CloudSec - Um Middleware para Compartilhamento de Informações Sigilosas em Nuvens Computacionais

Rick Lopes de Souza, Hylson Vescovi Netto, Lau Cheuk Lung, Ricardo Felipe Custódio

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

{rick.lopes, hylson.vescovi, lau.lung, custodio} @inf.ufsc.br

Abstract. *The need to share and manipulate sensitive data is a challenge for most content providers using the cloud for storage. However, developing applications that guarantee confidentiality in the cloud are complex and require integration of multiple aspects of security and interoperability. To circumvent these challenges, this paper aims to propose a middleware architecture to ensure secure sharing of documents using public cloud providers for data storage and hardware secure modules for cryptographic key management. This work has as main features: the use of identity-based encryption, use of hybrid clouds, simplified management of cryptographic keys, peer-to-peer security assurance and use of cryptographic security modules.*

Resumo. *A necessidade de compartilhar e manipular dados sensíveis é um desafio para grande parte dos provedores de conteúdo que utilizam a nuvem para armazenamento. No entanto, desenvolver aplicações que garamtam sigilo em nuvem é uma tarefa complexa e requer integração de múltiplos aspectos de segurança e interoperabilidade. Para contornar esses desafios, esse trabalho tem como principal objetivo propor uma arquitetura de middleware para garantir o compartilhamento seguro de documentos sigilosos em nuvem utilizando provedores de nuvens públicas para o armazenamento dos dados e módulos de segurança criptográficos para o gerenciamento de chaves criptográficas. Este trabalho tem como principais características: o uso da criptografia baseada em identidade, uso de nuvens híbridas, gerenciamento de chaves criptográficas simplificado, garantia de segurança ponto a ponto e utilização de módulos de segurança criptográficos.*

1. Introdução

A necessidade de garantir o sigilo utilizando provedores de nuvem pública para armazenamento é grande, entretanto, a implementação e integração de serviços para garantir todas as propriedades necessárias de segurança é complexo e pode comprometer o desenvolvimento de um produto, caso não seja bem feito. Propriedades como o correto e seguro gerenciamento das chaves criptográficas são essenciais em sistemas de sigilo, assim como tolerância a falhas [1] e a retirada da necessidade de se confiar inteiramente nos provedores de nuvens públicas.

Grande parte das empresas já possui um software para gerenciamento de documentos, assim como um controle de acesso já estabelecido. Entretanto, esses softwares

[1]Considera-se que uma falha poderá ocasionar um erro no sistema que levará a um defeito no funcionamento.

 ©2014 SBC — Soc. Bras. de Computação

carecem de mecanismos de segurança para proteger os dados dos usuários e corporações que os utilizam. O desenvolvimento de mecanismos de segurança não é uma tarefa trivial e quando feita por empresas não especializadas, pode levar a falhas de segurança, expondo assim dados sensíveis. Para que isso seja feito de uma maneira mais segura, necessita-se de um software que gerencie os mecanismos de segurança de uma forma transparente e integrando diferentes tipos de comunicações e operações criptográficas.

Problema: A dificuldade de integrar diferentes mecanismos de segurança de forma transparente à uma aplicação que deseja garantir o compartilhamento de informações sigilosas de forma segura e confiável. Dentre os mecanismos de segurança existentes, o gerenciamento de chaves criptográficas é um ponto crítico e deve ser tratado com mais atenção, assim como a tolerância a falha nos sistemas de distribuição de chaves criptográficas e armazenamento de dados em nuvens computacionais.

Motivação: A transparência na integração de serviços de segurança é essencial para outras aplicações que necessitam desses serviços. Grande parte das corporações não possuem conhecimento muito avançado sobre os conceitos de segurança da informação e, com isso, implementações com falhas podem ser colocada no mercado, podendo expor informações sensíveis de usuários e corporações. Dessa forma, necessita-se de soluções que envolvam proteções físicas e lógicas para o gerenciamento das chaves criptográficas dos usuários, fazendo com que seja oferecido segurança, disponibilidade e usabilidade. Além disso, deve-se utilizar mecanismos que garantam a tolerância a falhas e segurança na utilização dos provedores de nuvens públicas para armazenamento, fazendo com que estes não tenham condições de acessar os dados sigilosos dos usuários.

A partir do problema e motivação, conclui-se que são necessárias soluções para proteger de forma física e lógica o gerenciamento de chaves criptográficas dos usuários. Assim como outros mecanismos de segurança como tolerância a falhas para prover segurança, disponibilidade e usabilidade aos usuários. Essas soluções devem ser transparentes para as aplicações dos usuários.

Contribuição: A principal contribuição desta proposta é uma arquitetura de middleware para compartilhar documentos sigilosos por meio de nuvens públicas obtendo-se as principais garantias de segurança necessárias no gerenciamento de chaves criptográficas, fazendo com que a integração desses serviços fique transparente para uma aplicação do usuário que necessita utilizar esses serviços. A arquitetura proposta torna prática a manutenção de usuários e grupos utilizando Criptografia Baseada em Identidade com multi autoridades, segredo compartilhado e *erasure codes*. Este trabalho envolve temas de pesquisa como revogação segura de usuários, correto gerenciamento da privacidade dos documentos para grupos de usuários, chaves privadas geradas por demanda, segurança física e lógica utilizando módulos de segurança criptográficos 3.3, tolerância a falhas e um compartilhamento eficiente de documentos.

Este artigo está organizado da seguinte forma. Na seção 2 são apresentados a fundamentação matemática e os trabalhos relacionados. Na seção 3 é apresentada a arquitetura da solução proposta. A seção 4 apresenta algoritmos detalhados das principais funções do middleware. Na seção 5 são feitas avaliações a respeito da segurança e usabilidade da arquitetura proposta. Por fim, na seção 6 são apresentadas as conclusões do estudo.

