos os nós realizam o passo 3 da Figura 4.2. Ao receberem os parâmetros públicos cada nó irá calcular a chave compartilhada $K_{1\times2}$, o nó A calcula (4.12) e o nó B calcula (4.13) conforme o passo 4 da Figura 4.2. Ao final, ambos os nós obterão a mesma chave compartilhada, sendo que a chave gerada em cada nó é a mesma matriz, passo 5 Figura 4.2.

$$\Gamma = \begin{bmatrix} 2 & 3 \end{bmatrix} \tag{4.9}$$

$$z_1 = \begin{bmatrix} 2 & 3 \end{bmatrix}^5 \ mod\ 257 = \begin{bmatrix} 20 & 21 \end{bmatrix} \tag{4.10}$$

$$z_2 = \begin{bmatrix} 2 & 3 \end{bmatrix}^4 \ mod\ 257 = \begin{bmatrix} 56 & 55 \end{bmatrix} \tag{4.11}$$

$$K_A = (z_2)^x \ mod\ 257 = \begin{bmatrix} 250 & 249 \end{bmatrix} \tag{4.12}$$

$$K_B = (z_1)^y \ mod\ 257 = \begin{bmatrix} 250 & 249 \end{bmatrix} \tag{4.13}$$

4.5.1 Análise de Segurança DHA2

De forma semelhante ao que foi discutido sobre a segurança do DHA1, o mesmo pode ser aplicado ao DHA2. Também nesta proposta pode-se reduzir o DHA2 ao DH, sendo assim necessário realizar apenas um DLP, não dificultando a criptoanálise do DH. Da mesma forma utilizando-se do exemplo numérico apresentado na seção anterior, considerando a troca z_1 e somando os elementos resultantes $(\alpha_{1,1} + \alpha_{1,2}) \bmod p$ têm-se $\beta = (20 + 21) \bmod 257 = 41$ que é o mesmo resultado que se fosse realizada a seguinte operação: $\beta = (2 + 3)^5 \mod 257 = 41$. Ao observar-se cada uma das propostas de forma isolada, DHA1 e DHA2, notou-se que não são eficientes, pois recaem no DH. A partir da junção destas duas propostas (DHA1 e DHA2) resolveu-se implementar uma nova proposta chamada DHA3, com algumas diferenças matemáticas para dificultar a criptoanálise em relação ao DH e será apresentada a partir da próxima seção.

4.6 Protocolo de Acordo de Chaves DHA3

Esta proposta une as características do DHA1 e do DHA2 com o intuito de realmente dificultar a criptoanálise evitando assim, que o DHA3 possa ser reduzido a complexidade do DH. Nesta nova proposta ambos os nós compartilham um primo p uma matriz quadrada Γ de dimensão $n \times n$ e um vetor υ com dimensão $1 \times n$. Cada nó gera um valor secreto $1 \leq x \leq p-2$ (x para o nó A), e $1 \leq y \leq p-2$ (y para o nó B). Será considerado o caso onde a matriz Γ possui dimensão 2×2 (4.14) e o vetor υ (4.15) possui dimensão 1×2. Os elementos da matriz Γ devem ser diferentes entre si, tal que $\alpha_{1,1} \neq \alpha_{1,2} \neq \alpha_{2,1} \neq \alpha_{2,2}$. O mesmo vale para os elementos do vetor υ sendo $\alpha_{1,1} \neq \alpha_{1,2}$. Os elementos da matriz e do vetor devem estar entre $1 \leq \alpha_{ij} \leq p-2$, mas para garantir que tais elementos sejam diferentes e que não hajam relações óbvias entre os mesmos, tais elementos podem ser números primos.

$$\Gamma = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} \\ \alpha_{2,1} & \alpha_{2,2} \end{bmatrix} \tag{4.14}$$

$$\upsilon = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} \end{bmatrix} \tag{4.15}$$

O funcionamento do protocolo DHA3 é semelhante ao DH, DHA1 e DHA2 como descrito abaixo:

Passo 1 : Na Figura 4.3, ambos os nós compartilham uma matriz Γ, um vetor υ e um primo p.

Passo 2 : Neste passo têm-se a diferença em relação as outras propostas (DH, DHA1, DHA2) é que cada nó irá realizar apenas uma exponenciação que é a geração da matriz m_1 e m_2 na Figura 4.3, sendo que tais matrizes são valores mantidos em segredo pelos nós, além disso se a matriz gerada após a exponenciação e a aplicação do módulo gerar uma matriz com valores nulos, deverá então, gerar um novo valor aleatório para o expoente e realizar a operação novamente (Cada nó idividualmente deverá realizar este teste antes de gerar os *domain parameters*).

