NKey: Um Protocolo Para o Gerenciamento de Chaves de Grupo

Carlos Rocha
carlosgr@natalnet.br

Tatiana Tavares
tati@natalnet.br

Benjamín Bedregal
bedregal@dimap.ufrn.br

Guido Lemos
guido@natalnet.br

Laboratório NatalNet
DIMAp – Departamento de Informática e Matemática Aplicada
UFRN – Universidade Federal do Rio Grande do Norte
Campus Universitário – Lagoa Nova – 59072-970, Natal – RN - Brasil

Resumo

Atualmente muitas aplicações são fundamentadas no modelo de comunicação baseado em grupos. Essa tendência foi impulsionada pela grande popularidade e diversidade de serviços interativos e colaborativos que se firmaram com o advento da Internet. Todavia, outras necessidades foram se agregando ao desenvolvimento dessas aplicações, com o propósito de torná-las mais robustas, seguras e confiáveis. Neste sentido existem esforços que visam possibilitar a implementação de mecanismos que permitam o estabelecimento de um canal seguro entre os membros de um grupo. Este artigo apresenta uma extensão do algoritmo Diffie-Hellman [07] que permite calcular e gerenciar chaves assimétricas para grupos de usuários. Uma vez calculadas, as chaves podem ser usadas para viabilizar a comunicação segura entre os membros do grupo. Para validar a extensão proposta foi desenvolvido um simulador e são apresentadas provas matemáticas que validam sua corretude.

Palavras-chave: Segurança, Criptografia, Comunicação segura de Grupos.

Abstract

Nowadays many applications are based on group communication. This trend appeared with the advent of the Internet, especially to provide support to collaborative applications. Make these applications more robust, safe and reliable became a new requirement. In this way efforts are being done to implement secure group communication between the group members. This paper presents a proposal based in Diffie-Hellman algorithm [07]. This algorithm provides support to key exchange between users communication. Our proposal extends Diffie-Hellman algorithm to generate and to manage secure asymmetric group keys. In order to validate the proposed extension was implemented a simulator. We too present the mathematical model and proof which validate this extension.

Keywords: Security, Cryptography, Secure group communication.

1. Introdução

Atualmente muitas aplicações são fundamentadas no modelo de comunicação baseado em grupos. Essa tendência foi impulsionada pela grande popularidade e diversidade de serviços interativos e colaborativos que se firmaram com o advento da Internet. Todavia, outras necessidades foram se agregando ao desenvolvimento dessas aplicações, com o propósito de torná-las mais robustas, seguras e confiáveis.

Aplicações como White-boards, edição colaborativa de documentos, e sistemas de controle remoto distribuídos necessitam que a comunicação entre as entidades que compõem o sistema seja segura. Estas classes de aplicações são conhecidas como aplicações colaborativas, pois a interação entre as entidades que formam o sistema ocorre em um modelo *peer-to-peer* [01], onde os mesmos serviços são oferecidos e requisitados pelas entidades que formam o sistema. Esta característica é o principal diferencial em relação a aplicações baseadas em um modelo *client-server* [01] como a distribuição de vídeo digital.

Para dar um suporte adequado às aplicações colaborativas, vários esforços tem sido desenvolvidos no sentido de possibilitar a implementação de mecanismos usados para garantir a confidencialidade ou integridade dos documentos trocados por membros de um grupo. Este artigo apresenta um conjunto de protocolos, baseados em uma extensão do algoritmo Diffie-Hellman [07], que permite o cálculo e a gerencia de chaves assimétricas para grupos de usuários. O modelo proposto foi validado através de uma demonstração matemática, e pela implementação de um simulador.

A seções 2 e 3 procuram ilustrar os conceitos fundamentais ligados à comunicação de grupo e gerencia de chaves. A seguir, apresentamos nossa proposta para estender o algoritmo Diffie-Hellman para o cálculo e gerência de chaves de Grupo, bem como o modelo matemático e o simulador que validam o modelo. Apresentamos então uma comparação com alguns dos principais esforços de pesquisa nesta área. Finalmente, apresentamos as nossas conclusões e perspectivas de trabalhos futuros.

2. Comunicação em Grupo

Na visão de Cosquer [04] uma definição conceitual de grupo o descreve como um conjunto de entidades passivas (dados) ou ativas (processos) relacionadas, e que pode ser endereçado como sendo uma unidade.
De maneira geral pode-se definir um grupo através dos seguintes conceitos:

- Grupo
- Membro
- Mensagem
- Processos de Controle

Um **grupo** pode ser representado por um conjunto ***G***, composto de *X* elementos $\{E_1,...,E_X\}$**.** ***G(x)*** é uma forma abstrata de representar o conjunto de todos os elementos de ***G***. Assim, pode-se endereçar todos os elementos de ***G*** por ***G(X)*** sendo possível enviar uma mensagem para todos os elementos sem nomear explicitamente a ocorrência de cada um deles.

Um **membro** de um grupo é representado por cada elemento de ***G*** que interage com outros elementos através de troca de mensagens.

