Carlos Gustavo Araújo da Rocha

MKey: Um Protocolo Para o Gerenciamento de Chaves de Grupo

Natal

Março de 2003

Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemática Aplicada
Programa de Pós-graduação em Sistemas e Computação

MKey: Um Protocolo Para o Gerenciamento de Chaves de Grupo

Dissertação de mestrado apresentada ao Programa de Pós-graduação em Sistemas e Computação do Departamento de Informática e Matemática Aplicada do Centro de Ciências Exatas e da Terra da Universidade Federal do Rio Grande do Norte.

Carlos Gustavo Araújo da Rocha

Orientador:

Guido Lemos de Sousa Filho, Dr.Sc.

Co-orientador:

Benjamim René Callejas Bedregal, Dr.Sc.

Natal

Março de 2003

Dissertação de Mestrado sob o título *MKey: Um Protocolo Para o Gerenciamento de Chaves de Grupo*, defendida por Carlos Gustavo Araújo da Rocha e aprovada em 25 de março de 2003, em Natal RN, pela banca examinadora constituída pelos seguintes doutores:

Prof. Dr. Guido Lemos de Sousa Filho
Orientador

Prof. Dr. Benjamim René Callejas Bedregal
Co-orientador

Prof. Dr. Otto Carlos Muniz Bandeira Duarte
Universidade Federal do Rio de Janeiro

Dedicatória e Agradecimentos

A Deus e a todos aqueles que de alguma forma contribuíram para tornar este trabalho possível.

"Desgraça pouca é meio de vida"

Zé Duré

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. Esta dissertação apresenta uma extensão do algoritmo *Diffie-Hellman* que permite calcular chaves assimétricas para grupos. O texto inclui o modelo e prova matemática de validade da extensão. Uma vez calculadas, as chaves podem ser usadas para viabilizar a comunicação segura entre os membros do grupo. Para dar suporte ao gerenciamento das chaves foi desenvolvido um protocolo denominado Mkey, do inglês *Multicast Key*. A viabilidade dos conceitos, nos quais se basearam a concepção do protocolo, foi comprovada através da implementação e teste de suas funções básicas.

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 thesis presents a proposal based on the Diffie-Hellman algorithm [6]. This algorithm provides support to key exchange between a pair of users. Our proposal extends Diffie-Hellman algorithm to generate and to manage secure asymmetric group keys, providing a mathematical model and the proff that validates the extension made. We have also developed a protocol to provide support for group key management called MKey, from Multicast Key. In order to verify the viability of the proposed protocol, we have implemented and tested its basic functions.

Keywords: Security, Cryptography, Secure group communication.

Sumário

Lista de Figuras

Lista de Tabelas

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* [17] de modo que todas elas oferecem e requisitam os mesmos serviços para as outras entidades que formam o sistema. Esta característica é o principal diferencial em relação a aplicações centralizadas, baseadas em um modelo *client-server* [17] onde existe uma clara diferenciação entre as entidades que ofertam serviços (servidores), e aquelas que os requisitam (clientes). Os modelos *peer-to-peer* e *client-server* são exemplificados na figura 1.

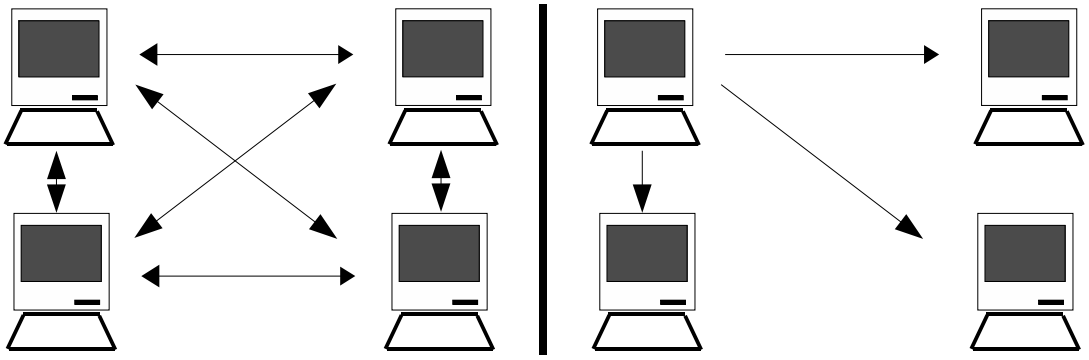

Figura 1: Modelos *peer-to-peer* (esquerda) e *client-server* (direita)

Como descrito em [7], apesar de grupos colaborativos possuírem uma tendência de serem relativamente pequenos, seus membros normalmente estarão distribuídos na Internet e devem ter a capacidade de lidar com problemas como falhas na rede, congestionamento e ataques externos. Estes problemas podem subdividir o grupo em vários sub-grupos disjuntos os quais devem persistir e continuar operando como um grupo colaborativo independente.

Um sistema de comunicação de grupo seguro deve oferecer serviços de envio e recebimento de mensagens baseados em mecanismos que garantam a integridade e autenticidade dessas mensagens, de forma que se possa, por exemplo, identificar uma mensagem forjada por um intruso. No entanto, as atuais estratégias disponíveis na literatura para a comunicação segura de grupos pecam pelo excessivo poder computacional necessário à sua execução, ou devido a um ou mais membros do grupo possuírem um papel especial no funcionamento do sistema, o que fere a natureza colaborativa do grupo e cria um ponto único de falha.

1.1 Objetivos

Os objetivos dessa dissertação podem ser resumidos em:

Desenvolver um algoritmo para cálculo de chaves de grupo: O enfoque desta etapa será o cálculo e gerenciamento das chaves geradas, que poderão ser utilizadas pelos membros do grupo para implementar o serviço de segurança que eles desejarem.

Desenvolver e validar um protocolo que dará suporte ao algoritmo: Este protocolo irá implementar, entre outras, as operações que lidam com alterações na estrutura do grupo, como as de entrada e saída de membros do grupo.

Implementar os algoritmos e protocolo propostos: Comprovar a viabilidade dos algoritmos e protocolo propostos através da implementação e testes de suas funções básicas.

1.2 Organização do texto

Esta dissertação de mestrado está organizada da seguinte forma: O capítulo 2 apresenta uma breve fundamentação das tecnologias envolvidas neste trabalho, a saber: A

Segurança de Redes, a Criptografia, o Gerenciamento de Chaves, a Comunicação de Grupo, e o Gerenciamento de Chaves de Grupo. O capítulo 3 discorre sobre alguns dos principais trabalhos correlatos encontrados na literatura. O capitulo 4 apresenta o algoritmo, a prova matemática de sua corretude e o protocolo concebido para dar suporte ao gerenciamento de chaves de grupo. A seguir, no Capítulo 5, são apresentados os principais aspectos relativos à implementação do protocolo proposto, acompanhada de resultados de testes de sua execucao. No capítulo 6, são apresentadas as contribuições decorrentes deste trabalho, e uma comparação com os trabalhos correlatos encontrados nas pesquisas bibliográficas realizadas. Ainda neste Capítulo, são apresentadas as conclusões obtidas e perspectivas de trabalhos futuros.

2 Fundamentação teórica

Neste capítulo são apresentados os conceitos das principais tecnologias envolvidas no desenvolvimento desta dissertação. Inicialmente é abordado o tópico de Segurança de Redes, em seguida a Criptografia e por fim a Comunicação em Grupo.

2.1 Segurança de Redes

O documento ISO 7498-2 [15] é uma extensão do documento ISO 7498-1 [11] onde a ISO propõe uma arquitetura para segurança de redes. Esta arquitetura define os serviços de segurança OSI bem como as camadas do RM-OSI onde devem ser oferecidos estes serviços e implementados os mecanismos de segurança. A arquitetura de segurança OSI trata exclusivamente dos aspectos de segurança relacionados a comunicação de sistemas finais, não abrangendo medidas de segurança que devem ser adotadas nos sistemas intermediários. Os serviços propostos neste documento são:

Autenticação: O Serviço de Autenticação está relacionado com a identificação das entidades que são parceiras em uma comunicação, bem como dos dados que elas trocam. Ele é subdividido em duas classes: Autenticação de entidades e autenticação da origem de uma unidade dados.

Autenticação de Entidades: Dentro de um ambiente de comunicação baseado no modelo OSI, o serviço de Autenticação de Parceiro, quando fornecido pela camada *N*, oferece a uma entidade da camada *N+1* uma comprovação de que sua parceira em uma comunicação, outra entidade da camada *N+1*, é realmente quem diz ser.

Autenticação da Origem de uma Unidade de Dados: O serviço de autenticação da Origem de uma Unidade de Dados, quando oferecido pela camada

N, oferece a uma entidade da camada *N+1* uma comprovação de que a origem de uma unidade de dados, outra entidade da camada *N+1*, é realmente quem afirma ser.

Controle de Acesso: O serviço de Controle de Acesso fornece proteção contra o uso não autorizado dos recursos cujo acesso se dê via sistema de comunicação de dados OSI. O requisito necessário para o fornecimento do serviço é o acesso ser viabilizado por protocolos OSI.

Confidencialidade de Dados: O serviço de Confidencialidade de Dados protege dados trocados no ambiente OSI contra revelação não autorizada da informação neles contida para todos, exceto para uma entidade autorizada. Este serviço pode ser oferecido em diferentes níveis como uma conexão, uma unidade de dados ou campos de uma unidade de dados.

Integridade de Dados: O Serviço de Integridade de Dados protege dados trocados num ambiente OSI contra ataques que impliquem na modificação, remoção ou inserção não autorizada de unidades de dados. Este serviço também pode ser fornecido em diferentes níveis: Em uma conexão ou unidade de dados, podendo ou não recuperar a informação original, e em campos selecionados nas unidades de dados.

Impedimento de Rejeição: O Serviço de Impedimento de Rejeição atua através da prova da identidade das entidades que solicitam a execução de serviços, impedindo a rejeição de serviços, ou através da prova que uma entidade de destino recebeu corretamente uma solicitação para realização de um determinado serviço.

2.2 Criptografia

Um mecanismo fundamental para o fornecimento dos serviços descritos na seção anterior é a criptografia. A criptografia é uma técnica bastante antiga, a 4000 anos já era usada pelos egípcios. Entretanto, sua importância e utilização cresceram de forma considerável nas últimas décadas junto com a proliferação de computadores e a necessidade de proteção da informação digital.

A criptografia é definida como o estudo do conjunto de técnicas matemáticas relacionadas a aspectos da segurança da informação como confidencialidade, integridade e

autenticação [10]. A criptografia define duas operações básicas: Encriptação e decriptação.

De maneira genérica, na operação de encriptação, uma mensagem original M é transformada em uma mensagem criptografada M', pela aplicação de um algoritmo A e de uma ou mais chaves K independentes da mensagem. O algoritmo deve produzir uma saída única para cada chave passível de ser utilizada.

Uma mensagem criptografada M' pode ser transformada novamente na mensagem original M pela aplicação do mesmo algoritmo A e da chave correspondente K, operação conhecida como decriptação. Estas operações são mostradas na figura 2.

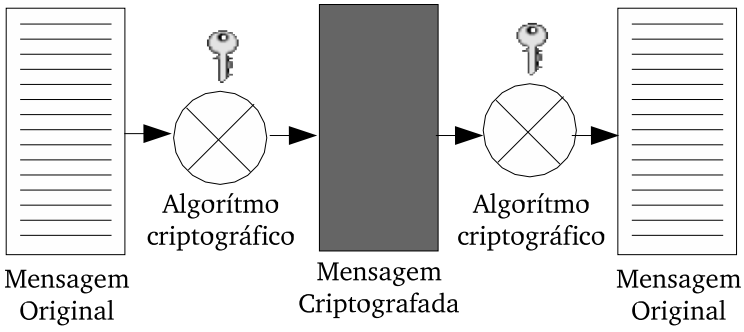

Figura 2: Processo genérico de criptografia

O algoritmo usado para a geração da mensagem criptografada pode ser uma informação pública, bem como a própria mensagem criptografada M'. Neste caso a segurança de um algoritmo de criptografia depende do segredo do conjunto de chaves e não do algoritmo utilizado. Deste modo deve ser impraticável para um atacante obter M pelo conhecimento de M' e do algoritmo usado para a sua geração.

Os algoritmos de criptografia podem ser classificados, de forma genérica, sob três aspectos [18]:

1. **Número de chaves:** Caso as entidades de origem e de destino de uma mensagem utilizem a mesma chave para encriptar e decriptar, o algoritmo é dito simétrico ou convencional. Se as entidades de origem e de destino usam chaves diferentes, o algoritmo é dito assimétrico ou de chave pública.

2. **Tipo das operações usadas para transformar a mensagem original na**

mensagem criptografada: De uma forma geral os algoritmos de criptografia convencionais são baseados em dois princípios: substituição ou transposição. Na substituição cada elemento do texto (uma letra, por exemplo), é mapeado em um outro elemento. Já na transposição, os elementos são rearranjados dentro do texto. Um requerimento comum é que estas operações devem ser reversíveis. Na prática um único algoritmo pode realizar várias substituições e transposições sobre a mensagem original. Os algoritmos de criptografia assimétrica representam uma excessão em relação a este aspecto, sendo baseados em técnicas de manipulação matemática que serão detalhadas posteriormente.

3. **Modo como a mensagem original é processada:** Caso a mensagem original seja processada utilizando como unidade de entrada e saída blocos de dados de tamanho fixo, o algoritmo é dito um *block cipher*. Caso a mensagem original seja processada de forma contínua, como uma única unidade de dados, não importando o seu tamanho, o algoritmo é dito um *stream cipher*.

Uma visão geral dos temas abordados pela criptografia, bem como da relação entre eles é dada pela figura 3. Os algoritmos de criptografia Simétricos e Assimétricos serão detalhados nas seções 2.2.1 e 2.2.2 respectivamente.