Passo 3 : Os nós multiplicam as suas matrizes pelo vetor compartilhado υ.

Passo 4 : Os nós realizam as trocas dos *domains parameters* z_1 e z_2 que são vetores de dimensão 1×2.

Passo 5 : Ambos os nós ao receberem o parâmetro público do outro nó, irão calcular o vetor chave que é a multiplicação dos *domain parameters* z_1 e z_2 pelas matrizes m_1

e m_2.

Passo 6 : Ao final, ambos os nós obtêm o mesmo vetor chave.

Passo	**Nó A**	**Trocas**	**Nó B**
1	Os nós concordam num primo p e matriz geradora Γ e num vetor υ.		
2	Escolhe um número aleatório grande $1 \leq x \leq p-2$ e calcula $m_1 = (\Gamma)^x \ mod \ p$		Escolhe um número aleatório grande $1 \leq y \leq p-2$ e calcula $m_2 = (\Gamma)^y \ mod \ p$
3	Calcula $z_1 = (\upsilon \cdot m_1) \bmod p$		Calcula $z_2 = (\upsilon \cdot m_2) \bmod p$
4		$z_1 \Rightarrow \Leftarrow z_2$	
5	Calcula sua chave $K_{1\times 2} = (z_2 \cdot m_1) \bmod p$		Calcula sua chave $K_{1\times 2} = (z_1 \cdot m_2) \bmod p = (z_2)^x \ mod \ p$
6	Ambos os nós obtêm a mesma chave $K_{1\times 2} = (\upsilon \cdot (\Gamma)^{xy}) mod \ p$.		

Figura 4.3: Protocolo de acordo de chave DHA3.

4.7 Exemplo Numérico DHA3

Para facilitar a compreensão do funcionamento do DHA3 serão definidos alguns valores para os parâmetros. Sendo $x = 5$, $y = 4$, $p = 257$ e matriz geradora com dimensão 2×2 dada por (4.16), o vetor *upsilon* dado por (4.17). Assim, o nó A realiza o cálculo de m_1 (4.18) e o nó B realiza o cálculo de m_2 (4.19), realizados no passo 2 da Figura 4.3. Após estes procedimentos os nós realizam o passo 3 da Figura 4.3, onde é calculado o z_1 em (4.20) e o z_2 em (4.21). Ao receberem os parâmetros públicos cada nó irá calcular a chave compartilhada $K_{2\times 2}$, o nó A calcula (4.22) e o nó B calcula (4.23) conforme o passo 5 da Figura 4.3. Ao final ambos os nós obterão a mesma chave compartilhada, sendo que a chave gerada em cada nó é o mesmo vetor, passo 6 da Figura 4.3.

$$Passo\ 1 \ \Rightarrow \ \Gamma = \begin{bmatrix} 2 & 3 \\ 5 & 7 \end{bmatrix} \tag{4.16}$$

$$Passo\ 1 \ \Rightarrow \ \upsilon = \begin{bmatrix} 11 & 13 \end{bmatrix} \tag{4.17}$$

$$Passo\ 2\ \Rightarrow\ m_1 = \begin{bmatrix} 2 & 3 \\ 3 & 7 \end{bmatrix}^5 \mod\ 257 = \begin{bmatrix} 222 & 122 \\ 101 & 66 \end{bmatrix} \tag{4.18}$$

$$Passo\ 2\ \Rightarrow\ m_2 = \begin{bmatrix} 2 & 3 \\ 5 & 7 \end{bmatrix}^4 \mod\ 257 = \begin{bmatrix} 34 & 185 \\ 137 & 171 \end{bmatrix} \tag{4.19}$$

$$Passo\ 3\ \Rightarrow\ z_1 = (\upsilon \cdot m_1) \mod\ 257 = \begin{bmatrix} 157 & 34 \end{bmatrix} \tag{4.20}$$

$$Passo\ 3\ \Rightarrow\ z_2 = (\upsilon \cdot m_2) \mod\ 257 = \begin{bmatrix} 99 & 146 \end{bmatrix} \tag{4.21}$$

$$Passo\ 5\ \Rightarrow\ K_A = (z_2 \cdot m_1) \mod\ 257 = \begin{bmatrix} 230 & 164 \end{bmatrix} \tag{4.22}$$

$$Passo\ 5\ \Rightarrow\ K_B = (z_1 \cdot m_2) \mod\ 257 = \begin{bmatrix} 230 & 164 \end{bmatrix} \tag{4.23}$$