Uma **mensagem** *m* compreende o envio de alguma informação, a partir de um elemento E_n de ***G*** para ***G(X)***. Desta forma, temos:

$$E_n \xrightarrow{m} G(X)$$

Já os **processos de controle** são responsáveis pelas atividades de gerência dessa sistemática de grupo. Segundo Tanenbaum [09], a implementação dessa característica define políticas para os serviços e para a comunicação interprocesso de um grupo. As políticas podem ser transparentes para um membro do grupo. Esse comportamento categoriza os grupos de acordo com sua estrutura interna em:

- Grupos Fechados ou Abertos. São chamados de grupos fechados aqueles onde somente os membros podem enviar mensagens para o grupo. Assim, processos que não pertençam ao grupo não podem enviar mensagens para o grupo, embora possam fazê-lo para membros individuais do grupo. Os grupos abertos permitem que qualquer processo seja capaz de enviar mensagens para o grupo.

- Grupos Não-Hierárquicos ou Hierárquicos. Nos grupos não hierárquicos, existe a igualdade entre os processos, onde nenhum processo é superior e todas as decisões são tomadas coletivamente. Nos grupos hierárquicos os processos podem ser organizados hierarquicamente, havendo um processo coordenador com missão de gerenciar as tarefas dos demais processos no grupo.

- Grupos Estáticos ou Dinâmicos. São chamados de grupos estáticos aqueles onde não é permitida a inclusão ou exclusão de membros (os membros não podem deixá-lo ou novos membros não podem juntar-se a ele). Nestes ambientes, tampouco são permitidas a criação e eliminação dinâmica de grupos. Nos grupos dinâmicos novos grupos podem ser criados e grupos antigos podem ser destruídos, além dos processos poderem, a qualquer momento, juntar-se ao grupo ou sair dele.

Um ambiente para comunicação de grupo representa e pode ser definido a partir dos membros do grupo e dos processos que controlam a interação entre eles. Mensagens para o grupo *(multicas*t) correspondem a mensagens de um membro para todos os demais.

3. Gerenciamento de Chaves

O gerenciamento de chaves é definido em [05,06] como o conjunto de técnicas e procedimentos que suportam o estabelecimento e manutenção de chaves entre partes autorizadas. O gerenciamento de chaves compreende:

1. Inicialização de usuários do sistema: Nesta fase devem ser gerados valores que serão utilizados na inicialização do algoritmo a ser utilizado para o cálculo das chaves.

2. Geração, distribuição, e instalação das chaves: Esta é a principal função de um sistema de gerenciamento de chaves. Uma vez geradas, as chaves podem ser utilizadas para prover qualquer serviço criptográfico (criptografia, autenticação, assinatura digital, etc) entre as partes autorizadas;

3. Controle de uso das chaves, atualização e revogação das chaves: Um sistema de gerenciamento de chaves também deve ser capaz de realizar tarefas de manutenção das chaves gerenciadas, tais como a sua atualização caso o conjunto das partes autorizadas a ter acesso a uma chave seja alterado.

Estas funcionalidades devem ser providas de forma a minimizar os recursos computacionais envolvidos na sua execução, tais como número de chaves a ser armazenado por cada usuário, número de cálculos a serem realizados para a geração de uma nova chave, etc.

Em um sistema de comunicação de grupo, a chave tem o papel de identificar cada membro e o próprio grupo. Para um grupo *G(x)* de *E* membros, cada membro detém, pelo menos, uma chave *K'* proprietária e uma chave *K''* pública, além da chave K conhecida como a chave do grupo. Desta forma, em um grupo *G(x)* tem-se *2E +1* chaves. A chave *K* é de conhecimento exclusivo dos membros do grupo e determina se um membro E_n tem permissão de interagir com os outros membros do grupo.

4. Extensão do Diffie-Hellman para o Gerenciamento de Chaves Seguras de Grupo

Nesta seção é proposto um protocolo para o gerenciamento de chaves de grupo baseado em uma extensão do algoritmo Diffie-Hellman chamado **NKey**. Para tanto, apresentamos inicialmente o algoritmo Diffie-Hellman, na sua forma original, utilizado para o intercâmbio de chaves entre dois usuários. Então, discutimos uma adaptação do algoritmo original bem como os protocolos desenvolvidos para suportar a entrada e saída de membros, bem como o recálculo das chaves de um grupo.

4.1. Algoritmo Diffie-Hellman

O Diffie-Hellman [07] é um algoritmo para intercâmbio de chaves que possibilita a dois usuários calcularem um par de chaves, uma secreta e uma pública, que podem ser então utilizadas para criptografar ou assinar mensagens. O algoritmo foi concebido considerando a dificuldade de se calcular um logaritmo discreto.

Podemos definir o algoritmo Diffie-Hellman como segue:

p	Número primo
r	Raiz primitiva de p
X_N	Chave privada do usuário N
Y_N	Chave pública do usuário N
K	Chave de sessão