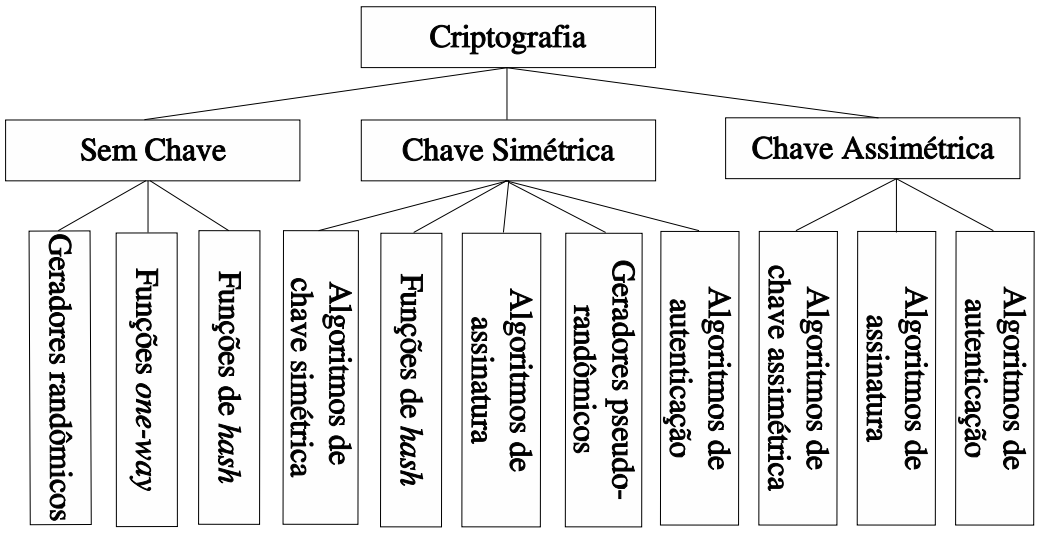

Figura 3: Visão geral dos temas abordados pela criptografia

2.2.1 Criptografia Simétrica

Os Algoritmos de criptografia simétrica, também conhecida como criptografia convencional ou *single-key encription*, englobam os primeiros algoritmos de criptografia desenvolvidos, possuindo um uso substancialmente maior quando comparados com a criptografia de chave pública ou assimétrica.

A substituição e a transposição formam a base das operações realizadas por qualquer algoritmo de criptografia simétrico. Nas técnicas de substituição cada um dos símbolos possíveis na mensagem a ser encriptada (um caracter ASCII, por exemplo) é substituído por um outro símbolo, com base em um padrão único válido para todos os símbolos presentes no alfabeto da mensagem original. Um exemplo clássico de técnica de substituição é a permutação das 26 letras do alfabeto de forma aleatória. Neste caso a chave seria representada pelo alfabeto com as 26 letras permutadas, tendo 26! possibilidades de chaves possíveis.

As técnicas de transposição se caracterizam pela realização de uma série de permutações nos símbolos da mensagem original. Um exemplo clássico de um algoritmo de transposição é o *rail fence* no qual o texto é escrito como uma seqüência de diagonais e lido como uma seqüência de colunas. A mensagem "segurança de redes de computadores"encriptada com o algoritmo *rail fence* com profundidade 2 seria representada por:

```
s g r n a d   e e   e c m u a o e
 e u a ç   e r d s d   o p t d r s
```

A mensagem criptografada final seria: *sgrnad ee ecmuaoeeuaç erdsd optdrs*.

Uma técnica mais apurada é a de escrever a mensagem em retângulo linha a linha e recuperá-lo coluna a coluna, em uma seqüência aleatória que será a chave do algoritmo. A mesma mensagem do exemplo anterior seria criptografada como:

```
KEY:4 1 5 6 2 3
    s e g u r a
    n ç a   d e
    r e d e s
    d e   c o m
    p u t a d o
```

```
r e s
```

A mensagem criptografada final seria: *eçeeuerdsod ae mo snrdprgad tsu eca.* A técnica de transposição é normalmente aplicada várias vezes sobre o texto, tornando o algoritmo bem mais seguro.

Técnicas como a substituição e a transposição podem ser usadas em conjunto e de forma repetida tornando os algoritmos de criptografia mais complexos. O principal algoritmo fruto da criptografia simétrica é o DES [1] (Data Encryption Standard) que foi adotado em 1977 pelo NIST [13] como o algoritmo de criptografia padrão para a troca de dados seguros nos Estados Unidos da América. Apesar de ser um algoritmo relativamente antigo, o DES, ou variações dele, ainda permanecem como o algoritmo de criptografia mais usado nos dias atuais.

No algoritmo DES uma mensagem é criptografada em blocos de 64 bits, usando uma chave de 56 bits. Na prática a cada sete bits da chave é adicionado um bit de paridade, o que perfaz uma chave de 64 bits. Para cada bloco de entrada o algoritmo gera uma saída, também de 64 bits. Caso o algoritmo receba como entrada uma mensagem criptografada e a chave que foi usada para criptografa-la, a saída será a mensagem original.

O DES é dividido em várias etapas sendo iniciado por uma transposição inicial nos 64 bits da mensagem, seguido por 16 ciclos de cifragem e por uma transposição final (inversa à transposição inicial). Os 16 ciclos intermediários são computacionalmente idênticos, envolvendo operações de permutação e substituição, tendo como entrada a mensagem gerada na saída do ciclo anterior e uma sub-chave de 48 bits, gerada a partir da chave principal.

2.2.2 Criptografia Assimétrica

Os algoritmos de criptografia assimétrica também conhecidos como algoritmos de chave pública tem como principal característica a utilização de um par de chaves único por cada usuário. Uma dessas chaves, conhecida como chave pública, é distribuída livremente pelo usuário para todos aqueles que a desejarem. A outra chave, conhecida como chave privada, é uma informação de conhecimento apenas do seu proprietário. A criptografia de chave pública foi proposta em 1976 por *Diffie, W. e Hellman, M.* no algoritmo *Diffie-Hellman* [6]. O algoritmo proposto na referência supracitada tem

como objetivo possibilitar 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. Utilizando a notação da tabela 1, o algoritmo *Diffie-Hellman* pode ser definido 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

Tabela 1: Notação para o *Diffie-Hellman*

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 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 de forma aleatória um inteiro $X_1 < p$, e calcula $Y_1 = r^{X_1} \bmod p$. Do mesmo modo o usuário U_2, de forma independente, escolhe de forma aleatória um inteiro $X_2 < p$, e calcula $Y_2 = r^{X_2} \bmod p$. Os 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 $K = Y_2^{X_1} \bmod p$. De forma semelhante U_2 calcula a chave $K = Y_1^{X_2} \bmod p$.

Neste ponto os dois usuários terão calculado a chave secreta que será usada em futuras trocas de mensagens. O valor resultante dos dois cálculos é idêntico, como demonstrado a seguir:

$$
\begin{aligned}
K &= Y_2^{X_1} \bmod p \\
&= (r^{X_2} \bmod p)^{X_1} \bmod p \\
&= (r^{X_2})^{X_1} \bmod p \\
&= r^{X_2 X_1} \bmod p \\
&= (r^{X_1})^{X_2} \bmod p \\
&= (r^{X_1} \bmod p)^{X_2} \bmod p \\
&= Y_1^{X_2} \bmod p
\end{aligned} \tag{2.1}
$$

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 $Y_1 = r^{X_1} \bmod p$, ou seja, o atacante teria que calcular um logaritmo discreto de Y_1 para a base r módulo p. 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*.

Atualmente, o algoritmo mais usado da criptografia assimétrica é o *Rivest Shamir Adleman*, ou simplesmente *RSA* [16]. Ele pode ser utilizado para prover tanto

criptografia como assinatura digital, e sua segurança é baseada na intratabilidade do problema de fatoração de números inteiros. Para a definição do algoritmo *RSA* utilizaremos a seguinte notação:

p,q	Números primos
M	Mensagem original
C	Mensagem criptografada

Tabela 2: Notação para o *RSA*

No *RSA* cada usuário que deseja utilizar o algoritmo é responsável por calcular, de forma independente, o seu par de chaves pública / privada. Este processo é iniciado pela geração de dois números primos (p e q). A seguir, é realizada a seguinte seqüencia de operações:

1. Calcula $n = p \times q$
2. Calcula $\Phi(n) = (p-1) \times (q-1)$
3. Calcula e, de modo que $MDC(\Phi(n), e) = 1; 1 < e < \Phi(n)$
4. Calcula $d = e^{-1} \bmod \Phi(n)$

A chave privada do usuário consiste na tupla $\{d, n\}$, a chave pública consiste na tupla $\{e, n\}$. De acordo com a matemática modular o cálculo de d a partir de $\{e, n\}$ é inviável. Uma mensagem é criptografada pelo cálculo de $C = M^e \bmod n$ e decriptografada pelo cálculo de $M = C^d \bmod n$.

No *RSA* a mensagem original é criptografada em blocos, cada bloco deve obrigatoriamente ter seu valor binário inferior ao número n, ou seja, o comprimento do bloco em bits deve ser menor ou igual a $\log(n)$. Para os usuários U_1 e U_2 trocarem uma mensagem criptografada, é necessária a seguinte seqüência de eventos:

1. U_1 requisita a U_2 sua chave pública $\{e, n\}$;
2. U_1 calcula $C = M^e \bmod n$ (mensagem criptografada), usando para tanto os valores de $\{e, n\}$ recebidos de U_2;
3. U_2 recebe C e calcula $M = C^d \bmod n$.

A segurança do *RSA* é diretamente ligada ao tamanho escolhido para as chaves, ou seja, ao comprimento de d e e em bits. Quanto maior for o tamanho escolhido, maior será a resistência do algoritmo aos ataques conhecidos, como os de força bruta. No entanto, devido a natureza dos cálculos realizados pelo algoritmo o tamanho da chave é um fator computacionalmente limitante.

2.2.3 Gerenciamento de Chaves

O Gerenciamento de chaves compreende os procedimentos necessários para o estabelecimento e manutenção de uma chave entre duas partes autorizadas, ou seja, dois usuários. De forma mais específica o Gerenciamento de Chaves trata a geração, distribuição, armazenamento, e remoção de chaves por um usuário.

A distribuição de chaves, na sua maneira mais simples, ocorre pela livre distribuição por parte dos usuários de sua chave pública como mostrado na figura 4. O principal problema deste esquema está no fato que um atacante A_1 pode fingir ser um usuário U_1 pelo envio de uma chave pública forjada para um ou mais usuários, figura 4. O atacante A_1 terá acesso a todas as mensagens enviadas para U_1, até que o ataque seja descoberto.

Um esquema mais aprimorado para a distribuição de chaves faz uso de um "intermediário confiável"que atua como um servidor de diretório para a distribuição das chaves. Neste esquema os usuários devem enviar as suas chaves públicas para uma entidade confiável que mantém um diretório das chaves recebidas. As chaves recebidas pelo intermediário confiável são mantidas em tuplas *{ID_USUÁRIO,chave}* e são publicadas livremente, por meios eletrônicos ou convencionais como uma página web e um catálogo impresso respectivamente. No momento do cadastramento da chave pública de um usuário, o intermediário confiável deve utilizar algum mecanismo para validar a origem da chave. A figura 5 demonstra o funcionamento deste esquema.

No esquema explicado no parágrafo anterior, uma falha no funcionamento no intermediário confiável inviabiliza o seu funcionamento como um todo, sendo o intermediário confiável um ponto único de falha do sistema. Um comprometimento, por parte de um atacante, do intermediário confiável revoga todas as chaves contidas no seu diretório.

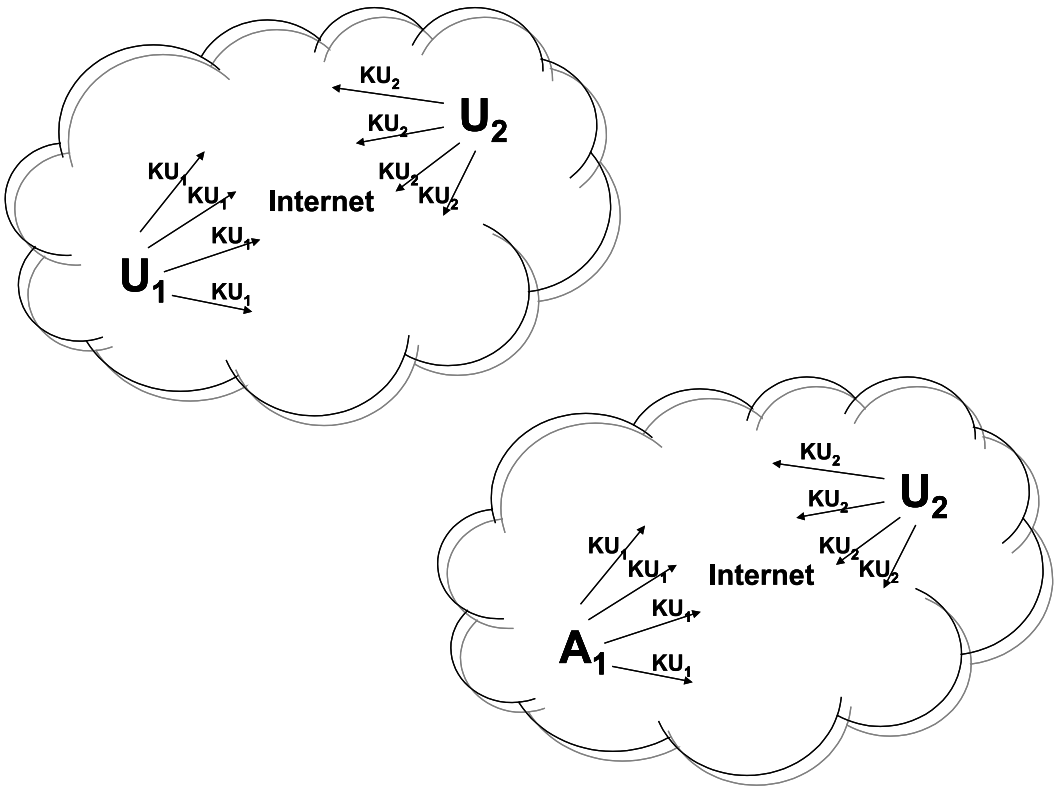

Figura 4: Distribuição de chaves públicas

2.3 Comunicação de grupo

Na visão de Cosquer & Veríssimo [5] 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 : É representado por um conjunto $\mathcal{G}$, composto de $\mathcal{X}$ elementos $\{E_1, ..., E_X\}$. $\mathcal{G}(\mathcal{X})$ é uma forma abstrata de representar o conjunto de todos os elementos de $\mathcal{G}$. Assim, pode-se endereçar todos os elementos de $\mathcal{G}$ por $\mathcal{G}(\mathcal{X})$ sendo possível enviar uma mensagem para todos os elementos sem nomear explicitamente cada um deles.