 ©2014 SBC — Soc. Bras. de Computação

2. Fundamentação Matemática e Trabalhos Relacionados

2.1. Geração de Chaves Distribuída

Neste trabalho utiliza-se um mecanismo distribuído de geração de chaves mestras de CBI baseado no protocolo de Joint-Feldman Distributed Key Generator (JF-DKG), o protocolo de geração distribuído modificado e proposto por Aniket [Kate et al. 2012]. O protocolo JF-DKG, o mais simples e eficiente dos geradores distribuídos de chaves, requer um número $n \geq 3t + 1$ de nodos para funcionar corretamente, sendo t a quantidade de nodos que podem falhar. Este trabalho utiliza a técnica de BF-IBE [Boneh and Franklin 2001] devido a sua simplicidade nos protocolos de inicialização e métodos de criptografia. Na inicialização, um Gerador de Chaves Privadas (GCP) gera as chaves privadas (d) dos usuários utilizando suas identidades (ID) e uma chave mestra (s). Este trabalho visa uma geração de chave distribuída (n, t) por meio de um grupo de curvas elípticas G com uma ordem q e um gerador U, onde n é o número total de nodos envolvidos e $t + 1$ nodos honestos são suficientes para gerar corretamente o segredo. Seja F(z) = $a_0 + a_1 z + ... + a_t z^t \in Z_q[z]$ o atual polinômio compartilhado e s = a_0, sendo s o segredo.

O protocolo proposto por Aniket utiliza uma versão melhorada do protocolo de Feldman Verifiable Secret Sharing (Feldman VSS) para gerar de forma distribuída a chave mestre. Este possui um "quadro de avisos" que gera os parâmetros públicos da inicialização do BF-IBE, publica os valores e inicializa os valores de A_k e A_{ik} em zero, para i = 1, ..., n e k = 0 ... t, onde $A_k = a_k U$ e $A_{ik} = a_{ik} U$. A chave mestra é inicializada em zero. Os nodos inicializam o protocolo Feldman VSS. Depois que $t + 1$ nodos terminarem com sucesso o protocolo, as partes são consideradas seguras. Estes nodos são então chamados de nodos qualificados. Aqui, estes nodos são denotados como ∂. O quadro de avisos então computa e transmite os coeficientes A_k (para k = 0... t) para os polinômios compartilhados F(z).U como $A_k = \sum_{P_j \in \partial} A_{ik}$. Depois de verificar os A_k valores, os nodos enviam assinaturas de confirmações para o quadro de avisos. Depois de receber $t + 1$ confirmações, o valor A_k é finalizado. Cada nodo então computa sua parte do segredo como $s_i = \sum_{P_j \in \partial} s_{ji}$.

2.2. Extração da Chave Privada

Para extrair a chave privada de forma distribuída, o usuário deve entrar em contato com os nodos e enviar um ID específico. Após receber o ID, autenticar e autorizar o usuário, os GCP $P_i \in O$ retornam uma parte da chave privada $S_i H(ID)$ por meio de um canal seguro. O símbolo H representa uma função de resumo criptográfico $H : (0, 1)^* \rightarrow G^*$. Depois de receber $t+1$ partes corretas da chave privada, o usuário pode reconstruir a chave privada da seguinte maneira: $D_{id} = \sum_{P_i \in O} \lambda_i s_i H(ID)$, onde o coeficiente de Lagrange é $\lambda_i = \prod_{P_j \in O, j \neq i} \frac{j}{j-i}$.

2.3. Trabalhos Relacionados

Trabalhos recentes ([Itani et al. 2009], [Pearson et al. 2009]) propõem o uso de serviços de privacidade para resolver o problema do armazenamento de documentos sensíveis, assim como outros trabalhos ([Padilha and Pedone 2011], [Singh et al. 2011]) que propõem não cifrar os arquivos, apenas quebrá-los e enviar para diferentes provedores de nuvem. Estes trabalhos tentam contornar os problemas envolvidos no armazenamento de documentos sensíveis na nuvem, entretanto, não conseguem prover as propriedades necessárias

 ©2014 SBC — Soc. Bras. de Computação

para garantir um compartilhamento seguro. O trabalho de Itani et al. ([Itani et al. 2009]) não provê um esquema de compartilhamento tolerante a falhas. Caso o serviço de privacidade seja interrompido, o cliente não pode cifrar ou decifrar arquivos. A proposta de Padilha e Pedone ([Padilha and Pedone 2011]) usa a técnica de homomorfismo para modificar as partes dos arquivos cifrados por meio de funções aditivas, contudo, as técnicas para prover sigilo utilizando homomorfismo total são teóricas e não existem implementações pragmáticas. Implementações práticas do homomorfismo total para sistemas de sigilo ainda são assuntos abertas de pesquisa, portanto, não são aplicáveis nas atuais circunstâncias das corporações.