Inicialmente são gerados dois valores públicos: Um número primo *p* e uma raiz primitiva *r* de *p* ou seja, um número cujas potências de *1* a *p-1* módulo *p* geram todos os inteiros de *1* a *p-1*. Isto é, se *r* é uma raiz primitiva de *p* então $r^1 \bmod p$, $r^2 \bmod p$, ..., $r^{p-1} \bmod p$ são distintos e consistem nos inteiros de *1* a *p-1* em qualquer ordem. Para qualquer inteiro *i* (entre *1* e *p-1*) e uma raiz primitiva *r* de um primo *p*, podemos encontrar um único expoente *e* tal que:

$$i = r^e \bmod p$$

O expoente *e* é conhecido como um logaritmo discreto de *i* para a base *r*, módulo *p*.
Supondo que dois usuários U_1 e U_2 desejam calcular uma chave, o usuário U_1 escolhe aleatoriamente um inteiro $X_1 < p$, e calcula $Y_1 = r^{X1} \bmod p$. Do mesmo modo o usuário U_2, de forma independente, escolhe aleatoriamente um inteiro $X_2 < p$, e calcula $Y_2 = r^{X2} \bmod p$. O valores X_1 e X_2 são mantidos como valores privados por cada um dos usuários, enquanto que os valores Y_1 e Y_2 são públicos e serão trocados entre os dois. O usuário U_1 calcula a chave $K= Y_2^{X1} \bmod p$. De forma semelhante U_2 calcula a chave $K = Y_1^{X2} \bmod p$. Neste ponto, os dois usuários terão calculado a chave secreta que será usada em futuras trocas de mensagens. Estes dois cálculos produzem um valor idêntico como é demonstrado a seguir:

$K = Y_2^{X1} \bmod p$
$K = (r^{X2} \bmod p)^{X1} \bmod p$
$K = (r^{X2})^{X1} \bmod p$
$K = r^{X2\,X1} \bmod p$
$K = (r^{X1})^{X2} \bmod p$
$K = (r^{X1} \bmod p)^{X2} \bmod p$
$K = Y_1^{X2} \bmod p$

Um possível atacante possuiria, no pior caso, os seguintes elementos: *p*, *r*, Y_1 e Y_2. Mesmo possuindo estes quatro valores ele é forçado a realizar o cálculo de um logaritmo discreto para determinar a chave K. Supondo um ataque ao usuário U_1, o atacante teria que calcular a chave secreta do usuário U_1 (X_1) a partir da equação $Y_1 = r^{X1} \bmod p$, ou seja, o atacante teria que calcular um logaritmo discreto de Y_1 para a base *r* modulo *p* [08]. Após o cálculo desse valor ele calcularia a chave K da mesma maneira que o usuário U_1 o fez.
O artifício matemático que fundamenta a robustez do Diffie-Hellman é a inviabilidade computacional de se calcular um logaritmo discreto. Enquanto é relativamente fácil calcular exponenciais e módulos de um número primo, é praticamente impossível calcular um logaritmo discreto, principalmente quando usamos números primos grandes.
Considere a seguinte equação

$$y = g^x \bmod p$$

Dados g, x e p é relativamente fácil calcular y. No pior caso, serão realizadas x multiplicações repetidas. Todavia dados y, g e p é impraticável o cálculo de x (logaritmo discreto). Atualmente, um dos algoritmos mais rápidos para cálculo de logaritmos discretos é da ordem de:

$$e^{((\ln p)1/3\ \ln(\ln p))2/3}$$

Que é computacionalmente inviável para um número primo p grande como os utilizados pelo algoritmo Diffie-Hellman.

4.2. NKey: Um protocolo para Gerenciamento de Chaves de Grupo

Nesta seção é apresentado o protocolo **NKey**. Como primeiro exemplo, um grupo de três usuários é mostrado na **figura 1.** No **NKey** um grupo é representado por uma árvore binária com as seguintes características:

- Os membros do grupo são representados por folhas (U_1, U_2 e U_3);
- Nós intermediários armazenam chaves intermediárias ($K_{1,2}$) geradas no decorrer da formação do grupo. Estas chaves podem ser utilizadas para promover comunicação segura entre os descendentes na árvore;
- A raiz da árvore detém a chave atual do grupo ($K_{1,2,3}$) de conhecimento de todos os membros do grupo.

Para lidar com a dinamicidade do grupo, o **NKey** define protocolos para entrada e saída de membros do grupo, bem como o protocolo para recálculo de chaves do grupo. Estes protocolos serão apresentados a seguir.

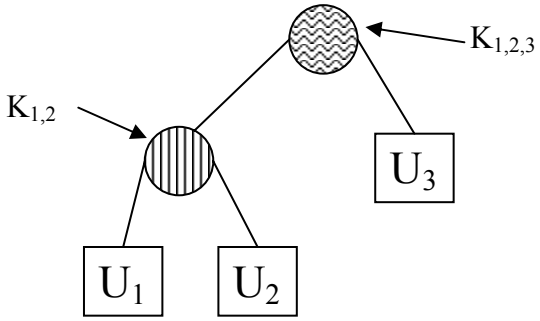

Figura 1: Árvore com três usuários

A **figura 2** exemplifica o protocolo de entrada de membros no grupo. Na **figura 2a** é mostrada a entrada do membro U_4 no grupo **figura 1**. Para tanto, é criado um novo nó intermediário $K_{3,4}$. O membro U_3 passa a ser filho do novo nó intermediário, bem como o novo membro do grupo U_4. As operações são realizadas desta maneira visando manter a árvore tão balanceada quanto possível.