Membro : Um membro de um grupo é representado por cada elemento de $\mathcal{G}$ que

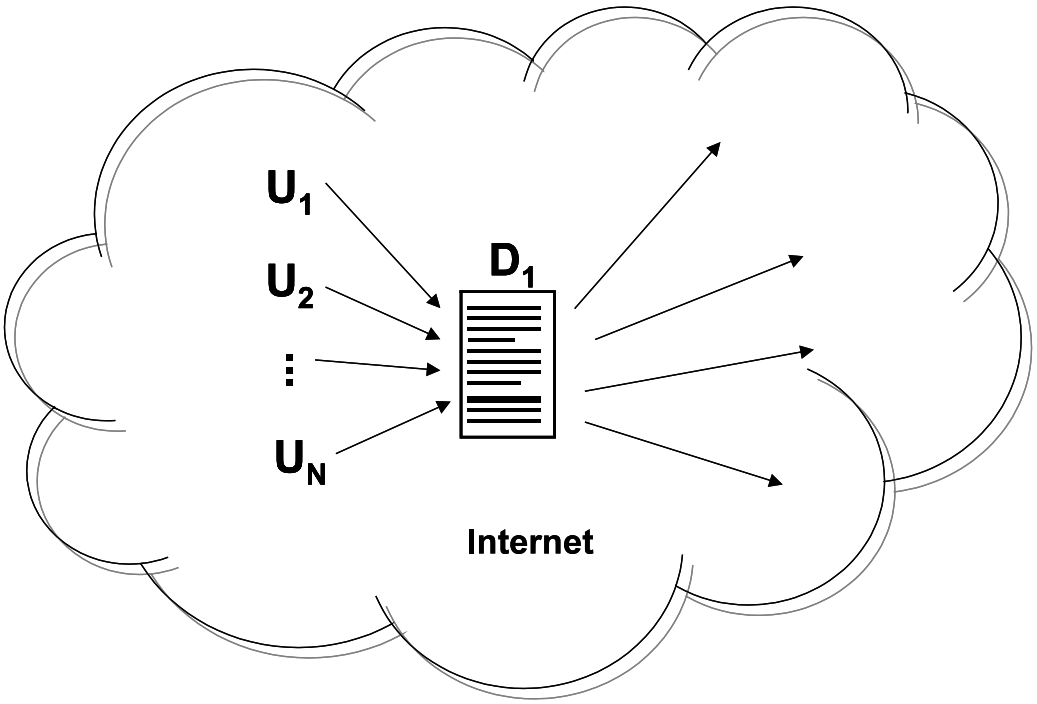

Figura 5: Diretório de chaves públicas

interage com outros elementos através de troca de mensagens.

Mensagem : Compreende o envio de alguma informação, a partir de um elemento E_n de $\mathcal{G}$ para $\mathcal{G}(\mathcal{X})$. Desta forma, temos:

$$E_n \overrightarrow{m} \mathcal{G}(\mathcal{X})$$

Processos de controle : São responsáveis pelas atividades de gerência dessa sistemática. Segundo Tanenbaum [21], a implementação dessa característica define políticas para os serviços e para a comunicação 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 : Nos grupos fechados somente seus membros podem enviar mensagens, ou seja, processos que não pertençam ao grupo não podem enviar mensagens para o mesmo, 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 a criação e eliminação dinâmica de grupos são permitidas. 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 deixá-lo. Nos grupos dinâmicos, pode-se ter métodos de controle de acesso para a entrada de novos membros no grupo.

2.3.1 Gerenciamento de Chaves de Grupo

Um ambiente de Comunicação de Grupo que ofereça serviços de segurança deve estender o modelo de gerenciamento de chaves, apresentado na seção 2.2.3, de forma que ele passe a compreender os procedimentos necessários para o estabelecimento e manutenção das chaves entre as N, ao invés de 2 partes autorizadas. O gerenciamento de chaves de grupo irá tratar a geração, distribuição, armazenamento e revogação das chaves do grupo.

Estas funcionalidades podem ser sumarizadas em:

Métodos para recálculo : Em grupos dinâmicos, onde são permitidas as operações de entrada e saída de membros do mesmo, deve-se definir protocolos que serão instânciados quando da ocorrência de um desses dois eventos, resultando na geração de um novo conjunto de chaves para o grupo.

Perfect Secret Forward: O *Perfect Secret Forward* é uma característica pela qual novos membros do grupo não tem acesso a nenhuma mensagem anterior a sua entrada no grupo, bem como membros que deixem o grupo não terão acesso a

nenhuma mensagem posterior a sua saída do grupo. Esta é uma das principais características no que tange a segurança de um sistema de gerenciamento de chaves de grupo.

Dentre os fatores que influenciam a eficiência de um sistema de gerenciamento de chaves de grupo podemos citar:

Número de cálculos : Representa o número e complexidade das operações de recálculo de chaves que deverão ser realizadas por cada membro do grupo quando da alteração da sua estrutura.

Número de mensagens : Representa o número de mensagens *unicast* e *multicast* que deverão ser trocadas pelos membros do grupo quando de uma alteração na sua estrutura.

Tamanho de mensagens : Representa o comprimento em bits das mensagens a serem trocadas quando da ocorrência de alguma alteração na estrutura do grupo.

Tamanho do chaveiro : Representa o número máximo de chaves a serem armazenadas por cada membro do grupo durante a execução do protocolo.

3 Trabalhos Correlatos

Existem diversos trabalhos disponíveis na literatura que abordam a área de comunicação segura de grupos. O trabalho [2] descreve um protocolo para comunicação segura de grupos composto de um sistema de comunicação de grupo confiável; mecanismos de autorização e controle de acesso; e um protocolo para o cálculo de chaves. Em [14] é apresentado um método para a geração e distribuição de chaves simétricas entre os membros de um grupo *multicast*. O trabalho [12] propõe um framework para *multicast* seguro baseado na realização de operações nos nós intermediários da comunicação. Os trabalhos [8, 9, 22] apresentam sistemas de comunicação segura de grupo com ênfase na minimização de complexidade. Em [4, 3] encontramos respectivamente uma análise da complexidade das operações necessárias a um sistema de comunicação segura de grupos e uma avaliação da performance de alguns dos principais trabalhos desenvolvidos na área.

Como critério para escolha dos trabalho que tomamos como base de comparação consideramos aqueles que utilizam abordagens semelhantes à nossa, além de serem considerados o estado da arte da comunicação segura de grupos, a saber: GDH [19]; TGDH [7]; e CLIQUES [20].

3.1 GDH

O trabalho abordado nesta seção, *Diffie-Hellman Key Distribution Extended to Group Communication* [19], apresenta uma série de protocolos considerados extensões naturais do *Diffie-Hellman* para o caso de n, ao invés de dois participantes. De forma semelhante ao *Diffie-Hellman* tradicional, todos os membros do grupo compartilham um número primo p e uma base exponencial a.

3.1.1 Protocolo GDH.1

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. Cada membro M_i do grupo recebe a^{N_1}, $a^{N_1 N_2}$,..., $a^{N_1 \cdots N_{i-1}}$, onde a é uma base exponencial gerada na inicialização do algoritmo e N_i representa a chave privada do membro i, e calcula $a^{N_1 \cdots N_i}$ pela exponenciação de $a^{N_1 \cdots N_{i-1}}$ elevado à potência de N_i (sua chave privada), e adiciona $a^{N_1 \cdots N_i}$ a mensagem que será enviada para M_{i+1}. O passo final do estágio de *upflow* ocorre quando o membro M_n calcula $a^{N_1 \cdots N_n}$ conhecida como a chave do grupo, K_n. O estágio de *downflow* é iniciado imediatamente após o término do *upflow* e se caracteriza pela realização de i exponenciacões por cada membro M_i do grupo, uma para calcular K_n e $i-1$ para prover os valores necessários para os membros seguintes.

3.1.2 Protocolo GDH.2

O protocolo *GDH.2* é uma evolução do *GDH.1* que reduz o número de ciclos e de mensagens a serem trocadas para o cálculo da chave do grupo. O estágio de *upflow* é modificado de forma que cada membro M_i deve calcular i valores intermediários, cada um com $i-1$ expoentes, e um *valor cardinal* contendo i expoentes. Quando o estágio de *upflow* chega ao membro M_n o valor cardinal será $a^{N_1 \cdots N_{n-1}}$ de forma que Mn é o primeiro membro do grupo a calcular a chave do grupo K_n. No estágio seguinte M_n realiza um broadcast do seu conjunto de valores intermediários para todos os membros do grupo.

3.1.3 Protocolo GDH.3

Os protocolos *GDH.1* e *GDH.2* requerem que cada membro do grupo realize $i+1$ exponenciacões, o que pode ser um fator limitante para grupos grandes. O protocolo *GDH.3* visa solucionar este problema reduzindo o número de exponenciacões realizadas por cada membro do grupo. O *GDH.3* consiste de quatro estágios sendo o primeiro semelhante ao estágio de *upflow* do *GDH.1*. No segundo estágio M_{n-1} realiza um broadcast do valor $a^{N_1 \cdots N_{n-1}}$ para todos os outros membros do grupo. Como terceiro estágio cada M_i com $i \neq n$ gera um expoente e o envia para M_n. No estágio final, M_n eleva cada valor recebido no estágio anterior à potência de N_n e envia os $n-i$ valores resultantes para os membros do grupo. Ao final da quarta fase cada membro do grupo

M_i possui um valor da forma: $\alpha^{\Pi\{N_k|K\in[1,n]\wedge k\neq i\}}$ podendo calcular então a chave do grupo.

3.1.4 Alterações na estrutura do grupo

Os protocolos GDH assumem que o conjunto exato de membros que irá formar o grupo é determinado antes do início da execução do protocolo. No entanto protocolos para entrada e saída de um membro do grupo também são definidos.

O principal requisito a ser atendido quando da entrada de um novo membro no grupo é o do segredo das chaves anteriores em relação ao novo membro do grupo. Nos protocolos *GDH.2* e *GDH.3* a adição de um usuário, assumindo que M_n salva o conteúdo da mensagem de *upflow* (estágio 1, ciclo $n-1$) ocorre da seguinte maneira:

1. M_n gera um novo expoente N'_n, calcula uma nova mensagem de *upflow* e a envia para o novo membro, M_{n+1}.
2. M_{n+1} gera seu expoente e calcula a nova chave $K_{n+1} = a^{N_1 \ldots N_n N_{n+1}}$.
3. M_{n+1} calcula as n novas sub-chaves na forma $\{\alpha^{\Pi\{N_k|K\in[1,i]\wedge k\neq j\}}|j \in [1,n]\}$ e as envia para os outros usuários do grupo.

Quando da remoção de um membro do grupo, o principal requisito a ser atendido é o da impossibilidade dos membros excluídos calcularem as futuras chaves do grupo. Para os protocolos *GDH.2* e *GDH.3* o protocolo de remoção de um membro M_p, com $p \neq n$ ocorre com a geração de um novo expoente N'_n por M_n e o cálculo de um novo conjunto de $n-2$ sub-chaves $\{\alpha^{\Pi\{N_k|K\in[1,i]\wedge k\neq j\}}|j \in [1,n-1] \wedge k \neq p\}$ que são enviadas para os membros do grupo. No caso da exclusão do membro M_n, M_{n-1} assume o papel descrito no parágrafo anterior.

3.2 TGDH (Tree Based Group Diffie-Hellman)

O gerenciamento de chaves apresenta-se como o principal obstáculo no desenvolvimento de um sistema de comunicação segura de grupo. Apesar de o gerenciamento de chaves centralizado parecer inicialmente bastante atrativo, por ser, ao nível de implementação, inerentemente menos complexo, ele é inadequado para grupos colaborativos.

Em primeiro lugar, a centralização no cálculo das chaves vai de encontro a natureza colaborativa do grupo, que se baseia em um modelo *peer-to-peer*. Segundo, um modelo centralizado insere um ponto único de falha e um alvo atrativo para ataques externos. Além disso, num modelo centralizado é necessário estabelecer um canal de comunicação seguro com cada um dos outros membros do grupo para distribuir as chaves, o que torna o método muito pesado computacionalmente, com o crescimento do número de membros no grupo.

O TGDH (Tree Based Group Diffie-Hellman) [7] baseia-se numa abordagem que integra duas importantes tendências no gerenciamento de chaves de grupo: O uso de árvores binárias, para calcular e atualizar eficientemente as chaves do grupo; e o algoritmo Diffie-Hellman [6] para o cálculo de chaves comprovadamente seguras.