Outra linha de trabalho é a Criptografia Baseada em Atributos (CBA), onde cada usuário recebe credenciais de autoridades confiáveis, liberando então o acesso aos dados sensíveis. Alguns trabalhos ([Ruj et al. 2011], [Jung et al. 2013], [Yang et al. 2012a] e [Yang et al. 2012b]) propõem o uso de multi autoridades para gerar as chaves privadas, evitando assim que as autoridades distribuidoras de chaves possuam controle das chaves privadas (*key escrow*). Esses trabalhos também funcionam de maneira distribuída para fornecer os atributos, conseguindo assim suprir as necessidades da confidencialidade das identidades. No entanto, as soluções que envolvem CBA possuem a desvantagem da revogação. Uma vez que uma chave é revogada, o sistema necessita recifrar todos os documentos sensíveis e, desta forma, gerar novas chaves para todos os usuários. Outra peculiaridade em quase todos os trabalhos é que a tolerância a falhas não é considerada. As propostas são baseada em multi autoridades apenas para a extração das chaves privadas e atributos, não obstante, armazenam os arquivos cifrados em apenas um provedor de nuvem. Caso este provedor de nuvem falhe por qualquer razão, o usuário não conseguirá ter acesso a seus dados sensíveis. Outro problema devido a este fato é no caso do comprometimento de uma chave, se o atacante tiver algum tipo de acesso ao provedor de nuvem, ele pode obter o dado integral do documento, pois ele estará cifrado e completo em apenas um local.

O sistema DepSky, de Bessani et al [Bessani et al. 2011], usa conceitos que foram utilizados neste trabalho: criptografia simétrica, segredo compartilhado de Shamir e *erasure optimal code*. Contudo, o artigo não propõe mecanismos necessários para compartilhar as chaves que garantem a integridade das partes dos arquivos sigilosos. O trabalho simplesmente admite que existe um mecanismo para compartilhamento, no entanto, este é exatamente um dos principais desafios para a garantia do sigilo na nuvem utilizando criptografia. Outra fragilidade no artigo é o algoritmo de leitura de um dado que não verifica a integridade do resumo criptográfico das partes com a chave pública. Caso o provedor de nuvem seja malicioso, poderá modificar as partes e prover falsos resumos criptográficos no momento da verificação da integridade, comprometendo então o funcionamento do sistema.

O trabalho de Zhou et al [Zhou et al. 2011] utiliza a técnica de BF-IBE modificada para impor uma criptografia baseada em papéis, possibilitando cifrar um documento para um usuário específico ou para grupos de usuários com um determinado papel. A proposta é eficiente apenas quando não há muitas revogações, caso contrário, apresentará alta complexidade. O trabalho de Zhou não resolve o problema das custódias das chaves (*key escrow*). O administrador do sistema tem acesso às chaves privadas no método de extração. Outro problema é o ponto único de falha, fazendo com que caso o administrador

 ©2014 SBC — Soc. Bras. de Computação

seja comprometido, parte do sistema de revogação também seja comprometida.

Baseado em pesquisas anteriores, este artigo identificou os principais desafios para compartilhar documentos sensíveis de uma forma segura. Foi proposta uma arquitetura de middleware para tentar superar todos os desafios envolvendo o uso da CBI para prover as seguintes propriedades: revogação segura de usuários, geração de chaves por demanda, gerenciamento seguro dos grupos, tolerância a falhas, economia no armazenamento e compartilhamento eficiente de documentos.

3. Arquitetura do CloudSec

3.1. Premissas

Este trabalho tem como principal foco propor uma arquitetura de middleware para prover um compartilhamento seguro de documentos sigilosos:

- O controle de acesso é replicado em cada servidor de nuvem ou módulo de segurança criptográfico 3.3;
- Não existe concorrência para realizar alterações no controle de acesso;
- Os provedores de nuvem são semi-confiáveis (Estes irão se comportar corretamente perante as requisições dos usuários, mas são curiosos para ver os dados armazenados);
- Existe um versionamento dos documentos que são escritos na nuvem;
- O middleware será disponibilizado como uma biblioteca que deverá ser integrada a aplicações já existentes.

3.2. Arquitetura

A arquitetura possui três componentes principais: Os provedores de nuvem pública para armazenamento, usuários finais que desejam compartilhar documentos sigilosos e gerenciadores de chaves criptográficas. A figura 1 ilustra a arquitetura do middleware. Cada componente é melhor detalhado a seguir:

- **Gerenciadores de Chaves Criptográficas:** Estarão instalados internamente em diferentes MSCs em nuvens privadas, com ambientes controlados fisicamente e com acesso à Internet. Deve-se empregar uma distribuição geográfica, fazendo com que os MSCs possam ser acessados por meio da Internet mas que estejam geodistribuídos o suficiente para não serem comprometidos facilmente de forma física. Esse gerenciamento utilizará conceitos de criptografia baseada em identidade em conjunto com um rígido controle de acesso.
- **Provedor de Nuvem Pública:** Os servidores de armazenamento precisam estar hospedados em diferentes provedores de nuvem. As principais características necessárias são: a distribuição do controle de acesso por meio de replicação de estados e o funcionamento distribuído nos nodos dos servidores de aplicação. Os provedores de nuvem são responsáveis pelo armazenamento dos documentos e pelo controle de acesso aos dados.
- **Usuário:** O lado do usuário é responsável por editar, cifrar e decifrar arquivos. Os usuários possuem uma aplicação local que interage com o middleware para realizar as funções criptográficas e funções de compartilhamento dos arquivos sigilosos. Os mesmos definem quem serão os custodiantes dos documentos sensíveis cifrados. Esses custodiantes são delimitados por regras de acesso específicas de cada aplicação.