4.8 Análise de Segurança DHA3

Nas definições do protocolo DHA3, os elementos da matriz geradora devem ser diferentes entre si para garantir que as matrizes m_1 e m_2 não apresentem relações óbvias com a matriz Γ, para evitar o que acontecia no DHA1 e no DHA2. A utilização do vetor υ foi feito para que algoritmos que resolvem DLPs com complexidade subexponencial não pudessem ser efetivos sobre o DHA3. Como por exemplo, o algoritmo IC (*Index Calculus*) e o NFS [3]. O motivo que faz com que tais algoritmos não sejam efetivos sobre o DHA3 é o fato de realizar os procedimentos apenas sobre uma única base (seria o caso de trabalhar apenas com a matriz Γ), no entanto existem duas bases a matriz Γ e o vetor υ. Outro ponto é realizar a decomposição dos vetores gerados nos procedimentos do DHA3 em vetores parciais. Como pode ser visto no Capítulo 2 desta dissertação, no Passo 4.2 do algoritmo Index - Calculus, sendo que não é uma operação tão trivial de ser realizada no contexto de vetores. De forma semelhante é realizado no NFS para resolver o DH, contudo ainda deve ser investigado a possível utilização do NFS para reduzir a complexidade do DHA3, mas não é o escopo deste trabalho investigar tais contextos.

O algoritmo clássico para resolver DLP é o algoritmo Baby-Step Giant-Step [3], que é baseado na seguinte observação: se $\beta = \alpha^a$, então pode-se escrever $a = im + j$, onde $0 \leq i, j \leq m$ e $m = \sqrt{p}$. Portanto, $\alpha^a = \alpha^{im}\alpha^j$, implica em $\beta(\alpha^{-m})^i = \alpha^j$. Este método pode ser aplicado no DHA3 porque bastaria realizar o seguinte procedimento:

sendo $\beta = \upsilon \cdot \Gamma^a$, $\beta \cdot (\alpha^{-m})^i = \upsilon \cdot \alpha^j$. Os outros métodos que possuem complexidade de solução $O(\sqrt{p})$ também podem ser adaptados para resolver o DHA3.

A complexidade para resolver o DH por busca exaustiva é $O(2^\eta)$, onde η é a magnitude do número primo p, que produz no pior caso 2^η multiplicações para o DH. Com o intuito de aplicar busca exaustiva no DHA3 são necessárias $((n^3+n^2)\cdot 2^\eta)$ multiplicações e também $((n^2\cdot(n-1)+2))$ adições. O parâmetro, n, define a dimensão da matriz geradora quadrada $(n \times n)$. Assim, considerando apenas as multiplicações, a complexidade para encontrar uma solução para o DHA3 é $O((n^3+n^2)\cdot 2^\eta)$. O número de parâmetros principais (*domain parameters*) trocados por cada nó pelo DHA3 é n, devido ser necessário enviar todos os elementos do vetor resultante. Então com o objetivo de produzir um menor *overhead* no protocolo de acordo de chaves, a dimensão ótima para a matriz quadrada é 2×2 e o vetor de 1×2, que gera a troca de dois *domain parameters*.

Apesar de que uma matriz com dimensões maiores cause um aumento na complexidade para resolver o DHA3, ainda assim a magnitude do primo p é a mais impactante na complexidade do protocolo. No entanto se, por exemplo, fosse utilizada uma matriz com dimensão 3×3 e um vetor com 1×3 seria necessário enviar 3 *domain parameters* que por sua vez encontram-se na magnitude do primo p, e 3 vezes a magnitude do primo acarretaria num overhead muito maior que o DH clássico. Uma matriz 2×2 e vetor 1×2 necessitaria enviar apenas 2 *domain parameters* a cada troca, e obter-se-ia praticamente o mesmo nível de segurança que uma 3×3, pois se substituirmos o valor das dimensões nas equações de complexidade do DHA3, teríamos para $n = 2$, $O(12 \cdot 2^\eta)$ e para $n = 3$, ter-se-ia $O(36 \cdot 2^\eta)$. Com $n = 3$ o DHA3 seria 3 vezes mais complexo que para $n = 2$, no entanto transmitiria 50% a mais de informação o que não tornaria o protocolo vantajoso se comparado com o DH.

Supondo o DHA3 sendo resolvido pelo BSGS que possui tempo de execução no pior caso de $O(\sqrt{p})$, se considerarmos o fato do DHA3 estar trabalhando com uma matriz de dimensão 2×2 e um vetor de 1×2 e transmitindo a mesma quantidade de informação que o DH, o DHA3 precisaria trabalhar com um primo de 512 bits e o DH com um primo de 1024. Se aplicássemos o BSGS no DHA3, de forma simplista o DHA3 seria resolvido no pior caso em $O(\sqrt{2^{512}}) = O(2^{256})$; se fosse aplicado o IC ou NFS no DH, sendo este operando com um primo de 1024 bits, seria resolvido em $O(2^{132})$ no pior caso. Assim, o DHA3 torna-se $O(\frac{2^{256}}{2^{132}}) = O(2^{124})$ mais complexo que o DH.