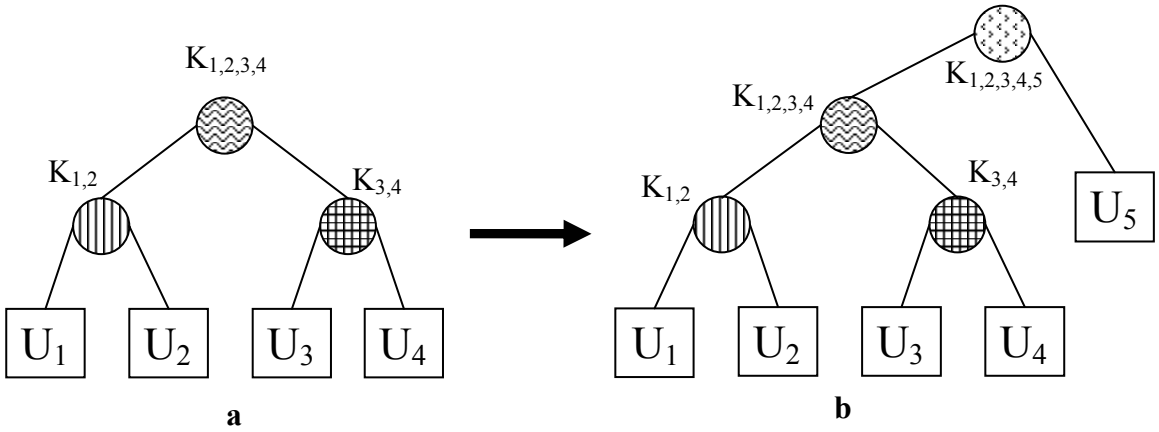

Figura 2: Protocolo de entrada no grupo

A entrada do membro U_5, na **figura 2b**, ocorre de forma semelhante. Como a árvore anterior à entrada de U_5 é uma árvore completa com altura h=3, o novo nó intermediário será a nova raiz da árvore, que passará a ter altura h=4. Novos membros que entrem no grupo serão inseridos como filhos da sub-árvore direita de $K_{1,2,3,4,5}$ até termos uma nova árvore completa de altura h=4. Neste momento, a entrada de um novo membro provocaria a criação de uma nova raiz e a altura da árvore passaria para h=5, e assim por diante.

Como conseqüência do protocolo de entrada de um novo membro no grupo, é disparada a execução do protocolo de recálculo de chaves. O objetivo deste protocolo é realizar o recalculo da chave do grupo, bem como das chaves intermediárias que necessitem ser atualizadas. Devido aos algoritmos de criptografia assimétrica serem computacionalmente pesados, este protocolo deve ser implementado de forma a minimizar o número de operações a serem realizadas para o recálculo da chave do grupo. Neste ponto, a estruturação do grupo em uma árvore possui um papel fundamental na diminuição do número total de cálculos a serem realizados quando da entrada de um novo membro no grupo.

Caso o grupo não possua nenhum tipo de estruturação, podendo ser representado por uma lista, por exemplo, a entrada de um novo membro, irá provocar o processo de recálculo da chave em todos os antigos membros do grupo. Neste caso, cada entrada de um novo membro no grupo provoca um total de *n* recálculos de chaves, onde *n* representa o número de membros do grupo. Com a estruturação do grupo em uma árvore binária, apenas os nós ascendentes diretos do novo membro disparam o processo de recálculo da chave, enviando o resultado para os seus descendentes através de uma mensagem segura. Neste caso, serão necessários no pior caso *h* recálculos, onde *h* representa a altura da árvore. A **Figura 3** mostra os pontos onde haverá recálculo, quando da entrada do usuário U_8 no grupo.

De forma genérica, o protocolo de recálculo de chaves que ocorre em cada um desses pontos funciona como segue:

Inicialização: O primeiro membro a entrar no grupo será responsável pela geração dos valores públicos que servirão como inicialização para o algoritmo, ou seja, o número primo *p* e uma raiz primitiva *r* de *p*. Adicionalmente ele também escolhe $X_1 < p$ e calcula $Y_1 = r^{X1} \bmod p$.

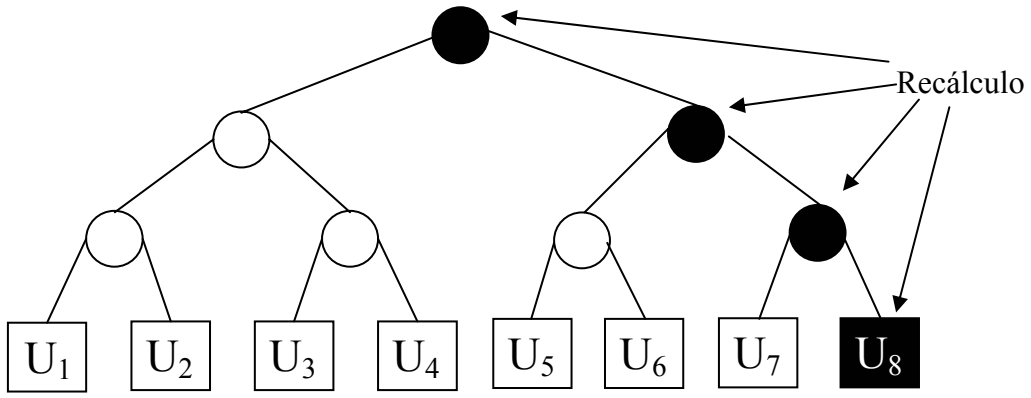

Figura 3: Pontos de recálculo de chave

Na entrada do segundo usuário o cálculo das chaves ocorre da seguinte maneira:
Usuário U_2:

- Recebe *p*, *r* e Y_1
- Escolhe $X_2 < p$ e calcula $Y_2 = r^{X2} \bmod p$; $X_{1,2} = Y_1^{X2} \bmod p$; $Y_{1,2}=r^{X12} \bmod p$.