 ©2014 SBC — Soc. Bras. de Computação

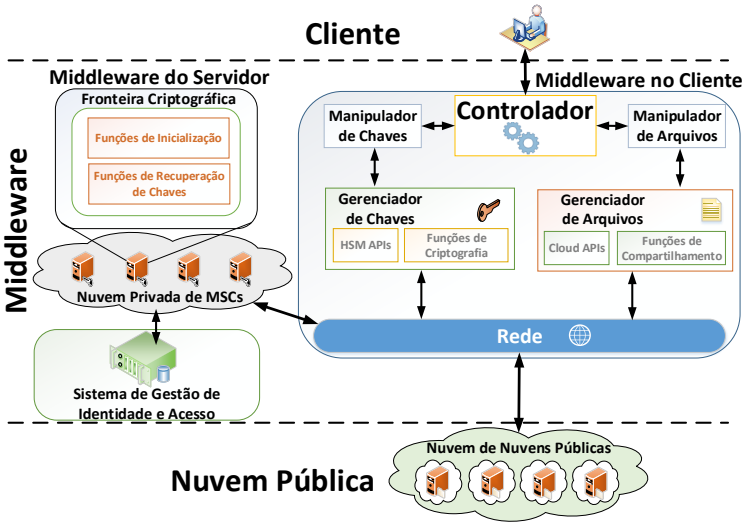

Figura 1. Arquitetura do Middleware de Privacidade em Nuvem.

A arquitetura do middleware possui diferentes módulos para a integração de serviços para aplicações que desejam proporcionar o sigilo no compartilhamento de documentos sensíveis. Os módulos são os seguintes:

- **Controlador:** Esse módulo será o responsável por disponibilizar uma interface de programação de aplicativos (*Application Programming Interface* - API) para o usuário, a fim de que seja feita uma integração entre os serviços de armazenamento em nuvem, gerenciamento de chaves nos módulos de segurança criptográficos e serviços de criptografia.
- **Manipulador de Chaves:** Módulo responsável pelas chamadas que envolvem criptografia, como a criação de chaves simétricas e a construção das chaves assimétricas baseada em identidade, assim como funções que envolve processos de transformação dos dados em claro para dados criptografados. Esse módulo também é responsável por criar localmente chaves simétricas para sigilo e chaves assimétricas baseadas em identidade por meio do módulo HSM API.
- **Manipulador de Arquivos:** Módulo responsável pelas chamadas que envolvem o armazenamento e recuperação de arquivos sigilosos e que estão cifrados e assinados pelo Manipulador de Chaves. Disponibilizará funções para separar e juntar arquivos e chaves criptográficas envolvidas no processo de sigilo. Disponibilizará também as diferentes APIs utilizadas por diferentes provedores de nuvens, com o objetivo de estabelecer a correta comunicação com todos para o compartilhamento de arquivos.

Essa arquitetura prevê que toda a infraestrutura de provedores de nuvens públicas para armazenamento e os módulos de segurança criptográficos estejam previamente configurados e com seus controles de acesso pré-definidos. O middleware ficará encarregado de fazer toda a integração entre os processos de autenticação, armazenamento e sigilo, fazendo com que todo o procedimento de compartilhamento de documentos sigilosos seja o mais seguro e transparente possível para o usuário.

A arquitetura utiliza internamente o esquema de segredo compartilhado para integrar a confidencialidade com a disponibilidade. Uma vez que todas as chaves simétricas

 ©2014 SBC — Soc. Bras. de Computação

cifradas são quebradas em N pedaços (sendo N o número total de provedores de nuvem), o usuário necessitará de um número mínimo de M partes (M é um número pré-estabelecido no momento da inicialização do sistema) para reconstruir a chave cifrada. De fato, este trabalho reutiliza o controle de acesso da aplicação para controlar quais leitores estarão habilitados para acessar o dado armazenado. Este trabalho também utiliza o mecanismo de *information-optimal erasure code* [Plank et al. 2008], possibilitando uma economia nos provedores de nuvem ao armazenar os arquivos de tal forma que cada parte é reduzida por um fator de $\frac{n}{f+1}$, considerando f o número de servidores com falhas.

Para garantir propriedades de segurança no gerenciamento das chaves criptográficas, utiliza-se o esquema BF-IBE para cifrar chaves simétrica construindo um identificador ID específico contendo as seguintes informações: Nome do Documento, Grupo de Custodiantes e Versão do Documento. Este identificador específico ID é utilizado devido a uma série de características que são necessárias para compartilhar documentos sensíveis. O Nome do Documento no ID é utilizado para que cada documento cifrado e armazenado na nuvem possua uma chave diferente. O Grupo de Custodiantes é para limitar o controle de acesso ao documento e como estaremos reutilizando o controle de acesso do sistema, este grupo de custodiantes será utilizado para autenticar e liberar o acesso aos documentos sigilosos. Para cada grupo diferente de custodiantes existirá uma chave diferente. O Número de Versão juntamente com os outros elementos são utilizados para controlar o acesso de diferentes versões dos documentos e, desta forma, garantir a segurança na entrada e saída de membros do grupo. A técnica de Assinatura Hess [Hess 2003] é utilizada para assinar as partes cifradas das chaves e documentos para garantir a integridade. Neste trabalho é utilizado o mesmo par de chaves para cifrar e assinar, facilitando assim o gerenciamento e compartilhamento das chaves.

3.3. Módulo de Segurança Criptográfico

O módulo de segurança criptográfico (MSC) [2] tem como principal função o correto gerenciamento das chaves criptográficas, protegendo-as de forma física e lógica. Para isso, os MSCs serão previamente configurados com um firmware contendo os aplicativos necessários para o seu correto funcionamento, incluindo o aplicativo gerenciador de chaves criptográficas. Com esse firmware, o MSC disponibilizará o aplicativo de gerenciamento das chaves privadas baseadas em identidade utilizando APIs próprias para comunicação com os usuários. Essa API deve ser simples o suficiente para não prejudicar a segurança do MSC, mas deve prover todas as funções necessárias para o correto gerenciamento das chaves criptográficas.

Os MSCs devem inicialmente ser configurados para compartilharem partes de uma chave mestra. Para isso, o aplicativo responsável nos MSCs deverá executar os passos citados na seção 2.1. Com isso, todos os MSCs envolvidos no gerenciamento das chaves criptográficas baseadas em identidade compartilharão uma parte da chave mestra que gerará as chaves privadas dos usuários do middleware.