O grande desafio do protocolo DHA3 é: dado p, Γ, υ e $(\upsilon \cdot \Gamma^a) \bmod p$ encontrar a. Em termos de algoritmos para resolver DLP o desafio do protocolo DHA3 é encontrar uma forma de adaptar o NFS ou o IC para que se possa realizar a decomposição de vetores geradas pelo protocolo, sendo que baseado nos estudos feitos até presente momento ainda

não foi encontrado nenhum método que realize tal operação de forma satisfatória. Assim, o método mais eficaz para resolver o DHA3 é através dos métodos com resolução em $O(\sqrt{p})$. Deste modo, o DH precisa trabalhar com primos com aproximadamente quatro vezes a magnitude que o DHA3 utiliza, para garantir aproxidamente o mesmo grau de complexidade a solução por DLP.

4.9 Aspectos da Implementação do DHA3

O protocolo DHA3 foi implementado em Linguagem C Posix com o compilador GNU/GCC [17] através do uso da biblioteca GNU/GMP [18]. Esta biblioteca permite trabalhar com números de alta precisão [19]. O elemento crucial na implementação do protocolo foi na escolha do melhor método de exponenciação a ser utilizado levando em consideração o fato da base a ser exponenciada, não ser apenas um valor inteiro, mas sim uma matriz de valores inteiros positivos não nulos. Devido ao uso de matrizes, o algoritmo 14.79 da página 615 de [3] foi o que melhor se enquadrou ao ser adaptado para o uso com matriz.

O algoritmo 14.79 da página 615 de [3] foi modificado para ser aplicado ao DHA3 conforme é mostrado no **Algoritmo 1**. Sendo que o expoente e é tratado na sua forma binária $(e_t e_{t-1} \cdots e_1 e_0)_2$, t é o bit mais significante, I_n é a matriz identidade de ordem n, $A_{n\times n}$ é matriz que armazena os resultados obtidos ao longo do processo de exponenciação.

Algoritmo 1 Exponenciação Binária da Esquerda para Direita

ENTRADA: Γ e o expoente positivo e inteiro $e = (e_t e_{t-1} \cdots e_1 e_0)_2$

SAÍDA: Γ^e

1. $A_{n\times n} \leftarrow I_n$.
2. Para i de t até 0 faça:
   2.1 $A_{n\times n} \leftarrow A_{n\times n} \cdot A_{n\times n}$.
   2.2 Se $e_i = 1$, então $A_{n\times n} \leftarrow A_{n\times n} \cdot \Gamma$.
3. Retorne $(A_{n\times n})$

Existem outros algoritmos que realizam a expoenenciação de forma mais rápida que o 14.79. No entanto necessitam de mais variáveis e operações de pré - cálculos de elementos utilizados ao longo do processo de exponenciação. Devido a estes e outros motivos adotou-se o algoritmo 14.79 por ser simples de implementar e requerer apenas uma variável de armazenamento (no caso do protocolo DHA3 uma matriz $n \times n$).

Devido aos aspectos de segurança discutidos na seção anterior mostrarem que a dimensão ideal para a matriz geradora ser 2×2, para garantir o menor *overhead* possível sem no entanto diminuir a complexidade, será utilizada sempre a dimensão 2×2 ao longo

das próximas seções. A implementação em linguagem C do DHA3 é apresentada pelo **Algoritmo 2**, onde é mostrado apenas o procedimento realizado por um nó, neste caso o nó A.

Algoritmo 2 Pseudo-Código do DHA3

1. Os nós compartilham uma matriz Geradora $\Gamma_{2\times2}$, o vetor υ e um primo p.
2. O nó A escolhe um valor inteiro x tal que $3 \leq x \leq p-2$.
3. Nó A usa o Algoritmo 1 para calcular $m_1 = (\Gamma_{2\times2})^x$.
4. O nó A calcula $z_1 = (\upsilon \cdot m_1) \bmod p$ 5. Envia z_1 para o nó B.
6. Recebe z_2 do nó B.
7. O Nó A calcula $K_{1\times2} = (z_2 \cdot m_1) \bmod p$.
8. $K_{1\times2}$ é a chave de seção compartilhada entre os nós.