Usuário U_1:

- Recebe Y_2
- Calcula $X_{1,2} = Y_2^{X1} \bmod p$; $Y_{1,2}=r^{X12} \bmod p$.

Passo 1: Na entrada do terceiro usuário ocorrerá a seguinte seqüência de eventos:
Usuário U_3:

- Recebe p, r e $Y_{1,2}$
- Escolhe $X_3 < p$ e calcula $Y_3 = r^{X3} \bmod p$; $X_{1,2,3} = Y_{1,2}^{X3} \bmod p$; $Y_{1,2,3}=r^{X1,2,3} \bmod p$.

Raiz:

- Recebe Y_3
- Calcula $X_{1,2,3} = Y_3^{X1,2} \bmod p$; $Y_{1,2,3} = r^{X1,2,3} \bmod p$; envia $X_{1,2,3}$ e $Y_{1,2,3}$ para os seus descendentes na árvore.

Passo N-1: De uma forma mais genérica, na entrada do usuário n+1, o recálculo ocorrerá da seguinte maneira:
Raiz:

- Recebe Y_{n+1}
- Calcula $X_{1,...,n+1} = Y_{n+1}^{X1,...,n} \bmod p$; $Y_{1,...,n+1} = r^{X1,...,n+1} \bmod p$; envia $X_{1,...,n+1}$ e $Y_{1,...,n+1}$ para os seus descendentes na árvore.

Note que apenas os usuários que já estavam no grupo podem realizar este cálculo, uma vez que eles têm acesso à chave privada do grupo $X_{1,...,n}$ e à chave pública Y_{n+1} do novo usuário. Já o novo usuário (U_{n+1}) cuja chave privada é X_{n+1} calcula as chaves do grupo da seguinte maneira:

- Recebe *p*, *r* e $Y_{1,..,n}$
- Escolhe $X_{n+1} < p$ e calcula $Y_{n+1} = r^{Xn+1} \bmod p$; $X_{1,...,n+1} = Y_{1,..,n}^{Xn+1} \bmod p$; $Y_{1,...,n+1} = r^{X1,...,n+1} \bmod p$.

Para comprovar a corretude matemática do protocolo, mostraremos que $Y_{n+1}^{X1,...,n} \bmod p = Y_{1,..,n}^{Xn+1} \bmod p$

$Y_{n+1}^{X\,1,...,n} \bmod p = (r^{X1,..,n} \bmod p)^{Xn+1} \bmod p$
$Y_{n+1}^{X\,1,...,n} \bmod p = r^{X1,..,n\ Xn+1} \bmod p$
$Y_{n+1}^{X\,1,...,n} \bmod p = r^{Xn+1\ X1,..,n} \bmod p$
$Y_{n+1}^{X\,1,...,n} \bmod p = (r^{\ Xn+1} \bmod p)^{X1,..,n} \bmod p$
$Y_{n+1}^{X\,1,...,n} \bmod p = Y_{n+1}^{X1,...,n} \bmod p$

Esta prova valida a corretude matemática do protocolo de recálculo de chaves.

O recálculo das chaves intermediárias ocorre de forma análoga ao cálculo da chave do grupo, considerando-se que cada nó intermediário pode ser considerado como uma raiz da sua sub-árvore formada por todos os seus descendentes.

O protocolo de saída de um membro do grupo é mostrado na **figura 4**. Para a saída do membro U_3 o nó intermediário $K_{3,4}$ é removido da árvore, e o membro U_4 é promovido a filho de $K_{1,2,4}$. Para efeito de recálculo de chave o membro U_4 "simula" uma nova entrada no grupo, provocando o disparo do protocolo de recálculo de chaves. O recálculo ocorre então como descrito anteriormente.

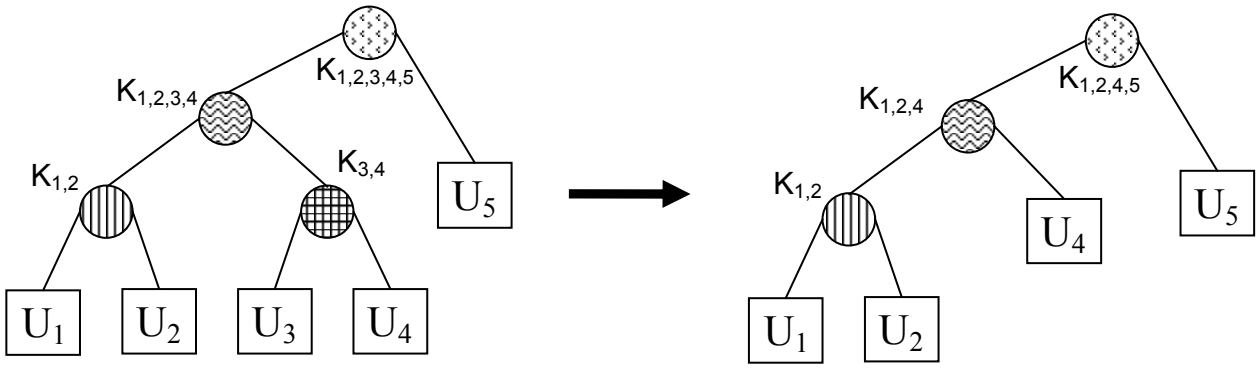

Figura 4: Protocolo de saída do grupo

O grupo formado no decorrer do processo pode ser classificado como:

- Fechado, pois não ocorre comunicação entre usuários externos e o grupo;
- Não hierárquico, pois não há a figura de um controlador do grupo;
- Dinâmico, pois os protocolos apresentados suportam as operações de entrada e saída de membros do grupo em qualquer instante do tempo.