3.4. Implementação

Com o CloudSec, pode-se compartilhar e recuperar dados sigilosos entre grupos de usuários utilizando nuvens híbridas por meio de funcionalidades específicas. Essas

[2] Exemplo de um MSC Nacional: ASI-HSM `http://www.kryptus.com/#!asi-hsm/c11e6`

 ©2014 SBC — Soc. Bras. de Computação

funções tem como principal objetivo prover ao usuário funções reutilizáveis e configuráveis para o desenvolvimento mais rápido de sistemas que necessitem de compartilhamento de documentos sigilosos. A figura 2 ilustra o diagrama com as principais classes referentes a essa arquitetura. As classes *Storage Handler* e *Crypto Handler* são as principais classes de controle do middleware e distribuem as requisições conforme forem solicitadas pela classe *Controller*.

Figura 2. Diagrama de Classe ilustrando os principais componentes do CloudSec.

Para a realização dos procedimentos de compartilhamento de documentos sigilosos, o CloudSec provê para as aplicações as funcionalidades de armazenamento e criptografia por meio das seguintes APIs:

- **searchForMetadata**$(dataId)$**:** Para o compartilhamento ou recuperação de arquivos sigilosos, deve-se procurar primeiramente pelos metadados de um determinado arquivo para verificar suas informações. Para isso, deve-se especificar o nome do arquivo que deseja-se compartilhar. Caso o documento exista nos provedores de nuvens públicas, os metadados serão retornados contendo informações a respeito dos grupos que possuem acesso, assim como as versões existentes no sistema de armazenamento. Caso o documento não exista, é retornado ao usuário um metadado nulo.
- **sharePrivateData**$(dataPath, Metadata)$**:** Para que um usuário do sistema possa compartilhar um arquivo, deve-se passar o caminho do mesmo ($dataPath$), assim como um metadado ($Metadata$) contendo informações desse arquivo, como nome, versão e grupo que poderá ter acesso ao mesmo. Antes de realizar essa operação, deve-se consultar os metadados relacionados ao arquivo que deseja-se compartilhar para verificar se existem e quais as informações que já estão disponíveis nos provedores de nuvens públicas.
- **recoverPrivateData(**$Metadata$**):** Para recuperar um determinado arquivo, deve-se especificar o metadado ($Metadata$) do arquivo desejado. Assim como no método de

 ©2014 SBC — Soc. Bras. de Computação

compartilhar, deve-se primeiramente tentar obter os metadados do arquivo que deseja-se recuperar para verificar quais as especificações do arquivo que deseja-se obter.

A implementação foi feita em módulos e serve como prova de conceito para validar as ideias aqui propostas, sendo assim, podem ser otimizadas para alcançar melhores resultados. Foram implementadas apenas as operações criptográficas para salvar e ler em disco. Portanto, não existe nenhum tipo de comunicação com servidores de nuvem para autenticação, envio ou recebimento de arquivos. Para a criptografia simétrica foi utilizado o algoritmo AES para cifrar os arquivos com um tamanho de chave de 128 bits. Para o resumo criptográfico foi utilizado o algoritmo SHA-1. NA cifragem das chaves simétricas foi utilizada a técnica de BF-IBE. Para assinar as partes cifradas foi utilizada a técnica de assinatura de Hess. Para os testes foram utilizados um total de quatro nodos, contando com um número de nodos redundantes igual a dois (este número está diretamente ligado ao processo de inicialização dos Distribuidores de Chaves Privadas (DCPs), do qual requerem um número mínimo de $3f + 1$, sendo $f + 1$ o quorum para recuperar a chave mestra), sendo f o número máximo de falhas toleradas no sistema.

Os protocolos foram implementados em C e C++. A implementação foi dividida em três partes: A inicialização dos DCP, o compartilhamento distribuído de documentos e os algoritmos de Leitura e Escrita. A inicialização dos DCPs foi implementada por Aniket em seu trabalho utilizando C++ e o protocolo modificado de JF-DKG. O trabalho de Aniket utiliza a biblioteca de emparelhamento bilinear PBC (*Pairing Based Cryptography*) [Lynn 2013]. As comunicações necessárias entre os clientes e servidores foram implementadas utilizando *sockets* e para a comunicação segura foi utilizada a biblioteca *OpenSSL*. Os algoritmos de Leitura e Escrita foram implementados em C/C++ utilizando as seguintes bibliotecas: *pbc library* [Lynn 2013] para realizar as operações de emparelhamento bilinear, a biblioteca *gfshare* para realizar as operações de segredo compartilhado de Shamir, a biblioteca *jerasure* para codificar e decodificar utilizando o *information-optimal erasure code* [Plank et al. 2008] e a biblioteca *OpenSSL* para realizar algumas das operações usuais de criptografia.

4. Algoritmos

O algoritmo 1 (Compartilhar Dado) autentica o usuário e solicita para o controle de acesso os metadados do documento (linha 5). O novo documento a ser armazenado terá a última versão encontrada (linha 6) mais um (linha 7). Uma chave simétrica é aleatoriamente gerada (linha 8) para cifrar o documento (linha 10). Um identificador ID para o documento é definido (linha 11) e uma chave pública baseada neste ID é criada (linha 12) utilizando a esquema de BF-IBE. A chave simétrica é então cifrada com a chave pública (linha 13) e então quebrada em pedaços utilizado o segredo compartilhado de Shamir (linha 14). O documento cifrado é então codificado utilizando-se o algoritmo de *information-optimal erasure code* (linha 15), reduzindo o tamanho dos dados que serão armazenados nos provedores de nuvem. A chave privada é então criada baseada no identificador ID (linha 17) de acordo com o procedimento da seção 2.2. Para cada parte das chaves e documentos cifrados, serão fornecidos os resumos criptográficos (linhas 19 e 20) e estes serão assinados (linhas 21 e 22) utilizando-se o esquema de assinatura Hess. Um bloco de dados é então criado, reunindo toda a informação necessária para armazenar o documento (linha 23). O bloco de dados é então enviado para os provedores de nuvem (linha 24) e a entrada deste armazenamento é enviado para o controle de acesso (linha 27).