Devido ao protocolo DHA3 gerar um vetor chave e não um valor inteiro como chave de seção, e devido a esta característica intrínseca do DHA3, ambos os nós poderão manipular os elementos resultantes do vetor chave $K_{1\times2}$. O DHA3 permite que os elementos resultantes da chave gerada sejam combinados de três maneiras, sendo $\oplus$ operação binária de XOR:

1. Gerar uma chave K que é a concatenação dos elementos gerados, dada por $K = \alpha_{1,1}||\alpha_{1,2}$, ainda pode-se aplicar esta chave gerada numa função de *hash* e obter o hash dos elementos, afim de obter uma única chave $K = h(\alpha_{1,1}||\alpha_{1,2})$.

2. Outra forma seria gerar o hash de cada um dos elementos gerando assim duas chaves $K_1 = h(\alpha_{1,1})$ e $K_2 = h(\alpha_{1,2})$.

3. A terceira forma seria realizar a operação binária XOR nos elementos resultantes afim de gerar-se apenas uma chave $K = \alpha_{1,1} \oplus \alpha_{1,2}$, ou ainda aplicar uma função de hash na chave K, gerando $K = h(\alpha_{1,1} \oplus \alpha_{1,2})$.

A utilização de funções de *hash* na geração das chaves é feita com o intuito de gerar uma chave no tamanho padrão utilizado pelo AES (*Advanced Encryption Standard*), que é um sistema criptográfico simétrico e opera com chaves de 128, 192 e 256 bits.

4.10 Análise de Desempenho DHA3

Foi implementada em Linguagem C uma versão do DH e do DHA3 usando a biblioteca GMP e utilizando o **Algoritmo 1** como método de exponenciação para o DHA3, e foi utilizado o algoritmo 14.79 da página 615 de [3] para o DH para que pudesse haver um bom nível de comparação entre o DHA3 e o DH. Considerando o DH trabalhando

com um primo de 1024 bits de magnitude e utilizando o NFS para resolver tal DLP, utilizando (4.1), logo $L_{2^{1024}}[\frac{1}{3}, 1.923] \cong O(2^{132})$. Baseado nisto, bastaria para o DHA3 trabalhar com um primo de 512 bits de precisão para que o DHA3 transmita a mesma quantidade de informação que o DH. Considere como exemplo a matriz geradora dada por (4.24) e o vetor υ dado por (4.25).

$$\Gamma = \begin{bmatrix} 2 & 3 \\ 5 & 7 \end{bmatrix} \tag{4.24}$$

$$\Gamma = \begin{bmatrix} 11 & 13 \end{bmatrix} \tag{4.25}$$

Com o primo p com 512 bits de magnitude para o DHA3 e 1024 bits para o DH, o DHA3 levou **2 vezes menos** tempo de processamento do que o DH e transmitiu 1024 bits de informação em cada troca a mesma quantidade que o DH, mas o DHA3 por ser resolvido pelo BSGS sua complexidade de solução foi para $O(2^{256})$, já o DH sendo resolvido pelo NFS passou a ter a complexidade de solução $O(2^{132})$, com isso o DHA3 tornou-se $\frac{2^{256}}{2^{132}} = O(2^{124})$ mais complexo que o DH.

Agora considerando o DHA3 trabalhando com um primo de 264 bits para que ao ser resolvido pelo BSGS, apresente a mesma complexidade de resolução $O(2^{132})$ que a do DH, sendo resolvido pelo NFS, contudo o DH deve operar com um primo de 1024. Nestas condições o DHA3 levou **5 vezes menos** tempo de processamento do que o DH e transmitiu apenas $2 \times 264 = 528$ bits de informação a cada troca.

4.11 Considerações

Ao longo do desenvolvimento do DHA1 e do DHA2, observou-se a possibilidade de implementar-se o DHA3, com o intuito de dificultar a criptoanálise para que o DHA3 não pudesse ser reduzido ao DH. A maioria dos protocolos que tentaram aumentar a complexidade do DH, alguns destes apresentados no Capítulo 3 desta dissertação, se ativeram em manipular os expoentes "segredos" ou utilizaram de assinatura digital ou mesmo de certificados para aumentar a segurança do DH. Mas a proposta apresentada nesta dissertação de mestrado trouxe uma nova área de estudo que a utilização de matrizes e vetores como geradores para dificultar a criptoanálise do DH.

O ElGamal [3] nada mais é do que o DH e a chave gerada é utilizada diretamente para se fazer a cifragem. O ElGamal utiliza-se da complexidade para resolver um DLP ao realizar a sua cifragem. O MTI é um protocolo que gera dois *domain parameters* extras

que são armazenados num servidor autenticado, os quais são utilizados para aumentar a complexidade dos expoentes, além de gerar autenticação mútua das chaves.