3.2.1 Método de Funcionamento

A figura 6 mostra uma árvore característica do *TGDH*. A raiz da árvore fica no nível 0 e as folhas, representando os membros do grupo, no nível n. Devido a árvore ser binária cada nó ou é uma folha, ou pai de outros dois nós. Cada nó é nomeado pela tupla $< l, v >$ onde l representa o nível do nó na árvore e $0 \leq v \geq 2^l - 1$. Cada nó possui duas chaves: $K_{<l,v>}$ e $BK_{<l,v>}$ que é calculada de forma análoga ao algoritmo *Diffie-Hellman*, ou seja, $BK_{<l,v>} = \alpha^{K_{<l,v>}} \bmod p$, onde p é um número primo e α é uma base exponencial, como requerido pelo algoritmo *Diffie-Helmman*.

Uma folha $K_{<l,v>}$ que abriga o membro M_i conhece $K_{<l,v>}$, além de cada chave no caminho de $< l, v >$ para $< 0, 0 >$, bem como cada chave $BK_{<l,v>}$ presente na árvore. Por exemplo o membro M_2 conhece $K_{<3,1>}$, $K_{<2,0>}$, $K_{<1,0>}$, $K_{<0,0>}$ e $BK_{<0,0>}$, $BK_{<1,0>}$, ..., $BK_{<3,7>}$.

Cada chave $K_{<l,v>}$ é calculada da seguinte maneira:

$$K_{<l,v>} = (BK_{<l+1,2v+1>})^{K_{<l+1,2v>}} \bmod p = (BK_{<l+1,2v>})^{K_{<l+1,2v+1>}} \bmod p$$

Em outras palavras, para se calcular uma chave em $< l, v >$ é suficiente que se conheça a chave pública de um dos filhos de $< l, v >$ e a chave secreta do outro filho de $< l, v >$. $K_{<0,0>}$ é um segredo compartilhado por todos os membros do grupo. O valor $K_{<0,0>}$ nunca é usado diretamente para promover encriptação, autenticidade, ou

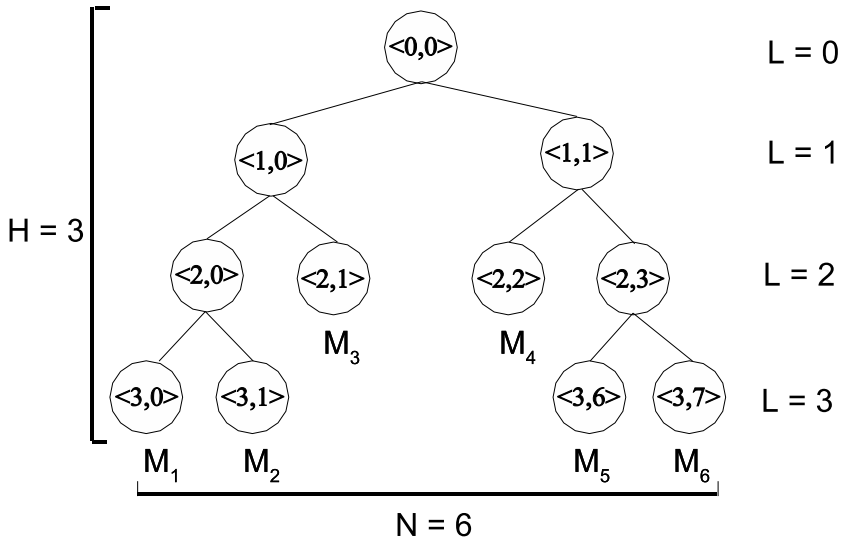

Figura 6: Árvore *TGDH*

integridade, sendo na verdade usado como valor de entrada para, por exemplo, uma função de Hash [18] $K_{group} = H(K_{<0,0>})$ que terá como resultado a chave a ser usada para estes fins.

3.2.2 Protocolos do TGDH

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 pode assumir um papel especial que envolve o cálculo das chaves $BK_{<l,v>}$ e seu envio 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.

Assumindo que o grupo possui n membros: $\{M_1, ..., M_n\}$, um novo membro M_{n+1} que deseja entrar no grupo inicia o protocolo enviando uma mensagem *Join* que contém sua chave $BK_{<l,v>}$. Quando os membros do grupo recebem esta mensagem eles determinam o ponto de inserção do novo membro na árvore, que será a folha mais a

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 a direita da subárvore cuja raiz é o ponto de inserção do novo membro. Como passo seguinte o *sponsor* cria dois novos nós: uma folha que representa o novo membro e um nó intermediário, que será promovido a pai do nó de inserção e do novo membro. O *sponsor* calcula então a nova chave do grupo e envia a nova árvore que possui todas as chaves $BK_{<l,v>}$ para os membros do grupo, que de posse das novas chaves, também calculam a nova chave do grupo. Uma operação de *join* é mostrada na figura 7.

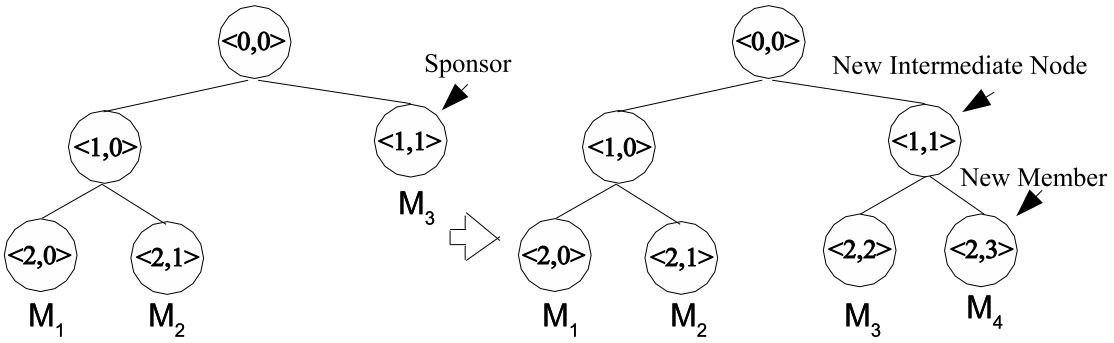

Figura 7: Operação Join *TGDH*

Na saída de um membro do grupo, operação *Leave*, o sponsor será a folha mais a direita da subárvore cuja raiz e o nó que será excluído, tem o mesmo pai. Para a saída de um membro M_d cada membro do grupo atualiza a árvore removendo a folha M_d e repondo o pai de M_d por seu outro filho. O *sponsor* calcula então as novas chaves no seu caminho até a raiz e envia o novo conjunto de chaves $BK_{<l,v>}$ para os outros membros do grupo, que de posse desses valores recalculam a chave do grupo. Uma operação Leave do membro M_3 é mostrada na figura 8.

3.3 CLIQUES

Este trabalho apresenta uma suite de protocolos chamada *CLIQUES* [20] desenvolvida para dar suporte unicamente a comunicação segura de grupos dinâmicos, deixando de lado outros serviços de segurança como integridade, autenticação e controle

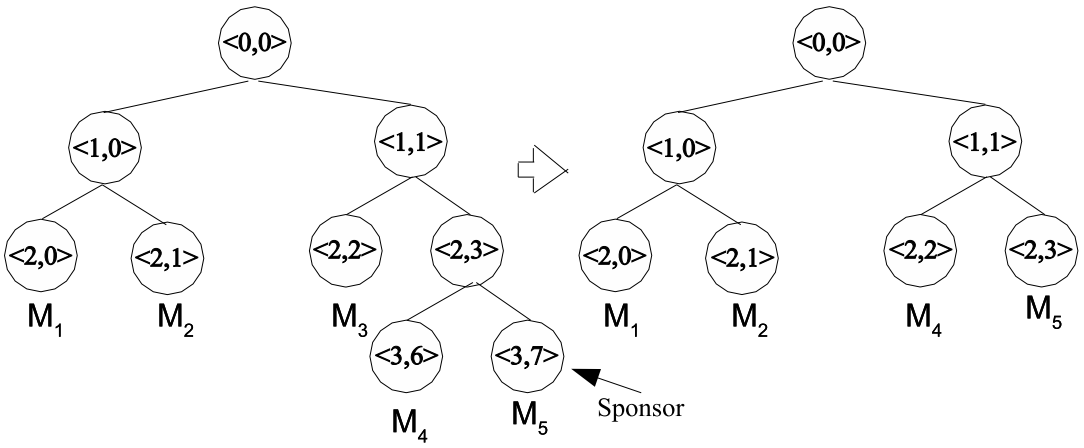

Figura 8: Operação leave *TGDH*

de acesso. O *CLIQUES* possui uma distinção entre as operações realizadas durante o *Initial Key Agreement (IKA)*, que ocorre apenas durante a geração da chave inicial do grupo no momento de sua criação e o *Auxiliary Key Agreement (AKA)*, que ocorre nas operações subsequentes.

3.3.1 Método de Funcionamento

O *CLIQUES* utiliza o Algoritmo *Diffie-Hellman* como o algoritmo básico para a geração das chaves, o que leva ao descarte de um modelo de geração de chaves centralizado. Um modelo totalmente distribuído apesar de desejável também é descartado pois implicaria na igualdade das operações de *IKA* e *AKA*. Estes dois fatos levam a adoção de um modelo parcialmente distribuído para a geração das chaves. Além disso, o *CLIQUES* opta pela utilização de um controlador de grupo que será utilizado apenas nas operações de adição e remoção de membros do grupo.

Como dito no início desta seção, operações de *IKA* requerem o contato de cada membro do grupo e ocorrem apenas durante a sua formação inicial, momento em que o *overhead* do protocolo de entrada dos membros no grupo deve ser minimizado pois o mesmo precederá qualquer comunicação segura. No entanto em grupos estáticos ou com baixa dinamicidade os protocolos de *AKA* podem ser desnecessários. As operações de *AKA* envolvem a adição ou remoção de um único membro do grupo possuindo os

seguintes requisitos:

- Um novo membro do grupo não pode ter a habilidade de calcular chaves usadas anteriormente à sua entrada no grupo;
- Um membro que sair do grupo não pode ter a habilidade de calcular as chaves que serão usadas após a sua saída.

A base de funcionamento do *CLIQUES* está no protocolo *GDH.2* [19]. Este protocolo necessita de n ciclos, onde n representa o número de membros no grupo. O ciclo n é um estágio especial onde a nova chave do grupo é calculada. Os ciclos 1 a $n-1$ ocorrem como segue:

1. M_i recebe uma seqüencia de $i-1$ valores, chamados de valores intermediários, e um valor cardinal, K_{i-1}, que é uma chave do grupo para os $i-1$ membros anteriores;
2. M_i gera sua chave secreta N_i;
3. M_i eleva cada valor recebido a potência de N_i;
4. M_i adiciona K_i ao conjunto de valores intermediários;
5. M_i calcula o novo valor cardinal $K_i = (K_{i-1})^{N_i}$;
6. Se $i < n$, M_i envia K_i e o conjunto de valores intermediários para M_{i+1};

No ciclo n, M_n envia os $n-1$ valores intermediários para o todo o grupo, exceto o valor cardinal. Cada membro M_i exponencia o seu valor intermediário (a chave do grupo para os outros $n-1$ membros) com N_i gerando a chave final do grupo.

3.3.2 Protocolos do CLIQUES

No protocolo de entrada no grupo, o novo membro (M_{n+1}) torna-se o controlador do grupo. M_n estende o protocolo de *IKA* por um ciclo, gerando um novo expoente N'_n. Este expoente é utilizado (ao invés de N_n) para criar uma nova memsagem de *upflow* que é enviada a M_{n+1}. Neste momento M_n realiza a mesma sequencia de operações do último ciclo do *IKA*, gerando as $n-1$ sub-chaves que serão enviadas em uma mensagem aos membros do grupo, que por sua vez irão recalcular a nova chave do grupo.

A única entidade com direitos para excluir um membro do grupo é o seu controlador. Neste protocolo, N_n (o controlador do grupo) executa novamente o último ciclo do protocolo de *IKA*. Para tanto ele gera um novo expoente N'_n e utiliza a última mensagem referente a adição de um membro recebida (ou última mensagem do protocolo de *IKA*), para gerar um novo conjunto de $n-2$ chaves intermediárias e envia a mensagem resultante para o grupo. No caso da exclusão do controlador do grupo, o membro M_{n-1} assume a função de controlador do grupo e o resto do protocolo ocorre de forma semelhante.

4 *MKey: Um Protocolo para o Gerenciamento de Chaves de Grupo*

Neste capítulo introduzimos o sistema de gerenciamento de chaves de grupo proposto. Para tanto, primeiramente é apresentada a estrutura lógica utilizada para a representação do grupo, bem como a justificativa de sua utilização. Também são apresentados os algoritmos que compõem o MKey, a saber, o algoritmo de entrada de membros no grupo, o algoritmo de saída de membros do grupo e o algoritmo de recálculo de chaves do grupo. O conjunto dos três algoritmos implementam o MKey. O algoritmo de recálculo de chaves do grupo é baseado em uma extensão do algoritmo *Diffie-Hellman* [6], que o adapta para o caso de N, ao invés de 2 participantes, desenvolvida no decorrer deste trabalho. A seguir são apresentadas as primitivas de serviços de rede utilizadas e as mensagens trocadas pelos membros do grupo, constituindo o protocolo MKey.

4.1 Estrutura do Grupo

No MKey um grupo é representado por uma árvore binária com as seguintes características:

- os membros do grupo são representados por folhas;
- nós intermediários armazenam chaves intermediárias geradas no decorrer da formação do grupo. Estas chaves podem ser utilizadas para promover comunicação segura aos seus descendentes na árvore;
- a raiz da árvore detém a chave atual do grupo sendo de conhecimento de todos

os membros do grupo, e apenas deles.