 ©2014 SBC — Soc. Bras. de Computação

Algoritmo 1 COMPARTILHARDADO$(nomeArquivo, grupoArquivo, usuario, senha)$

1: total $\longleftarrow n$
2: redundante $\longleftarrow m$
3: dado_ver $\longleftarrow 0$
4: token $\longleftarrow autenticar(usuario, senha)$
5: mt $\longleftarrow buscarMetadados(nomeArquivo, grupoArquivo, usuario, token)$
6: dado_ver $\longleftarrow max(mt[i].ver : 0 \leq i \leq n-1)$
7: dado_ver_novo $\longleftarrow dado_ver + 1$
8: ks $\longleftarrow gerarChaveSim()$
9: dado $\longleftarrow abrirArquivo(nomeArquivo)$
10: e_dado $\longleftarrow cifrar(dado, ks)$
11: id $\longleftarrow nomeArquivo + "/" + grupoArquivo + "/" + dado_ver_novo$
12: pubk_id $\longleftarrow gerar_chave_pub(id)$
13: e_ks $\longleftarrow cifrar(ks, pubk_id)$
14: enc_ks[0 .. n-1] $\longleftarrow partir(e_ks, total - redundante, total)$
15: enc_dado[0 .. n-1] $\longleftarrow codificar(e_dado, total - redundante, redundante)$
16: $i \longleftarrow 0$
17: privk_id $\longleftarrow gerar_chave_priv(id)$
18: **for** $(0 \leq i \leq total - 1)$ **do**
19: dado_hash $\longleftarrow H(enc_data[i])$
20: ks_hash $\longleftarrow H(enc_ks[i])$
21: dado_hash_assinado $\longleftarrow assinar(dado_hash, privk_id)$
22: ks_hash_assinado $\longleftarrow assinar(ks_hash, privk_id)$
23: blocoDeDado $\longleftarrow (id, enc_ks[i], enc_dado[i], dado_hash_assinado, ks_hash_assinado)$
24: ack $\longleftarrow mensagemDeEnviarDados(cloud_i, blocoDeDado, token)$
25: **if** $(ack =' ok')$ **then**
26: controleBlocoDeDado $\longleftarrow (id, usuario, grupoArquivo)$
27: $enviarMsgControleDeAcesso(cloud_i, controleBlocoDeDado)$
28: **end if**
29: $i \longleftarrow i + 1$
30: **end for**

O algoritmo 2 (Recuperar Dado) primeiramente autentica o usuário e busca no controle de acesso pelos metadados do documento (linha 4). A última versão é então escolhida (linha 5) e o identificador ID é composto (linha 6). Um par de chaves é gerado a partir do ID utilizando o esquema BF-IBE (linhas 7 e 8) e a requisição pelos dados é iniciada (linha 11). Os blocos de dados são requeridos dos provedores de nuvem (linha 12), onde cada bloco de dado é verificado a partir das suas assinaturas e resumos criptográficos utilizando o esquema de assinatura Hess (linha 15). Após obter o número mínimo de blocos de dados (linha 23), as partes do documento cifrado são então decodificadas (linha 31), a chave simétrica é recomposta utilizando o segredo compartilhado de Shamir (linha 32), decifrada (linha 33) e finalmente, o documento original é decifrado (linha 34) e retornado para a aplicação.

5. Avaliação

5.1. Avaliação da Arquitetura

Baseado na pesquisa realizada, pode-se estabelecer os principais requisitos para elaborar uma arquitetura de middleware para garantir o compartilhamento sigiloso de documentos. Os requisitos necessários para o gerenciamento das chaves criptográficas e as soluções encontradas por este middleware proposto foram:

Custódia das Chaves: Apenas o usuário deverá ter posse da sua chave, evitando assim que as entidades distribuidoras de chaves possam ter posse das mesmas. Um dos principais problemas apontado pelo documento SP800-144 [Jansen and Grance 2011] é a vulnerabilidade de ataques internos e a falta de suporte legal em casos de intrusão devido a localização geográfica dos servidores. Para resolver este problema, este artigo propõe o uso de multi autoridades de CBI embarcados em módulos de segurança criptográficos (MSCs) utilizando o protocolo modificado de JF-DKG para