O protocolo CRTDH [14] faz uma junção das características do DH e do CRT para aumentar a complexidade do protocolo, no entanto, este necessita realizar muitas trocas para obter as chaves compartilhadas entre os membros do grupo, uma vez que o objetivo deste protocolo é ser aplicado em redes sem fio no modo Ad-hoc. O protocolo SPAKE [7] utiliza uma senha pré-compartilhada entre os nós, para então realizar os procedimentos da geração da chave compartilhada. Esta "senha" é aplicada no expoente "segredo", mas tal protocolo acaba se tornando vulnerável ao ataque de "passphrase". Uma vez que cada caractere da senha terá no máximo 102 possibilidades, enquanto que um byte de uma chave criptográfica apresenta o universo de 256 possibilidades. Além deste protocolo necessitar de um servidor para realizar a autenticação dos nós.

O protocolo S-3PAKE também utiliza "senhas" entre o servidor e os nós [7], e justifica-se que é um protocolo inovador por não necessitar de um servidor de chaves públicas, no entanto, tal protocolo necessita de um servidor confiável para autenticar as partes comunicantes. Neste caso haverá apenas a troca de um servidor por outro (chaves públicas por um servidor confiável) de forma que a proposta não poderia ser considerada inovadora. Além disso, toda a "segurança" deste protocolo está depositada no servidor confiável, uma vez que este servidor possa vir a ser comprometido, todo o protocolo estará comprometido. Tal condição não ocorre com o DHA3 uma vez que não necessita de servidor confiável nem de servidores de chaves públicas, pois é um protocolo de acordo de chaves criptográficas sem autenticação das partes.

O protocolo Matrix Rings [8] foi a primeira proposta que utilizou uma matriz como gerador base, no entanto tal proposta apresenta o inconveniente de ser necessário gerar um polinômio $f(x)$ não redutível, ter mais um parâmetro público extra que é o expoente r (ordem da matriz geradora) compartilhado pelos nós. A principal desvantagem de tal protocolo é que a matriz geradora possui uma dimensão maior que 2×2, conforme mostrado no Capítulo 3. Trabalhar com matrizes com dimensões maiores que 2×2 não torna-se vantajoso devido à quantidade de informação que é necessária ser trocada entre os nós. Supondo uma matriz com dimensão 6×6 e um primo p com 1024 bits de precisão, em cada troca realizada pelos nós seriam necessários enviar $n^2 \cdot p$ dados que neste caso seria $36 \cdot 1024 = 36864$ bits, o que seria enviar 36 vezes mais informação que o DH. Logo tal proposta em contextos onde os nós (dispositivos) apresentem limitação de transmissão de dados pelo canal de comunicação, não seriam beneficiados. Contudo, o protocolo DHA3 gera menos dados que o DH e garante um nível considerável de segurança.

O protocolo que trabalha com matrizes inversas [15] tentou aumentar a complexi-

dade de resolução do DH, mas o mesmo não atentou para o fato de que o sistema gerado pelo seu protocolo poderia ser resolvido através da resolução de sistemas lineares sobre campos finitos, como demonstrado em [16]. Além da proposta apresentada em [15] trabalhar apenas com matrizes sobre campo binário, logo o universo de busca fica muito limitado. O mesmo não ocorre no DHA3, pois este utiliza matrizes e vetores sobre campos finitos, sendo que o primo utilizado apresenta uma magnitude considerável, gerando assim um universo de busca grande.

O grande desafio desta dissertação foi encontrar um protocolo pelo qual não pudesse ser mais resolvido através de algoritmos que tem complexidade de resolução em tempo subexponencial como o IC e o NFS, por isso é que a junção de matrizes com vetores tiveram um importante papel na contrução do protocolo DHA3. A seguir, será apresentada uma tabela comparativa entre o DH e o DHA3, onde serão comparados a magnitude do primo utilizado pelos protocolos, a complexidade, o tamanho da chave gerada e a quantidade de informação transmitida na duas trocas realizadas pelos nós. Na Tabela 4.1 a complexidade do DH foi feita considerando o algoritmo NFS para resolver o DLP através da equação 4.1 e considerando o NFS com $L_p[\frac{1}{3}, 1.923]$, e para o DHA3 foi considerado que o mesmo seja resolvido pelo algoritmo BSGS.