4.3. Simulador do NKey

Para validar o funcionamento dos protocolos de entrada e saída do grupo, bem como do protocolo de recálculo de chaves foi implementado um simulador que executa os três protocolos propostos. O simulador é constituído principalmente pelos seguintes módulos:

- *primo, raizprimitiva*: As classes contidas nestes módulos, implementam a fase de inicialização do protocolo de recálculo de chaves, disponibilizando métodos para, por exemplo, a geração de números primos e da base exponencial que serão usadas nas exponenciações seguintes.
- *fastexp*: Este módulo implementa, principalmente, um método que calcula de forma eficiente uma exponenciacão da forma $Y = a^b \bmod n$. Os valores *a, b* e *n* (um número primo) servem de entrada para o algoritmo. O cálculo de Y deve ocorrer de forma linear com o tamanho de *a, b* e *n*.
- *membro:* Esta classe implementa um membro do grupo. No momento de sua criação, cada instância dessa classe dispara uma *thread* que ira executar enquanto a instância existir. Esta *thread* verifica periodicamente a estrutura da árvore. Alterações na árvore disparam a execução de um método que determina o tipo de evento ocorrido, bem como se esta instância deve executar o protocolo de recálculo de chaves. Em caso afirmativo, os novos valores públicos são requisitados e o recálculo das chaves é realizado.
- *arvore:* Esta classe implementa a árvore que representa a estrutura do grupo atual. Ela oferece métodos que disparam os protocolos de entrada e de saída de um membro do grupo. Neste simulador, todos os nós da árvore, não apenas as folhas, possuem uma mesma implementação, instanciando a classe *membro*. A diferenciação entre nós intermediários e membros efetivos do grupo é feita pelo fato dos membros efetivos serem as folhas da árvore. Esta classe disponibiliza métodos que implementam os protocolos de entrada e saída do grupo. O protocolo de entrada do grupo primeiramente determina o ponto de inserção do novo membro na árvore, após isso, ele cria um novo nó contendo uma nova instância da classe membro. Todos os outros nós da árvore perceberão a mudança na sua estrutura, e caso necessário, irão iniciar de forma independente o protocolo de recálculo de chaves. O protocolo de saída do grupo primeiramente remove o nó da árvore, realizando os ajustes na estrutura da árvore que se tornarem necessários. Neste momento, se

houverem folhas cuja posição na árvore foi alterada elas irão realizar uma entrada no grupo. Como no protocolo de entrada de membros no grupo, caso necessário, cada membro ira iniciar de forma independente o protocolo de recálculo de chaves.

O simulador foi desenvolvido na linguagem de programação C++, sobre o sistema operacional Linux. Um diagrama de classes resumido do simulador é mostrado na **figura 5**.

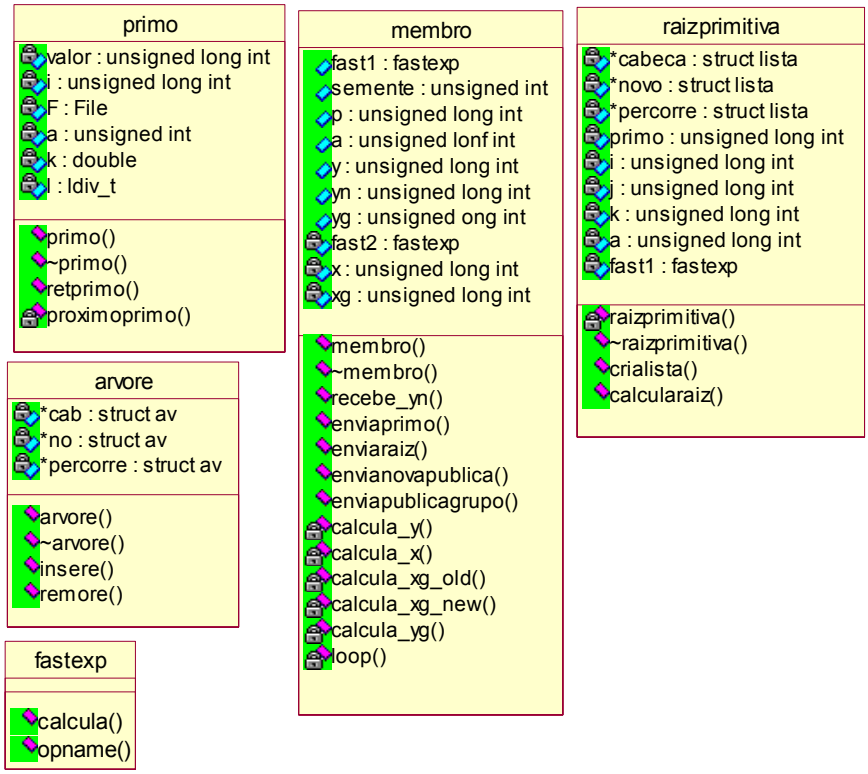

Figura 5: Classes do NKey

5. Comparação com trabalhos relacionados

Apesar de ser uma área relativamente nova, podemos encontrar na literatura diversos trabalhos [02,03,11,12,13,14,15,16,17,18] que abordam o assunto de comunicação segura de grupos. Uma comparação entre estes trabalhos deve levar em consideração pelo menos, os seguintes parâmetros:

- Inicialização: Número de operações a serem realizadas para a inicialização do algoritmo;
- Número de recálculos: Número total de chaves a serem recalculadas quando da alteração da estrutura do grupo, ou seja quando da entrada ou saída de algum membro;
- Mensagens: Número de mensagens a serem trocadas quando da alteração da estrutura do grupo;
- Centralização: O esquema possui ou não um controlador de grupo.

Um dos primeiros trabalhos desenvolvidos nesta área foi desenvolvido por M. STEINER, G. TSUDIK e M. WAIDNER [02], apresentando uma série de protocolos considerados extensões naturais do Diffie-Hellman para o caso de *n*, ao invés de dois parceiros. De forma semelhante ao Diffie-Hellman tradicional, todos os membros do grupo compartilham um número primo *p* e uma base exponencial *a*. Os protocolos apresentados são denominados de GDH.1, GDH.2 e GDH.3. O protocolo GDH.1 consiste de dois estágios *upflow* e *downflow*. O propósito do estágio de upflow é coletar uma contribuição de cada membro do grupo. O estágio de *downflow* é iniciado imediatamente após o término do *upflow* e se caracteriza pela realização de *i* exponenciações, onde i representa o número de membros do grupo, por cada membro M_i do grupo, uma para calcular K_n (a chave do grupo) e i-1 para prover os valores necessários para os membros seguintes. Os protocolos GDH.2 e GDH.3 são extensões do GDH.1 que visam diminuir o número de exponenciações a serem realizados por cada membro do grupo. Os protocolos propostos neste trabalho assumem que o conjunto aproximado de membros que irá formar o grupo é conhecido antes do início da execução do protocolo. No entanto, protocolos para entrada e saída de um membro do grupo também são definidos.

O conjunto de protocolos GDH [02] é o único trabalho que possui uma fase de inicialização. Isto se deve à premissa que o conjunto aproximado dos membros que irão formar o grupo são conhecidos no momento de sua formação. Para o GDH.3, no processo de inicialização são necessárias 5*n*-6 exponenciações e 2*n*-1 trocas de mensagens, sendo *n* o número de membros que irão formar inicialmente o grupo. Após a fase de inicialização a entrada de cada novo membro requer 2*n*-1 exponenciações e 2 trocas de mensagens.

O TGDH (Tree Based Group Diffie-Hellman) [03] baseia-se numa abordagem que integra duas importantes tendências no gerenciamento de chaves de grupo: O uso de árvores binárias, para eficientemente calcular e atualizar as chaves do grupo; e o algoritmo Diffie-Hellman para o cálculo de chaves comprovadamente seguras. O TGDH define protocolos para a entrada (Join), e para a saída (Leave) de um membro do grupo. Após cada uma dessas operações cada membro tem a responsabilidade de atualizar a sua visão da árvore de forma independente. Nestes protocolos um membro assume um papel especial que envolve o cálculo e envio de chaves para os outros membros do grupo. O membro do grupo indicado para assumir esta responsabilidade é denominado de *sponsor*. Os protocolos definidos pelo TGDH assumem que cada membro tem a capacidade de determinar, de forma exata, o ponto de inserção de um novo membro na árvore (no caso de um join), bem como que membro é o *sponsor* atual do grupo. Quando da entrada de um novo membro, é determinado o ponto de inserção na árvore, que será a folha mais à direita em que a operação não aumenta a altura da árvore, ou a raiz caso a árvore esteja completamente balanceada. O *sponsor* será a folha mais à direita da sub-árvore cuja raiz é o ponto de inserção do novo membro.

O TGDH [03] é o único protocolo a possuir a figura de um controlador de grupo, (*sponsor)* possuindo um papel fundamental no recálculo da chave do grupo. A centralização cria um ponto único de falha no protocolo, pois uma falha no *sponsor* pode provocar o impedimento da entrada de um usuário no grupo. No TGDH são necessárias 2 mensagens e *n*+1 exponenciações a cada entrada de um novo membro no grupo.

O **NKey** possui uma estrutura descentralizada não havendo a figura do controlador do grupo. Dado que todas as operações de entrada e saída do grupo são tratadas de forma idêntica, também não há uma fase de inicialização. Para a entrada de um novo membro no grupo, o **NKey** realiza, no pior caso, *h-1* trocas de mensagens e *h* exponenciações, onde *h* representa a altura da árvore no momento da entrada do novo membro. Uma árvore com altura *h* possui um máximo de 2^{h-1} folhas. Como quando do aumento da altura da árvore a sub-arvore à esquerda já esta completa, teremos um máximo de $(2^{h-1}/2)$ entradas de novos membros no grupo, sem que seja necessário um novo aumento na altura da árvore. Desta forma teremos um máximo de $(2^{h-1}/2\ )h$ exponenciações, com h variando de 2 até o seu valor máximo, para a formação de um grupo com 2^{h-1} membros. Uma generalização semelhante pode ser feita em relação ao número de mensagens a serem trocadas.