 ©2014 SBC — Soc. Bras. de Computação

Algoritmo 2 RECUPERARDADO$(nomeArquivo, grupoArquivo, usuario, senha)$

1: total $\longleftarrow n$
2: redundante $\longleftarrow m$
3: token $\longleftarrow autenticar(usuario, senha)$
4: mt $\longleftarrow buscarMetadados(nomeArquivo, grupoArquivo, usuario, token)$
5: dado_ver $\longleftarrow max(mt[i].ver : 0 \leq i \leq n-1)$
6: id $\longleftarrow nomeArquivo + "/" + grupoArquivo + "/" + dado_ver$
7: privk_id $\longleftarrow gerar_chave_privada(id)$
8: pubk_id $\longleftarrow gerar_chave_pub(id)$
9: $i \longleftarrow 0$
10: $ERRO \longleftarrow 0$
11: **while** $(i \leq n-1)$ **do**
12: t_b $\longleftarrow cloud_i.buscarBlocoDeDado(id, token)$
13: t_eks $\longleftarrow t_b.retorna_enc_ks()$
14: t_edado $\longleftarrow t_b.retorna_enc_data()$
15: rt $\longleftarrow verifica(t_b.ks_hash_assinado_i, t_b.data_hash_assinado_i, t_eks, t_edado, pubk_id)$
16: **if** $(rt = true)$ **then**
17: $enc_ks[i] \longleftarrow t_eks$
18: $enc_dado[i] \longleftarrow t_edado$
19: **else**
20: $ERRO \longleftarrow ERRO + 1$
21: **end if**
22: $i \longleftarrow i + 1$
23: **if** $(i > redundante - 1)$ **then**
24: $Break$
25: **else**
26: **if** $(ERRO > redundante - 1)$ **then**
27: **retorna** $ERRO$
28: **end if**
29: **end if**
30: **end while**
31: e_dado $\longleftarrow decodificar(enc_dado, total - redundante, redundante)$
32: e_ks $\longleftarrow combinar(enc_ks, total - dedundante, total)$
33: ks $\longleftarrow decifrar(e_ks, privk_id)$
34: **retorna.** Decifrar(e_dado, ks)

gerar a chave mestra sem que nenhuma das autoridades tenha controle total da mesma. Utilizando este esquema, dois parâmetros são atribuídos: t e N, onde N é o número total de autoridades e t representa o número mínimo de partes que serão necessárias para recuperar a chave privada. Ao embarcar as autoridades em MSCs, consegue-se obter níveis altos de segurança física e lógica devido à natureza dos mesmos. Com isso, diminui-se a probabilidade de um ataque às autoridades e exposição das chaves privadas. Mas mesmo em um caso excepcional de sucesso de um ataque, um agente malicioso precisa descobrir um total de t partes para recompor o segredo, reduzindo assim as chances de sucesso de um ataque. Cada MSC terá posse de apenas uma parte do segredo mestre s_i e com isso, mesmo que obtenham o identificador que será utilizado, terão acesso apenas a uma parte $S_iH(ID)$ do segredo do usuário. Para a reconstrução total da chave, o atacante deverá possuir um número mínimo t. Sem este número mínimo, o atacante não conseguirá executar a equação $D_{id} = \sum_{P_i \in O} \lambda_i s_i H(ID)$ para obter a chave privada do usuário.

Revogação do Acesso: Após a retirada de um usuário de um grupo, o mesmo não deve mais ter acesso aos documentos cifrados e não poderá decifrar novos documentos que foram compartilhados. Para manter a confidencialidade da informação e evitar problemas com a revogação das chaves privadas, é recomendado utilizar parâmetros adicionais para identificar a chave pública da Criptografia Baseada em Identidade (CBI). Uma parte da solução é não vincular uma chave por usuário, mas vincular grupos de usuários com documentos, tendo assim chaves semânticas. Este trabalho propõe o uso de identificadores contendo regras de acesso concatenado com o nome do documento e a versão do mesmo. Estas regras são verificadas pelos DCPs por meio do controle de acesso que deve ser realizado de maneira distribuída e de uma forma confiável. O nome do documento vincula uma chave pública a um documento específico. A versão faz com

 ©2014 SBC — Soc. Bras. de Computação

que a cada modificação do documento, seja gerado um novo par de chaves. Essas propriedades são garantidas pelo uso de um algoritmo de resumo criptográfico na geração dos identificadores e chaves, onde ID será a composição destas regras e a chave privada será definida por: $D_id = sH(ID)$. Portanto, um membro que faz parte de um grupo e obteve acesso a chave privada para decifrar um documento em uma versão X, não conseguirá obter uma chave consequente para decifrar um documento na versão $X + 1$ caso não faça mais parte deste grupo. Se um usuário já obteve a chave privada, este em algum momento teve acesso a um documento em uma versão específica. Desta forma, não existe a necessidade de recifrar o conteúdo do documento que já foi exposto. Caso um usuário não tenha ainda obtido a chave privada e já não pertence mais a um determinado grupo, este não mais terá acesso às partes das chaves e documentos cifrados, pois o controle de acesso não permitirá mais o acesso devido ao não cumprimento das regras pré-estabelecidas para aquele documento. Para mais detalhes a respeito do controle de acesso, o trabalho de dissertação de Rick [de Souza 2014] pode ser consultado.

Tolerância a Falhas: O sistema deve fornecer dois níveis de tolerância a falha. Tanto para o gerenciamento de chaves quanto para o armazenamento de documentos cifrados. Caso um servidor fique indisponível, deve-se prover mecanismos de contingência dos servidores para manter a solução em funcionamento. Utilizando DCPs distribuídos, segredo compartilhado e *erasure code*, consegue-se garantir tolerância a falhas. Este trabalho propõe protocolos baseados no quorum secreto compartilhado e verificável, aumentando assim o rigor na verificação das partes distribuídas. A tolerância a falhas bizantinas e a disponibilidade são garantidas com a propriedade de que em um total de $3f + 1$ DCPs, apenas f podem falhar e consequentemente, são necessários $2f + 1$ servidores para garantir o correto funcionamento do sistema, garantindo assim a disponibilidade e confiança. Desta forma, mesmo que f DCPs falhem, ao obter $f + 1$ respostas do restante dos servidores, pode-se reconstruir a chave privada do usuário utilizando-se do método de extração de chaves privadas mencionado na sub-sessão 2.2.