Tabela 4.1: Comparação dos protocolos DH e DHA3

DH				
Primo p (bits)	768	1024	2048	4096
Chave (bits)	768	1024	2048	4096
Tráfego (bits)	1536	2048	4096	8192
Complexidade	$O(2^{117})$	$O(2^{132})$	$O(2^{178})$	$O(2^{238})$
DHA3				
Primo p (bits)	384	512	1024	2048
Chaves (bits)	$2 \times 384 = 768$	$2 \times 512 = 1024$	$2 \times 1024 = 2048$	$2 \times 2048 = 4096$
Tráfego (bits)	1536	2048	4096	8192
Complexidade	$O(2^{192})$	$O(2^{256})$	$O(2^{512})$	$O(2^{1024})$
DHA3 Com Menor Tráfego				
Primo p (bits)	234	264	356	476
Chaves (bits)	$2 \times 234 = 468$	$2 \times 264 = 528$	$2 \times 356 = 712$	$2 \times 476 = 952$
Tráfego (bits)	936	1056	1424	1904
Complexidade	$O(2^{117})$	$O(2^{132})$	$O(2^{178})$	$O(2^{238})$

4.12 Conclusão

Neste Capítulo foram apresentados os protocolos propostos DHA1, DHA2 e o DHA3, bem como os quesitos para sua implementação, análise desempenho, análise da

complexidade, vantagens sobre o DH, mostrando que apenas o DHA3 trouxe benefícios em termos criptográficos. A Tabela 4.1 sumariza a comparação entre os protocolos DH e DHA3 para mostrar que o protocolo proposto nesta dissertação trabalha com primos menores que o DH e alcança uma complexidade de resolução maior. No próximo Capítulo será apresentada a conclusão final desta dissertação de mestrado.

Capítulo 5

Conclusão

Esta dissertação de mestrado versou sobre protocolos de acordo de chaves criptográficas principalmente nos baseados em problemas de DLP, como DH, ElGamal, MTI, SPAKE, S-3PAKE, Matrix Rings e outros. Ao longo desta pesquisa e estudo, pode-se observar que os protocolos de acordo de chaves clássicos atinham-se em aumentar a complexidade dos expoentes ao qual o gerador base (no caso do DH α) era elevado, pois tais valores são os "segredos" armazenados pelos nós. Contudo ao longo da análise, desenvolvimento e estudo do protocolo DHA3, pode-se notar que o elemento mais importante era o gerador base e não somente os expoentes, e através desta observação é que pode-se implementar o DHA3.

A primeira premissa que foi idealizada para o DHA3 foi a de não precisar de um servidor de chaves públicas ou mesmo um servidor autêntico para garantir a segurança do protocolo. O S-3PAKE necessita de um servidor autêntico para garantir a segurança do mesmo, contudo se este servidor for invadido toda a segurança do protocolo estará comprometida. O protocolo DHA3 independe de servidores o que facilitaria a sua aplicabilidade em ambientes formados por redes no modo *ad-hoc* onde a "autenticação" é de responsabilidade dos membros da rede. Esta premissa foi alcançada como pode ser visto ao longo do Capítulo 4, desta dissertação.

O segundo objetivo deste protocolo era de transmitir menos informação que o DH e mesmo assim garantir ao menos a mesma complexidade que o DH clássico. Tal objetivo foi alcançado através do uso de matrizes e vetores como geradores base. Conforme foi analisado para o DHA3 pode-se considerar o ataque pelo BSGS como um dos métodos mais eficazes para resolver o protocolo proposto nesta dissertação. Uma vez que o grande desafio apresentado pelo protocolo proposto será encontrar um método eficiente para realizar a decomposição de vetores para que possa então vir a ser resolvido por algoritmos mais eficientes que o BSGS. Tal argumentação é apresentada, pois uma vez encontrado

um meio de realizar tal decomposição este poderia ser adaptado ao método NFS para reduzir a complexidade do protocolo DHA3 e com isso evitar que a solução mais efetiva seja através do BSGS, por exemplo.

O terceiro objetivo desta dissertação foi criar um protocolo que diminuísse o processamento computacional para gerar os parâmetros pré-compartilhados. Conforme visto no protocolo Matrix Rings apresentado no Capítulo 3 desta dissertação, este necessita gerar pelo menos um polinômio irredutível, calcular a matriz geradora A, encontrar um conjunto de matrizes B_i afim de poder calcular A e determinar a ordem de A através do encontro do expoente r, tal que $A^r \ mod \ p = I$ gere uma matriz identidade. Tais procedimentos geram um consumo computacional significativo para os nós comunicantes. O protocolo DHA3 não precisa de uma quantidade tão significativa de procedimentos como no Matrix Rings, basta no caso do DHA3 utilizar uma matriz e um vetor com elementos distintos entre si, por exemplo, números primos e com uma dimensão de 2×2 e 1×2, que é bem menos oneroso que outros protocolos.