Para um grupo com 1000 usuários teríamos:

GDH: Como não há o conhecimento de quais usuários formarão o grupo, a fase de inicialização é usada apenas para os dois primeiros usuários. Desta forma temos:

Exponenciações na inicialização: 5*n*-6 = 5*2-6 = 4
Exponenciações após inicialização: $\sum_{n=3}^{1000} 2n-1 = 999996$
Mensagens na inicialização: 2n-1 = 2*2-1 = 3
Mensagens após inicialização: 2*997 = 1994

TGDH:
Exponenciações: $\sum_{n=1}^{1000} n+1 = 499504$
Mensagens: 1*1000 = 2000

NKey: Para um conjunto de 1000 usuários a altura máxima da árvore será igual a 10, logo:

Exponenciações: $\sum_{h=2}^{10} (2^{h-1}/2\ )h = 4608$

Mensagens: $\sum_{h=2}^{10}(2^{h-1}/2)(h-1) = 4097$

Apesar de realizar um número maior de envio de mensagens, o **NKey** realiza um número bastante inferior de exponenciações, que são operações computacionalmente pesadas, quando comparado ao GDH e TGDH.

6. Conclusões e trabalhos futuros

Este trabalho apresentou um conjunto de protocolos que permitem o cálculo e gerência de chaves de grupos. Os protocolos propostos possuem como principais contribuições o fato de serem adequados a grupos dinâmicos, dado que as operações de entrada e saída do grupo são computacionalmente leves.

Visando dar continuidade a este trabalho pretendemos estender o simulador apresentado, de forma a integrá-lo ao simulador de redes NS [10], o que irá permitir a realização de testes e simulações que levem em consideração fatores como retardo e erros na transmissão de mensagens, falhas em links ou em usuários, entre outros. No âmbito do Laboratório NatalNet, está sendo desenvolvido um sistema de telemedicina que irá utilizar o **NKey** para prover segurança na transmissão de informações sensíveis como imagens de exames ou prontuários e diagnósticos de procedimentos realizados.

7. Referências Bibliográficas

[01] SOARES, LEMOS, COLCHER. *Redes de Computadores: das LANs, MANs e WANs às redes ATM*. 2° edição. Editora Campus,1995.

[02] M. STEINER, G. TSUDIK e M. WAIDNER. Diffie-Hellman Key Distribution Extended to Group Communication. In ACM Symposium on Computer and Communication Security, Março de 1996.

[03] KIM, YONGDAE. PERRIG, ADRIAN. TSUDIK, GENE. *Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups.* Technical Report 2, USC Technical Report 00-737, Agosto de 2000.

[04] COSQUER, F. J. N., e VERÍSSIMO, P. *Survey of Selected Groupware Applications and Supporting Platforms*. Lisboa-Portugal: INESC – Technical Report 21-94. 1994.

[05] ATENIESE, Giusepe. *Secure and Efficient Group Communication in Wide and Local Area Networks*. Tese de Doutorado (Theses Series DISI-TH-1999-XX). Departamento de Informática e e Ciência da Informação – Gêneva – Itália. 1999.

[06] A. MENEZES, P. van OORSCHOT e S. VANSTONE. *Handbook of Applied Cryptography.* 2001.

[07] DIFFIE, W & HELLMAN, M. *New Directions In Cryptography*. IEEE Transactions on Information Theory, IT-22(6):644-654, Novembro 1976.

[08] STALLINGS, Willian. *Cryptography and network security: Principles and practice*. Prentice Hall. 1998

[09] TANENBAUM, A.S. *Sistemas Operacionais Modernos*, Prentice Hall do Brasil. 1992

[10] *The Network Simulator - ns-2*. http://www.isi.edu/nsnam/ns/

[11] D. A. Agarwal, O. Chevassut, M. R. Thompson and G. Tsudik. *An Integrated Solution for Secure Group Communication in Wide-Area Networks*. In 6th IEEE Symposium on Computers and Communications. pp22-28. 2001

[12] P. McDaniel, Atul Prakash, and P. Honeyman. *ANTIGONE: A Flexible Framework For Secure Group Communication*. Proceedings do 8th USENIX Security Symposium, pp 99-114, Agosto de 1999.

[13] M. Steiner, G. Tsudik and M. Waidner. *CLIQUES: A New Approach to Group Key Agreement*. IEEE ICDCS'97, Maio de 1997

[14] C. Becker and U. Wille. *Communication Complexity of Group Key Distribution*. In 5th ACM Conference on Computer and Communications Security. Novembro de 1998.

[15] Peyravian, M., Matyas, S.M., and Zunic, N. *Decentralized group key management for secure multicast communications*. Computer Communications vol.22. no.13. pp 1183-1187. 1999.

[16] D. McGrew, A. T. Sherman. *Key Establishment in Large Dynamic Groups Using One Way Function Trees*. IEEE Transactions on Software Engineering. Maio de 1998.

[17] Refik Molva, Alain Pannetrat. *Scalable Multicast Security in Dynamic Groups*. Proceedings of the 6th ACM Conference on Computer and Communications Security, Novembro de 1999

[18] C.K. Wong, M. Gouda, and S.S. Lam. *Secure Group Communications Using Key Graphs*. In ACM SIGGCOM. ACM, Setembro de 1998.