A figura 9 exemplifica um grupo MKey. Neste exemplo temos:

- três usuários, a saber U_1, U_2, U_3;
- um nó intermediário $K_{1,2}$ que pode ser utilizado por U_1 e U_2 como uma chave de subgrupo. Chaves intermediárias possuem características idênticas às da chave do grupo;
- a raiz da árvore, que armazena a chave do grupo $K_{1,2,3}$.

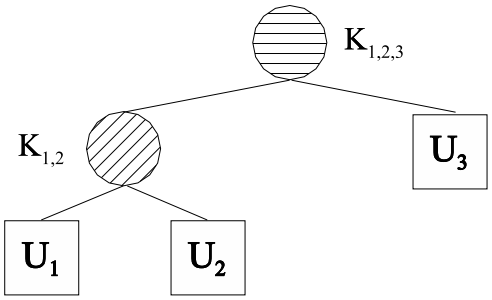

Figura 9: Árvore MKey com três usuários

A estruturação do grupo em uma árvore binária irá minimizar o número de operações necessárias ao algoritmo de recálculo de chaves do grupo que será executado sempre que houver uma alteração na estrutura do mesmo. Um número reduzido de operações a serem realizadas quando da alteração da estrutura do grupo é fator preponderante no que tange a eficiência dos algoritmos propostos.

4.2 Algoritmos do MKey

Nesta sessão serão apresentados os algoritmos que constituem o MKey a saber: o algoritmo de entrada de membros do grupo, o algoritmo de recálculo de chaves do grupo e o algoritmo de saída de membros do grupo.

4.2.1 Algoritmo de Entrada de Membros no Grupo

As figuras 10 e 11 exemplificam o algoritmo de entrada de membros no grupo. Na figura 10 é mostrada a entrada do membro U_4 no grupo da figura 9. 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.

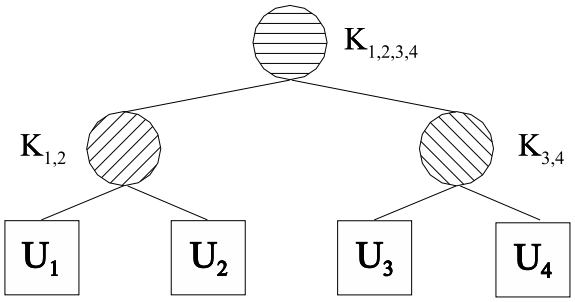

Figura 10: Entrada de U_4 no Grupo

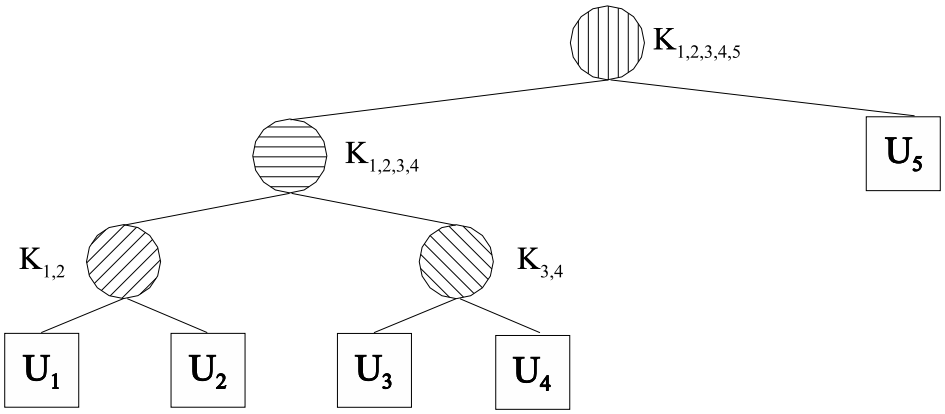

Figura 11: Entrada de U_5 no Grupo

A entrada do membro U_5, figura 11, 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 por ventura 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 $h = 5$. Como conseqüência do algoritmo de entrada de um novo membro no grupo, é disparada a execução do algoritmo de recálculo de chaves .

4.2.2 Algoritmo de Recálculo de Chaves na Entrada no Grupo

O objetivo deste algoritmo é de realizar o recálculo da chave do grupo, bem como das chaves intermediárias que necessitem ser atualizadas, sendo executado sempre que houver alguma alteração na estrutura do grupo. Devido aos algoritmos de criptografia assimétrica serem computacionalmente pesados, este algoritmo 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 ocorrência de alguma alteração na estrutura do grupo.

A primeira operação realizada por cada novo membro é o envio de uma mensagem *multicast* anunciando a sua chegada. A sintaxe e semântica de todas as mensagens trocadas por membros do MKey serão detalhadas no capítulo 5. Esta mensagem é recebida por cada um dos membros do grupo, sendo utilizada para determinar a chegada de um novo membro.

Na entrada de novos membros o MKey exige a realização de uma única operação de recálculo de chave. Esta operação é baseada em uma extensão do algoritmo *Diffie Hellman*, desenvolvida no decorrer deste trabalho, sendo executada de forma independente por cada membro do grupo. Para garantir a segurança das operações (envolvendo chaves privadas de usuários ou do grupo) os membros que já integravam o grupo (membros antigos) irão realizar o recálculo de forma diferente à do novo membro. As operações de recálculo para a entrada de um novo membro são descritas a seguir:

Passo 1 (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 $Y_1 = r^{X_1} \bmod p$.

Passo 2: Na entrada do segundo membro o recálculo das chaves ocorrerá como segue:

Novo membro (U_2):

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

Membro antigo (U_1):

- Recebe Y_2
- Calcula $X_{1,2} = Y_2^{X_1} \bmod p$; $Y_{1,2} = r^{X_{1,2}} \bmod p$.

Passo 3: Na entrada do terceiro membro o recálculo das chaves ocorrerá como segue:

Novo membro (U_3):

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

Membros antigos (U_1, U_2):

- Recebem Y_3
- Calculam $X_{1,2,3} = Y_3^{X_{1,2}} \bmod p$; $Y_{1,2,3} = r^{X_{1,2,3}} \bmod p$.

Passo N+1: De uma forma mais genérica, na entrada do membro U_{n+1}, o recálculo ocorrerá da seguinte maneira:

Membros antigos (U_1, ..., U_n):

- Recebem Y_{n+1}
- Calculam $X_{1,\ldots,n+1} = Y_{n+1}^{X_{1,\ldots,n}} \bmod p$; $Y_{1,\ldots,n+1} = r^{X_{1,\ldots,n+1}} \bmod p$.

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 $X_{1,\ldots,n}$ e à chave pública Y_{n+1} do novo usuário. Já o novo usuário (U_{n+1}) calcula as chaves do grupo da seguinte maneira:

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

Para comprovar a corretude matemática do algoritmo devemos demonstrar que as duas formas de cálculo de $X_{1,\ldots,n+1}$ produzem valores idênticos, ou seja, demostraremos que $Y_{n+1}^{X_{1,\ldots,n}} \bmod p = Y_{1,..,n}^{X_{n+1}} \bmod p$.

$$\begin{aligned} Y_{n+1}^{X_{1,\ldots,n}} \bmod p &= Y_{1,..,n}^{X_{n+1}} \bmod p \\ &= (r^{X_{1,..,n}} \bmod p)^{X_{n+1}} \bmod p \\ &= r^{X_{1,..,n} X_{n+1}} \bmod p \\ &= r^{X_{n+1} X_{1,..,n}} \bmod p \\ &= (r^{X_{n+1}} \bmod p)^{X_{1,..,n}} \bmod p \\ &= Y_{n+1}^{X_{1,\ldots,n}} \bmod p \end{aligned} \tag{4.1}$$

Esta prova valida a corretude matemática do algoritmo de recálculo de chaves [1].

As chaves resultantes da execução do algoritmo possuem as mesmas propriedades matemáticas daquelas do *Diffie-Hellman* original, sendo inviável para um atacante o cálculo de uma chave privada a partir dos valores públicos do grupo.

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. A figura 12 mostra as operações a serem realizadas por cada membro, na entrada do oitavo usuário no grupo, para o recálculo da chave do grupo. As figuras 13 e 14 mostram as operações a serem executadas para o recálculos das chaves intermediárias.

[1] A propriedade (4.1) é provada pelas propriedades da aritmética modular [18]

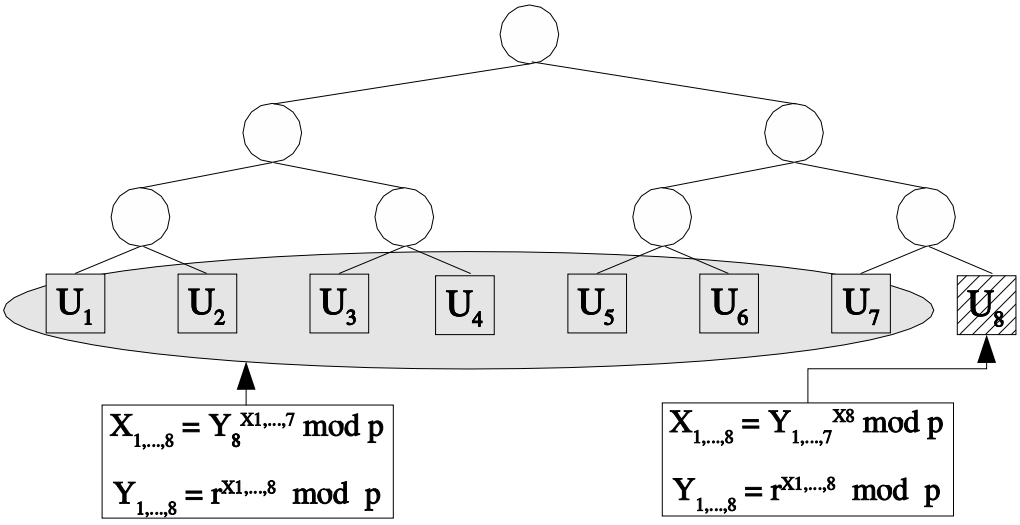

Figura 12: Recálculo da chave do grupo

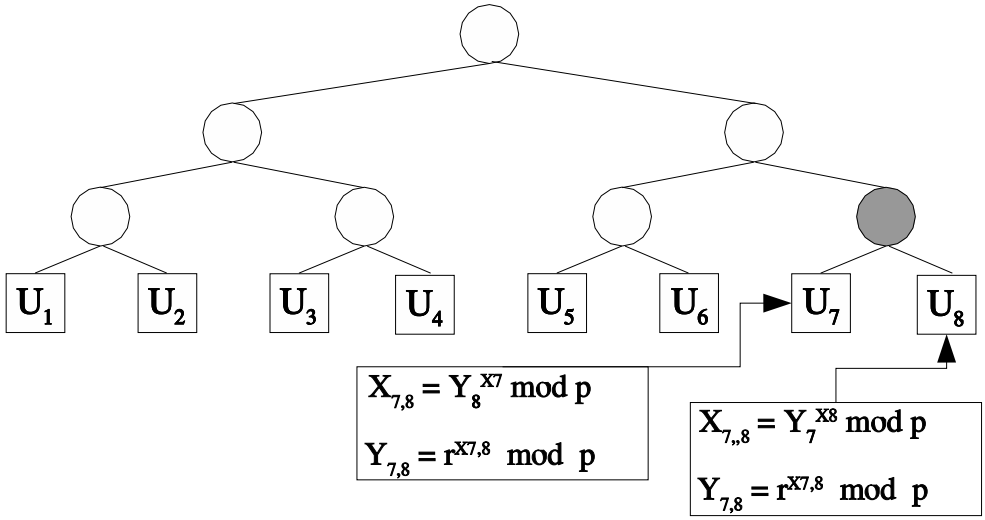

Figura 13: Recálculo das chaves intermediárias

4.2.3 Algoritmo de Saída de Membros do Grupo

No momento da saída de um membro as chaves atuais do grupo devem ser recalculadas, dado que:

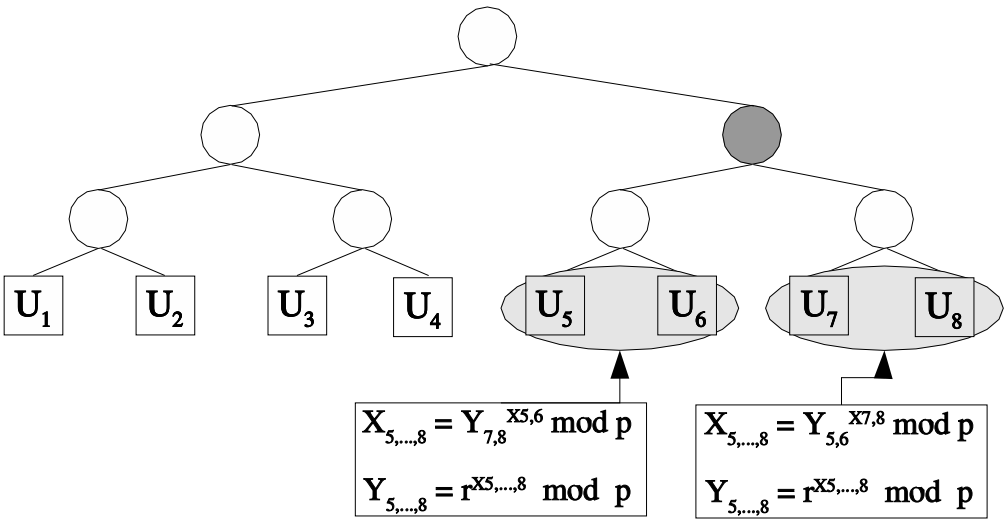

Figura 14: Recálculo das chaves intermediárias

1. O membro que deixou o grupo ainda terá acesso as mensagens trocadas, dado que ele possui as chaves atuais do grupo;

2. Mesmo com a entrada de novos membros no grupo, as novas chaves podem ser calculadas pelo membro excluído. Para tanto basta a obtenção das chaves públicas dos novos membros e a realização das seguintes operações: $X_{1,\ldots,n+1} = Y_{n+1}^{X_{1,\ldots,n}} \mod p$; $Y_{1,\ldots,n+1} = r^{X_{1,\ldots,n+1}} \mod p$.