Devido ao protocolo DHA3 ser embasado no funcionamento do DH, o protocolo proposto nesta dissertação também apresenta característica intrínseca de ser analisado numa complexidade em tempo exponencial, conforme visto no Capítulo 4 sobre Análise de Segurança. Como dito anteriormente a primeira proposta de se alcançar maior complexidade computacional para o DH utilizando-se para isto matrizes, foi apresentada pelo protocolo Matrix Rings, no entanto a maior contribuição apresentada nesta dissertação é o fato de utilizar matrizes e vetores, diferentemente de outros protocolos que utilizam-se apenas de matrizes.

Como possíveis aplicabilidades do protocolo proposto nesta dissertação de mestrado são: servir de protocolo de acordo de chaves criptográficas anônimo em ambientes que operem no modo *ad-hoc*; substituir o DH no TLS com o intuito de diminuir o custo computacional e a quantidade de informação sendo transmitida pelos nós; poder ser aplicável aos dispositivos móveis que utilizam o TLS para gerar o par de chaves criptográficas como é o caso do WAP2 utilizado nas redes de telefonia móvel; ser aplicado como protocolo de acordo de chaves em redes sem fio ou mesmo cabeadas de forma geral.

O grande benefício gerado por essa dissertação foi trazer para a sociedade científica mais uma área de estudo e análise sobre criptografia, utilizando-se de matrizes e vetores, uma vez que o protocolo DHA3 diminuiu a quantidade de informação trocada entre os nós, exigiu números primos menores que o DH e seus correlatos; garantiu um nível de segurança considerável em relação ao DH, ElGamal e MTI; e devido requerer números primos de menor magnitude acabou gerando menos custo computacional e sendo facilmente implementável.

Referências Bibliográficas

[1] R. Housley, W. Ford, W. Polk, and D. Solo. Rfc 3280 - internet x.509 public key infrastruture certificate and certificate revogation list (crl).

[2] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. Spins: Security protocols for sensor networks. *Wireless Networks* 8^{th}, pages 521–534, 2002.

[3] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography - HAC*, chapter 2, 3, 12, pages 59–60, 103–114, 515–519. CRC Press, Inc., 1997.

[4] Bruce Schneier. *Applied Cryptography.* John Willey and Sons, 2nd edition, 1996.

[5] Sunil Kumar Kashyap, Birendra Kumar Sharma, and Amitabh Banerjee. A cryptosystem based on dlp $\gamma \equiv \alpha^a \beta^b \ mod\ p$. *International Journal of Network Security*, vol. 3(1):pp. 95–100, July 2006.

[6] Michal Sramka. Cryptanalysis of the cryptosystem based on dlp $\gamma = \alpha^a \beta^b$. *International Journal of Network Security*, vol. 6(1):80–81, January 2008.

[7] R. Lu and C. Zhenfu. Simple three-party exchange protocol. In Elsevier, editor, *Computer & Security*, number 26, pages 94–97, 2007.

[8] R. W. K. Odoni, V. Varadharajan, and P. W. Sanders. Public key distribution in matrix rings. *Electronic Letters*, 20(9):386–387, April 1984.

[9] V. Varadharajan and R. W. K. Odoni. Security of public key distribution in matrix rings. *Electronic Letters*, 22(1):46–47, January 1986.

[10] J. Rothe. Some facets of complexity theory and cryptography: A five-lecture tutorial. In *ACM computing surveys*, volume 34, pages 504–549. ACM Press, New Yourk, December 2003.

[11] PGP Pretty Good Privacy. The international pgp home page. http://www.pgpi.org/, May 2007.

[12] Oliver Schirokauer. Discrete logarithms and local units. *Philosophical Transactions of the Royal Society of London*, vol. A 345(1676):409–423, November 1993.

[13] David S. Dummit and Richard M. Foote. *Abstract Algebra*, volume 1 of *ISBN: 0-13-005562-X*. Prentice-Hall International Editions, United States of America, 1st edition, 1991.

[14] R. K. Balachandran, B. Ramamurthy, and X. Zou. An efficient key agreement scheme for secure communications in wireless ad hoc networks. In *IEEE International Conference in Communications*, number 1, pages 1123–1127, 2005.

[15] E. Dawson and C. Wu. Key agreement scheme based on generalised inverses of matrices. *Electronic Letters*, 33(14):1210–1211, July 1997.

[16] A. M. Youssef and S. E. Tavares. Cryptanalysis os "key agreement scheme based on generalised inverses of matrices". *Electronic Letters*, 33(21):1777–1778, October 1997.

[17] GNU GCC 4.0.2. A compiler for language c, c++, fortran and java. http://gcc.gnu.org/, July 2007.

[18] GNU GMP. Gnu multiple precision artithmetic library for language c. http://www.swox.com/gmp/, July 2007.

[19] D. H. Bailay. High-precision floating-point arithmetic in scientific computing. *IEEE Computing in Science & Engeneering*, pages 54–61, May/June 2005.