Os dois fatos descritos no parágrafo anterior também implicam que, quando da saída de um membro, a execução de uma operação semelhante à realizada na entrada de novos membros não é suficiente para garantir a segurança das novas chaves geradas. Isto leva a necessidade da definição de um algoritmo específico para a saída de membros do grupo, algoritmo 4.1.

Neste algoritmo, as chaves são recalculadas seguindo-se o caminho entre o avô do membro a ser removido até a raiz da árvore. Cada nó no caminho entre o avô e a raiz da árvore terá seu par de chaves recalculado utilizando-se para tanto a chave pública de um de seus filhos e a chave privada do outro.

A utilização das chaves privadas intermediárias inviabiliza o recálculo da nova chave do grupo por parte do membro que o deixou.

Algorítmo 4.1 Algoritmo de saída de membros do grupo

Requer: p é o número primo gerado na inicialização do algoritmo de entrada de membros no grupo
Requer: r é a raiz primitiva de p gerada na inicialização do algoritmo de entrada de membros no grupo
Requer: *member* é o membro a ser removido do grupo
$node = avo(member)$;
if $node = root()$ **then**
if no atual pertence à subárvore a direita de *node* **then**
a=chave_publica_do_filho_a_esquerda_de_node();
b=chave_privada_do_filho_a_direita_de_node();
else
a=chave_publica_do_filho_a_direita_de_node();
b=chave_privada_do_filho_a_esquerda_de_node();
end if
$X_{m,\ldots,n} = a^b \bmod p$
$Y_{m,\ldots,n} = r^{X_{m,\ldots,n}} \bmod p$
end if
while $node \neq root()$ **do**
if no atual pertence à subárvore a direita de *node* **then**
a=chave_publica_do_filho_a_esquerda_de_node();
b=chave_privada_do_filho_a_direita_de_node();
$X_{m,\ldots,n} = a^b \bmod p$
$Y_{m,\ldots,n} = r^{X_{m,\ldots,n}} \bmod p$
else
a=chave_publica_do_filho_a_direita_de_node();
b=chave_privada_do_filho_a_esquerda_de_node();
$X_{m,\ldots,n} = a^b \bmod p$
$Y_{m,\ldots,n} = r^{X_{m,\ldots,n}} \bmod p$
end if
$node = pai(node)$;
end while

O algoritmo de saída de um membro do grupo é exemplificado na figura 15. 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 recáculo das chaves do grupo primeiramente é recalculado o par de chaves do nó $K_{1,2,4}$ da seguinte maneira:

- Membros a direita de $K_{1,2,4}$ (U_4):
  $X_{1,2,4} = Y_{1,2}^{X_4} \bmod p$
  $Y_{1,2,4} = r^{X_{1,2,4}} \bmod p$

- Membros a esquerda de $K_{1,2,4}$ (U_1, U_2):
  $X_{1,2,4} = Y_4^{X_{1,2}} \bmod p$
  $Y_{1,2,4} = r^{X_{1,2,4}} \bmod p$

De posse desse novo par de chaves intermediárias, o novo conjunto de chaves do grupo é calculado da seguinte forma:

- Membros a direita de $K_{1,2,4,5}$ (U_5):
  $X_{1,2,4,5} = Y_{1,2,4}^{X_5} \bmod p$
  $Y_{1,2,4,5} = r^{X_{1,2,4,5}} \bmod p$

- Membros a esquerda de $K_{1,2,4,5}$ (U_1, U_2, U_4):
  $X_{1,2,3,5} = Y_5^{X_1,2,4} \bmod p$
  $Y_{1,2,4,5} = r^{X_{1,2,4,5}} \bmod p$

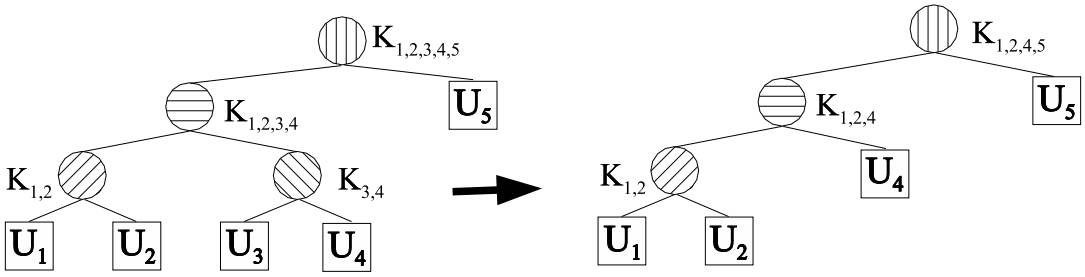

Figura 15: Saída do Grupo

4.3 protocolo MKey

Nesta seção são descritas as primitivas de serviço de rede utilizadas pelo MKey, o conjunto de mensagens trocadas pelos membros de um grupo e a sintaxe e semântica a elas associadas.

4.3.1 Primitivas de Serviço

O Nkey não realiza nenhuma alteração nas primitivas de rede disponibilizadas pelo Sistema Operacional, utilizando a API de programação de *sockets multicast* como disponibilizadas pelo mesmo. Cada membro do grupo faz uso da função *setsockopt* da biblioteca padrão de *sockets* do Linux. Na entrada do membro no grupo é realizada uma chamada da função utilizando-se o parâmetro `IP_ADD_MEMBERSHIP`. Na saída do grupo o membro realiza uma nova chamada da função utilizando o parâmetro `IP_DROP_MEMBERSHIP`.

4.3.2 Diagramas de Mensagens

A figura 16 mostra o diagrama das mensagens trocadas para a entrada de um novo membro no grupo. As mensagens a serem trocadas são descritas a seguir:

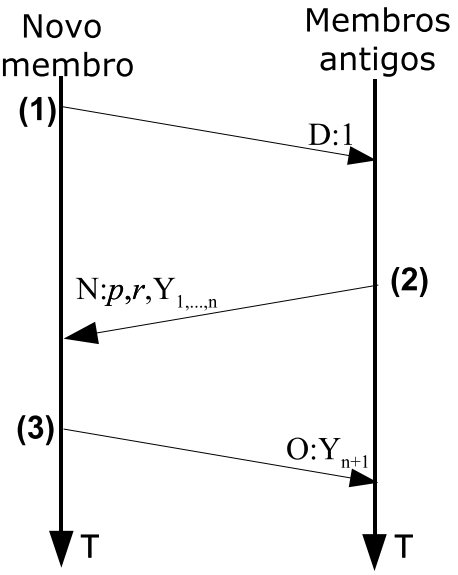

Figura 16: Diagrama de mensagens para a entrada de um membro no grupo

- Primeiramente (1), o novo membro envia uma mensagem *multicast* de forma a informar a sua entrada no grupo;
- Em (2) o novo membro recebe os valores públicos necessários para sua inicialização, bem como para possibilitar o cálculo das novas chaves do grupo.

- Finalmente (3) o novo membro envia uma nova mensagem contendo a sua chave pública. De posse dessa informação os membros antigos também realizam o recálculo das chaves do grupo.

Na figura 17 é mostrado o diagrama de troca de mensagens para a saída de um membro do grupo. Para tanto é necessário apenas o envio, por parte do membro que irá deixar o grupo, de uma mensagem do tipo *'d'* seguido de sua chave pública (4).

A seqüencia de eventos que envolvem trocas de mensagens entre membros para a formação de um grupo com três membros é mostrada na figura 18.

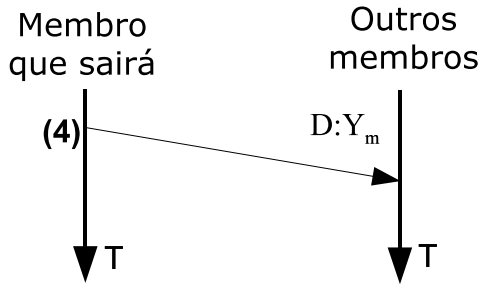

Figura 17: Diagrama de mensagens para a saída de um membro no grupo

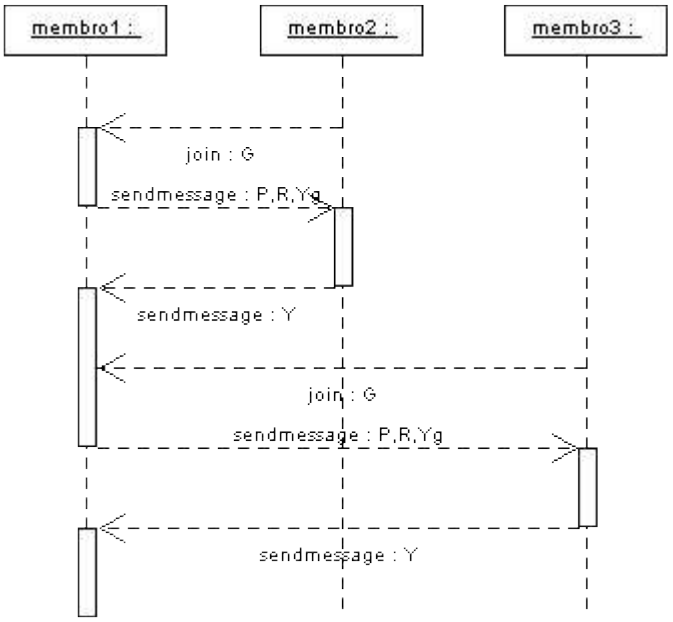

Figura 18: Diagrama de seqüencia

5 *Implementação dos Algoritmos e Protocolos Propostos*

Para validar o funcionamento dos algoritmos e protocolos de entrada e saída do grupo, bem como do algorítmo de recálculo de chaves apresentados no capítulo 4 foi desenvolvido um software que implementa os algortimos e o protocolo MKey.

5.1 Arquitetura da Implementação do MKey

Esta implementação foi desenvolvida na linguagem de programação *C*, sobre o sistema operacional *Linux*, sendo constituído pelos seguintes componentes:

prime: Este componente, figura 19, implementa um gerador aleatório de números primos, estando sua operação relacionada com a fase de inicialização do protocolo. O teste de primalidade é executado de forma determinística implicando em um menor desempenho, mas fortalecendo a segurança das operações subsequentes do protocolo.

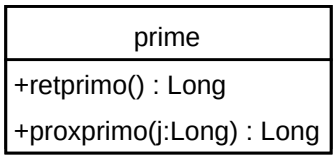

Figura 19: Componente *prime*

primitiveroot: Este componente, também relacionado com a fase de inicialização do protocolo, disponibiliza uma função para a geração de uma raiz primitiva de um

número primo. O valor de um número primo p é fornecido como entrada para o algoritmo que retornará uma raiz primitiva do mesmo. A figura 20 mostra a interface do componente.

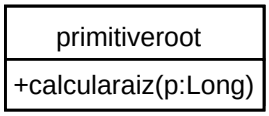

Figura 20: Componente *primitiveroot*

fastexp: Este componente, figura 21 , disponibiliza uma função para o cálculo (de forma eficiente) de uma exponenciação da forma $c = a^b \bmod p$. Os valores a, b e p (um número primo) devem ser providos como entrada para o algoritmo. O cálculo de c ocorre de forma linear com o tamanho de a, b e p.

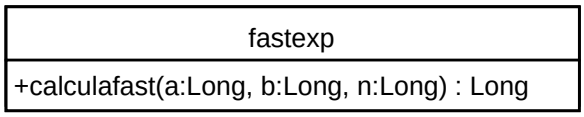

Figura 21: Componente *fastexp*

user: Implementa um membro do grupo. Disponibiliza funções para o cálculo das chaves pública / privada do membro (no momento de sua entrada), bem como para o recálculo das chaves do grupo à medida que ocorram entradas ou saídas de membros do mesmo. Cada instância do *software* implementa um membro do grupo, podendo haver diversas instâncias em uma mesma máquina. A interface deste componente é mostrada na figura 22. A comunicação entre os membros é realizada pelo envio/recebimento de mensagens *multicast* sobre o protocolo de transporte UDP. Nesta implementação foram utilizados endereços de grupo com abrangência local, porém a operação em um ambiente geograficamente distribuído é imediata, bastando a utilização de endereços de grupo globais e roteadores *multicast*. Todos os membros que formam um grupo possuem uma "visão"atualizada da árvore utilizando-a para, por exemplo, determinar o ponto de inserção de um novo membro ou as chaves a serem recalculadas quando da saída de um membro.

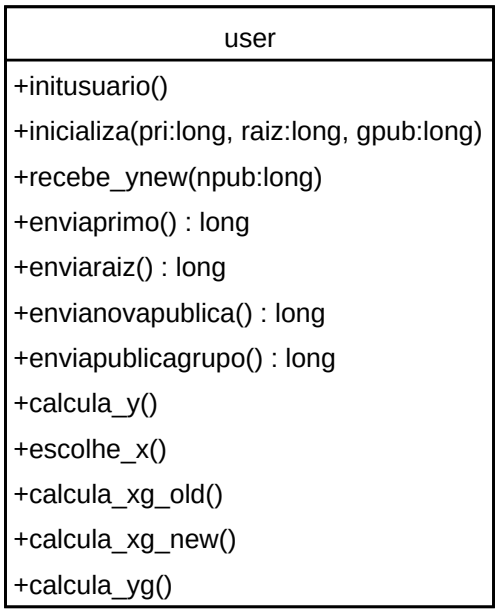

Figura 22: Componente *user*

sendreceive: Neste componente estão implementadas as funções que possibilitam a comunicação entre os membros de um grupo, compreendendo:

- abertura e encerramento de *sockets multicast*: Realiza o *setup* necessário para a abertura dos *socket´s* e para o envio e recebimento dos pacotes;
- envio e recebimento de mensagens UDP: A comunicação entre os membros do grupo ocorre somente pela troca de mensagens *multicast* UDP. Para tanto foram definidas uma série de mensagens de controle com uma sintaxe e semântica associadas, sendo sua interpretação realizada diretamente por cada instância do *software*;
- codificação e decodificação de mensagens: A sintaxe das mensagens de controle foi definida como uma série de campos separados pelo delimitador ':'. O primeiro campo determina o tipo da mensagem, sendo representado por um caracter ASCII. Os tipos atualmente definidos são mostrados na tabela 3:
  - as mensagens do tipo *'n'* possuem três campos adicionais: O número primo (p), a raiz primitiva de p e a chave pública atual do grupo ($Y_{1,\ldots,n}$).
  - as mensagens do tipo *'o'* possuem um campo adicional que representa a chave pública do novo usuário.

Tipo da mensagem	Semântica associada
n	Mensagem para novo usuário
o	Mensagem para usuários antigos
d	Indica a entrada ou saída de um membro

Tabela 3: Tipos de mensagem de controle do MKey

– as mensagens do tipo *'d'* também possuem um campo adicional. Na entrada de um membro seu valor será igual a 1 e na saída de um membro, seu valor será igual a sua chave pública.

A interface do componente **sendreceive** é mostrada na figura 23.

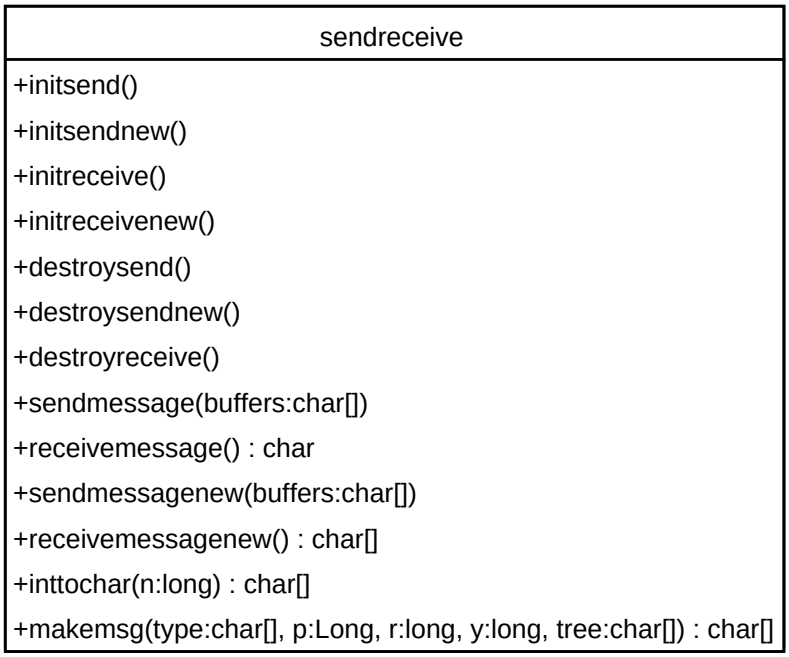

Figura 23: Componente *sendreceive*

tree: Este componente implementa a árvore que representa o grupo. Ele implementa a estrutura da árvore e uma série de funções relacionadas com a manipulação da mesma. Nesta implementação todos os nós da árvore possuem uma mesma estrutura. A diferenciação entre nós intermediários e membros efetivos do grupo é feita pelo fato dos membros serem as folhas da árvore. A figura 24 mostra a interface deste componente.

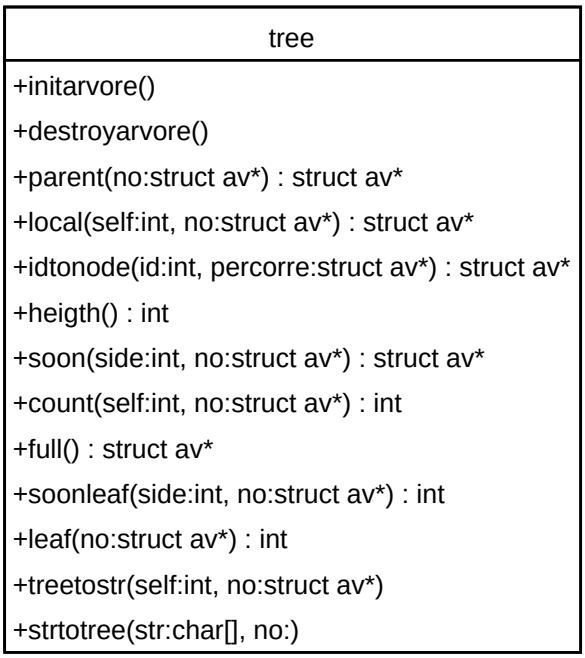

Figura 24: Componente *tree*

newuser: Este componente, figura 25, realiza a inserção de um novo membro na árvore. Para tanto primeiramente ele determina o ponto de inserção do novo membro na árvore, após isso ele cria um nó (que representa o novo membro) e o insere na árvore.

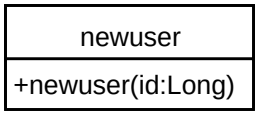

Figura 25: Componente *newuser*

deluser: Este componente realiza a remoção de um membro da árvore. Ele primeiramente remove o nó da árvore, realizando os ajustes na estrutura da árvore que se tornarem necessários. A seguir, cada membro restante irá realizar de forma independente o recálculo das novas chaves do grupo. A interface deste componente é mostrada na figura 26.

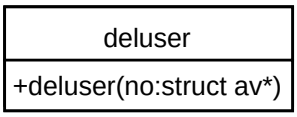

Figura 26: Componente *deluser*

Uma visão geral da arquitetura do MKey demostrando as dependências entre os componentes é dada na figura 27. O componente ***main***, não descrito na seção anterior apenas implementa o programa principal, utilizando para tanto as funções disponibilizadas por outros componentes.

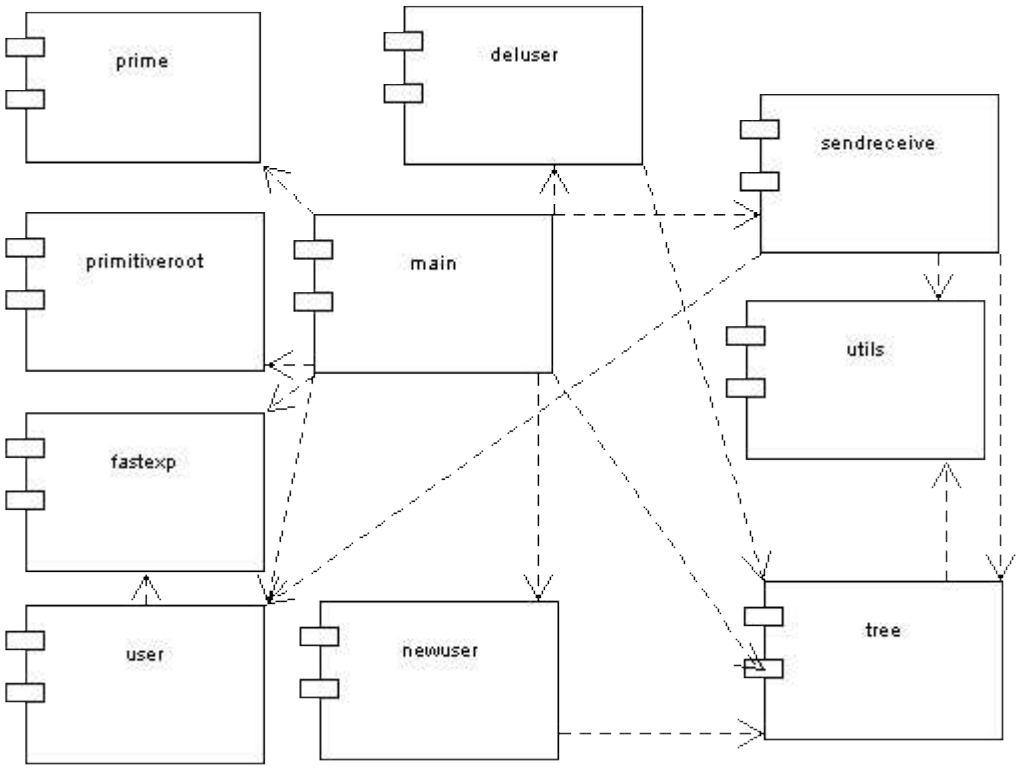

Figura 27: Diagrama de componentes

O primeiro membro a entrar em um grupo é responsável pela geração dos valores públicos iniciais a serem usados por todos i os membros do grupo. Para tanto ele utiliza as funções *retprimo()* e *calculararaiz()*, disponíveis nos componentes ***prime*** e ***primitiveroot***, respectivamente. A seguir ele escolhe a sua chave privada e calcula a

sua chave pública, utilizando as funções do componente ***u*ser**. Depois disso, o membro realiza a sua entrada efetiva no grupo chamando a função *newuser()*. Cabe salientar que o grupo é criado quando o primeiro membro executa as funções que acabaram de ser descritas. Após a entrada no grupo, o membro entra em um *loop* infinito onde aguarda pela chegada de alguma mensagem. Cada mensagem recebida é decodificada e as ações necessárias são realizadas, tais como o envio de valores públicos ou o recálculo de uma nova chave de grupo.

No caso do membro não ser o primeiro do grupo, no momento de sua instanciação, ele envia uma mensagem (*sendmessage()*) anunciando a sua chegada no grupo. Como resposta, ele recebe os valores do número primo e de sua raiz primitiva (receivemessage()). De posse destes valores ele escolhe a sua chave privada e calcula a sua chave pública, também utilizando para tanto funções do componente ***u*ser**. A seguir ele envia o valor de sua chave pública para o grupo (*sendmessage()*), de forma que cada membro possa calcular a nova chave do grupo. Então ele entra em um loop infinito, idêntico ao do primeiro membro do grupo.

5.2 Teste de Execução

Um exemplo de execução é mostrada a seguir. Neste exemplo é formado um grupo de quatro membros, com ordem de entrada `membro1`, `membro2`, `membro3` e `membro4`. Após as quatro operações de entrada `membro2` deixa o grupo. O exemplo é mostrado pela captura da tela de saída das quatro instâncias do software.

Tela de saída: membro1

```
[carlosgr@carcara MKey-c]$ ./MKey init
I'm the first user in the group
Calculated :
Prime = 13921
Primitive root = 13914
Private user key = 11645
Public user key = 7346
_____Waiting for users events_____

New user has arrived
Received:
New user public key = 3255
Calculated:
New group private key = 12909
New group public key = 6168
_____Waiting for users events_____

New user has arrived
Received:
New user public key = 10316
Calculated:
New group private key = 2407
New group public key = 3670
_____Waiting for users events_____

New user has arrived
Received:
New user public key = 12647
Calculated:
New group private key = 13811
New group public key = 12509
_____Waiting for users events_____

Removing user 3255 ...
Calculated:
New group private key = 3016
New group public key = 12239
_____Waiting for users events_____
```

Tela de saída: membro2

```
[carlosgr@carcara MKey-c]$ ./MKey notinit
I'm not the first user in the group
Reveived :
Prime = 13921
Primitive root = 13914
(Old) Public group key = 7346
Calculated :
Private user key = 10925
Public user key = 3255
Private group key = 12909
Public group key = 6168
_____Waiting for users events_____

New user has arrived
Received:
New user public key = 10316
Calculated:
New group private key = 2407
New group public key = 3670
_____Waiting for users events_____

New user has arrived
Received:
New user public key = 12647
Calculated:
New group private key = 13811
New group public key = 12509
_____Waiting for users events_____

Caught a signal. Going out !
[carlosgr@carcara MKey-c]$
```

Tela de saída: `membro3`

```
[carlosgr@carcara MKey-c]$ ./MKey notinit
I'm not the first user in the group
Reveived :
Prime = 13921
Primitive root = 13914
(Old) Public group key = 6168
Calculated :
Private user key = 10487
Public user key = 10316
Private group key = 2407
Public group key = 3670
_____Waiting for users events_____
New user has arrived
Received:
New user public key = 12647
Calculated:
New group private key = 13811
New group public key = 12509
_____Waiting for users events_____
Removing user 3255 ...
Calculated:
New group private key = 3016
New group public key = 12239
_____Waiting for users events_____
```

Tela de saída: `membro4`

```
[carlosgr@carcara MKey-c]$ ./MKey notinit
I'm not the first user in the group
Reveived :
Prime = 13921
Primitive root = 13914
(Old) Public group key = 3670
Calculated :
Private user key = 12835
Public user key = 12647
Private group key = 13811
Public group key = 12509
_____Waiting for users events_____
Removing user 3255 ...
Calculated:
New group private key = 3016
New group public key = 12239
_____Waiting for users events_____
```

O teste mostrou que o software comportou-se como esperado. As chaves foram calculadas conforme o previsto e as sequências de mensagens do protocolo foram geradas e intercambiadas corretamente. Foram realizados testes adicionais com outras configurações e em todos eles o software executou suas funções corretamente. O estudo do comportamento do protocolo considerando variações de parâmetros de rede, bem como sua validação através do uso de técnicas de descrição formais foi deixada para trabalhos futuros.

6 *Conclusões e Perspectivas Futuras*

Neste capítulo são apresentadas as contribuições decorrentes deste trabalho, as conclusões obtidas após o desenvolvimento do mesmo e as perspectivas de trabalhos futuros.

6.1 Contribuições e Conclusões

Este trabalho apresentou um conjunto de algoritmos e um protocolo que permite o cálculo e gerência de chaves de grupos. O protocolo proposto possui como principal contribuição o fato de serem adequados a grupos dinâmicos, dado que as operações de entrada e saída do grupo são computacionalmente leves. Uma comparação com os principais trabalhos correlatos encontrados nesta área é apresentada a seguir.

6.1.1 Comparação com trabalhos Correlatos

Apesar de ser uma área relativamente nova, podemos encontrar na literatura diversos trabalhos que abordam o assunto de comunicação segura de grupos. Uma comparação entre estes trabalhos deve levar em consideração pelo menos, os seguintes parâmetros:

Iniciação: Número de operações a serem realizadas para a iniciação 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 para o recálculo da chave do grupo;

Mensagens: Número e tamanho das mensagens a serem enviadas quando da alteração da estrutura do grupo;

Centralização: O esquema possui ou não um controlador de grupo;

Tamanho do chaveiro : Número máximo de chaves a serer armazenadas por cada membro do grupo.

Os três primeiros parâmetros irão influenciar diretamente na eficiência do sistema, enquanto que os dois últimos na sua tolerância a falhas e confiabilidade.

Os protocolos GDH [19] são os únicos que possuem uma fase de iniciação. Isto se deve ao fato de que eles pressupõem e foram concebidos considerando que conjunto aproximado dos membros que irão formar o grupo é conhecido no momento de sua formação. Para o GDH.3, o de menor complexidade, no processo de iniciação são necessários $5n - 6$ recálculos e $2n - 1$ trocas de mensagens, sendo n o número de membros que irão formar inicialmente o grupo. Após a fase de iniciação a entrada de cada novo membro requer $2n + 1$ recálculos (n para o novo membro e 1 para cada membro antigo) e 2 trocas de mensagens, sendo uma de M_n (o membro que entrou por último no grupo) para M_{n+1} (o novo membro) e uma mensagem *broadcast* para M_1,...,M_n, onde M_i é o iésimo membro a entrar no grupo. Para a saída de um membro são realizados $2n-2$ recálculos ($n-1$ por M_n e 1 para cada membro que irá permanecer no grupo) e uma mensagem *broadcast* é enviada de M_n para os membros restantes no grupo.

Para a entrada e saída de um membro do grupo o GDH assume que M_n salva o conteúdo das mensagens trocadas na última execução do protocolo, o que implica em um chaveiro contendo n sub-chaves para o membro M_n. Isto também implica que as mensagens trocadas para a entrada ou saída de um membro possuem n sub-chaves. O tamanho das mensagens cresce linearmente com o tamanho do grupo. O chaveiro do membro M_n apresenta-se como um ponto atrativo para ataques de força bruta, dado que a chave do grupo pode ser calculada a partir de qualquer uma das n sub-chaves contidas em seu chaveiro. Um atacante que comprometa M_n tera uma chance maior de calcular a chave do grupo. Além disso, o tamanho das mensagens a serem trocadas em um grupo com um número grande de membros torna-se um fator limitante para o uso do GDH.

No GDH não fica claro a forma utilizada para o envio das mensagens ditas broadcast para conjuntos de membros do grupo. Também merece nota o fato dos membros do

grupo responsáveis pelo envio destas mensagens (M_{n+1} no caso de uma entrada e M_n no caso de uma saída do grupo) desempenharem um papel especial no protocolo de recálculo das chaves, constituindo-se em pontos críticos de falha.

O TGDH [7] é o único dos protocolos que avaliamos em detalhes que possui explicitamente a figura de um controlador de grupo, *o sponsor*, que desempenha um papel fundamental no recálculo da chave do grupo. A centralização cria um ponto crítico de falha no protocolo, pois uma falha no *sponsor* provocará o impedimento da execução de alterações na estrutura do grupo. No TGDH são necessárias, no total, 3 mensagens e $n(3h/2)$ recálculos a cada entrada de um novo membro no grupo, onde n representa o número de membros no grupo, e h a altura da árvore ($3h/2$ recálculos para cada membro [3]). Na saída de um membro são realizados, no total, $(n-1)(3h/2)$ recálculos e 1 mensagem é enviada.

No TGDH cada membro conhece as chaves no caminho da folha que o representa até a raiz da árvore, bem como todas as *blinded keys* da árvore. No pior caso há um total de $(2^h - 1) + H_i$ chaves armazenadas por M_i. Nos protocolos de entrada e saída do membro do grupo, o *(sponsor)* realiza um *broadcast* do novo conjunto de *blinded keys* para os outros membros do grupo, o que leva a um crescimento desta mensagem linearmente com o número de nós da árvore. No pior caso, esta mensagem possuirá um tamanho equivalente às $(2^h - 1)$ *blinded keys* da árvore.

O MKey possui uma estrutura descentralizada não havendo a figura do controlador do grupo. Dado que todas as operações de entrada são tratadas de forma idêntica, também não há uma fase de iniciação. Para o recálculo da chave do grupo, na entrada de um novo membro, o MKey realiza exatamente 2 trocas de mensagens (sendo uma do grupo para o novo membro informando os valores de p, r e $Y_{1,\ldots,n}$ e uma do novo membro para o grupo, informando o valor de Y_{n+1}). Em relação ao número de recálculos, todas as chaves no caminho entre o primeiro ascendente comum de cada M_i pertencente ao grupo e M_{n+1}, até a raiz da árvore precisam ser recalculadas. No melhor caso (árvore completa), o número de recálculos a ser realizado será igual a n. No pior caso (árvore quase cheia) serão necessários $2^h - 2$ recálculos, onde n representa o número de membros no grupo, e h a altura da árvore. Para a saída de um membro do grupo são realizados $2^h - H - 2$ recálculos, onde H representa a altura da árvore, e uma mensagem é enviada.

O tamanho das mensagens trocadas no MKey é fixo, tendo o seu valor afetado somente pelo tamanho da chave escolhida. Cada membro do grupo necessita armazenar apenas as chaves no caminho da folha que o representa e a raiz da árvore, o que leva

a um total de h chaves armazenadas, no pior caso, por cada membro do grupo.

A tabela 4 apresenta uma comparação dos protocolos GDH, TGDH e MKey no que tange ao tamanho das mensagens trocadas e ao número de chaves armazenadas por cada membro do grupo.

	GDH	**TGDH**	**MKey**
Tamanho máximo das mensagens	linear com n	linear com $(2^h - 1)$	fixo
Tamanho máximo do chaveiro	n	$(2^h - 1) + H_i$	h

Tabela 4: Tamanho de mensagens e chaveiros dos membros do grupo

Assumindo a escolha de uma chave de 128 bits os tamanhos máximos das mensagens trocadas e dos chaveiros para um grupo com 1000 de membros são exemplificados a seguir.

Para o GDH, na entrada do membro de número 1001 será gerada uma mensagem contendo 1000 sub-chaves, cada uma calculada através da expressão $\alpha^{N_1,\ldots,N_i,N_{i+1},\ldots,N_{1000}}$ onde α e N_i são números de 128bits, e N_i representa a chave privada do membro i. O comprimento total da mensagem sera dado pelo somatório dos comprimentos de cada uma das 1000 sub-chaves. Dado que os valores de α e N_i possuem 128 bits de comprimento, para o grupo com 1000 menbros a mensagem gerada possuira mais que 128Kb. O chaveiro do membro M_{1000} irá possuir 1000 de chaves.

No TGDH o *sponsor* irá enviar uma mensagem contento todas as *blinded keys* da árvore ($2^h - 1$ chaves de 128 bits cada), resultando em uma mensagem de aproximadamente 256Kb. Cada membro possuirá em seu chaveiro cerca de 2000 de chaves.

O MKey possui um tamanho de mensagem fixo, neste caso de 128 bits e cada membro do grupo irá possuir um chaveiro com apenas 11 chaves.

Na tabela 5 é apresentada uma comparação entre as operações de entrada de um membro no grupo dos protocolos GDH, TGDH e MKey.

A tabela 6 mostra o número de recálculos a serem realizados para a entrada do milésimo membro em um grupo. Considera-se para efeito dos cálculos que todos os 1.000 membros entraram no grupo de forma seqüencial, mantendo as árvores do TGDH e MKey balanceadas. Como não há o conhecimento prévio dos membros que formarão o grupo, a fase de iniciação do protocolo GDH é usada apenas para os dois primeiros membros que iniciam a formação do grupo.

	GDH	**TGDH**	**MKey**
Mensagens na inicialização	$2n-1$	0	0
Mensagens após inicialização	2	2	2
Recálculos na inicialização	$5n-6$	0	0
Recálculos após inicialização	$2n+1$	$n(3h/2)$	2^h-2

Tabela 5: Comparação entre protocolos na entrada no grupo

	GDH	**TGDH**	**MKey**
Mensagens na inicialização	3	0	0
Mensagens após inicialização	2	2	2
Recálculos na inicialização	4	0	0
Recálculos após inicialização	2001	15000	2046

Tabela 6: Operações realizadas na entrada do milésimo membro

A tabela 7 apresenta a comparação no que tange a saída de um membro do grupo. O número de operações realizadas para a remoção de um membro em um grupo com 1.000 membros é mostrada na tabela 8. De forma semelhante à comparação anterior considera-se que as árvores do TGDH e MKey estão totalmente balanceadas.

	GDH	**TGDH**	**MKey**
Mensagens	1	1	1
Recálculos	$2n-2$	$(n-1)(3h/2)$	2^h-H-2

Tabela 7: Comparação entre protocolos na saída no grupo

	GDH	**TGDH**	**MKey**
Mensagens	1	1	1
Recálculos	1996	14985	2036

Tabela 8: Operações na saída do grupo com 999 membros restantes

O MKey realiza um número recálculos menor que o o TGDH e equivalente ao do GDH, tanto na operação de entrada como na saída de membros do grupo, sendo este o principal parâmetro a afetar a eficiência de um sistema de comunicação de grupo. As operações de entrada e saída de membros do grupo no MKey são menos complexas. Os chaveiros do MKey também são bem menores que os dos protocolos TGDH e GDH.

6.2 Trabalhos Futuros

Como continuidade do trabalho desenvolvido visamos realizar as seguintes atividades:

- Especificar os protocolos e algoritmos propostos usando uma técnica de especificação formal que irá permitir a realização de provas de propriedades como corretude e estabilidade;
- Realizar experimentos em ambientes WAN (geograficamente distribuídos) que levem em conta a ocorrência de eventos como: perda de mensagens UDP; retardo e variação de retardo na entrega das mensagens; falhas em links e membros de grupos; mensagens com erros; entre outros.
- No âmbito do Laboratório NatalNet, estão sendo desenvolvido sistemas de videoconferência e telemedicina, sendo nossa intenção utilizar o MKey como protocolo de gerenciamento de chaves nestes sistemas.

Referências Bibliográficas

[1] Federal Information Processing Standards Publication 46-2. Data Encryption Standard. Technical report, FIPS, 1988.

[2] D. Agarwal, O. Chevassut, M. R. Thompson, and G. Tsudikz. An integrated solution for secure group communication in wide-area networks. *6th IEEE Symposium on Computers and Communications*, 2001.

[3] Yair Amir, Yongdae Kim, Cristina Nita-Rotaru, and Gene Tsudik. On the performance of group key agreement protocols. *ICDCS 2002*, Julho 2002.

[4] Klaus Becker and Uta Wille. Communication complexity of group key distribution. *5th ACM Conference on Computer and Communications Security*, Novembro 1998.

[5] F. J. N. Cosquer and P. Veríssimo. Survey of selected groupware applications and supporting platforms. Technical report, INESC, 1994.

[6] Whitfield Diffie and Martin Hellman. New direction in cryptography. *IEEE Trans. on Inf. Theory*, pages 664–654, 1976.

[7] Yongdae Kim, Adrian Perrig, and Gene Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. *USC Technical Report 00-737*, Agosto 2000.

[8] Patrick McDaniel, Atul Prakash, and Peter Honeyman. Antigone: A flexible framework for secure group communication. *8th USENIX Security Symposium*, Agosto 1999.

[9] David A. McGrew and Alan T. Shermany. Key establishment in large dynamic groups using one way function trees. *Cryptographic Technologies Group TIS Labs at Network Associates Inc*, Maio 1998.

[10] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* RC PRESS, 1996.

[11] Information Technology Open Systems Interconnection Basic Reference Model: The Basic Model. www.iso.org. Acessado em janeiro de 2003.

[12] Refik Molva and Alain Pannetrat. Scalable multicast security in dynamic groups. *6th ACM Conference on Computer and Communications Security*, Novembro 1999.

[13] National Institute of Standards and Technology. www.nist.org. Acessado em janeiro de 2003.

[14] M. Peyravian, S.M. Matyas, and N. Zunic. Decentralized group key management for secure multicast communications. *Elsevier Computer Communications*, 1999.

[15] Information processing systems Open Systems Interconnection Basic Reference Model Part 2: Security Architecture. www.iso.org. Acessado em janeiro de 2003.

[16] Rivest, Shamir, and Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, Fevereiro 1978.

[17] Soares, Lemos, and Colcher. *Redes de Computadores: das LANs, MANs e WANs às redes ATM*. Editora Campus, 1995.

[18] William Stallings. *Cryptography and Network Security: Principles and practice*. Prentice Hall, 1998.

[19] Michael Steiner, Gene Tsudik, and Michael Waidner. Diffie-hellman key distribution extended to group communication. *ACM Symposium on Computer and Communication Security*, Março 1996.

[20] Michael Steiner, Gene Tsudik, and Michael Waidner. Cliques: A new approach to group key agreement. *IEEE ICDCS'97*, Maio 1997.

[21] Andrew S. Tanenbaum. *Sistemas Operacionais Modernos*. Guanabara Koogan, 1995.

[22] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure group communications using key graphs. *ACM SIGGCOM*, Setembro 1998.