5.2. Análise de Performance

Com a implementação pode-se avaliar o desempenho dos algoritmos Compartilhar Dado e Recuperar Dado, mencionados na seção 4. Foram feitas sequências de execuções utilizando diferentes tamanhos de arquivos. Os testes partiram de arquivos com tamanho de 1 Kbyte até 524288 Kbytes (512 MBytes). Os testes foram executados em um computador com as seguintes características: Processador Intel i3, 4GB RAM com um sistema operacional Linux Ubuntu. Os testes foram executados durante uma semana e cada algoritmo avaliado conforme o desvio padrão. Entre 1 Kbyte e 16384 Kbytes o tempo foi instável, superando 5% de desvio padrão; isto se deve ao pequeno tamanho dos arquivos. Para arquivos de tamanho entre 1 Kbyte e 16384 Kbytes os tempos variaram até um valor máximo de 300 milissegundos. A partir de 16384 Kbytes, o tempo começou a crescer linearmente dobrando conforme o tamanho do arquivo. A figura 3 ilustra o comportamento conforme o aumento do tamanho dos arquivos entre 1 Kbyte e 8 Mbytes e a figura 4 ilustra o desempenho dos arquivos entre 16 Mbytes e 512 Mbytes.

6. Conclusão

Com a arquitetura proposta atingiu-se os objetivos propostos inicialmente de criar um middleware cujas funcionalidades integrassem os serviços de gerenciamento seguro de

 ©2014 SBC — Soc. Bras. de Computação

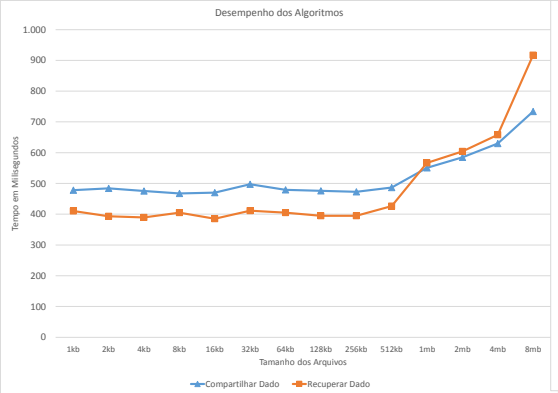

Figura 3. Desempenho para Compartilhar e Recuperar Dado com arquivos entre os tamanhos de 1Kbyte e 8Mbytes.

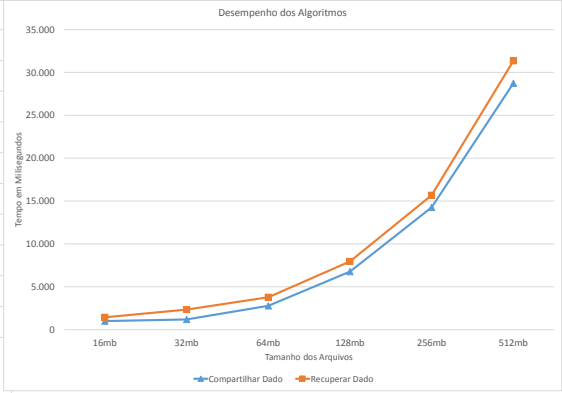

Figura 4. Desempenho para Recuperar e Compartilhar dado para arquivos entre os tamanhos de 16Mbytes e 512Mbytes.

chaves criptográficas juntamente com o compartilhamento de documentos em diferentes provedores de nuvem pública. Para o gerenciamento das chaves utilizou-se módulos de segurança criptográficos distribuídos, fazendo com que os mesmos possuam segurança física e lógica, garantindo a integridade das chaves mestras utilizadas para gerar as chaves privadas dos usuários. Para a segurança e tolerância a falhas no armazenamento das partes dos documentos, a arquitetura utiliza diferentes provedores de nuvem pública para armazenamento. Dessa forma, ao utilizar uma nuvem híbrida, consegue-se garantir o nível de segurança necessário para o compartilhamento seguro de documentos sigilosos.

A garantia de segurança em caso de saída e entrada de membros dos grupos foi garantida por meio do gerenciamento de grupos juntamente com o nome utilizado na geração das chaves criptográficas baseadas em identidade. Ao utilizar um identificador que é composto por nome do documento, grupo custodiante e versão, consegue-se alcançar um nível de unicidade suficiente para a garantia de segurança nos casos de saída e entrada de um novo membro. Dessa forma, o controle de acesso juntamente com as propriedades de segurança da criptografia baseada em identidade garantem a segurança na revogação das chaves privadas.

O uso de um conjunto de geradores de chaves privadas distribuídos garante segurança contra das chaves privadas e também garante tolerância a falhas. Dessa forma, ao estabelecer um procedimento de inicialização dos geradores de chaves privadas (n, t), consegue-se estabelecer um número mínimo t de partes necessárias para recompor as chaves privadas, tal que $n \geq t$, de tal forma que apenas o usuário que se autenticar de maneira correta em t gerenciadores de chaves possa recuperar sua chave. Isso também garante a tolerância a falhas, devido ao fato de que caso o sistema feito seja $n = 4$ e $t = 3$, mesmo com um servidor comprometido, pode-se conseguir recompor a chave privada utilizando os outros três servidores.

Referências

Bessani, A., Correia, M., Quaresma, B., André, F., and Sousa, P. (2011). Depsky: dependable and secure storage in a cloud-of-clouds. In *Proceedings of the sixth conference*

 ©2014 SBC — Soc. Bras. de Computação

on Computer systems, pages 31–46. ACM.

Boneh, D. and Franklin, M. (2001). Identity-based encryption from the weil pairing. In *Advances in Cryptology - CRYPTO 2001*, pages 213–229. Springer.

de Souza, R. L. (2014). Um Middleware para Compartilhamento de Documentos Sigilosos em Nuvens Computacionais. Master's thesis, Departamento de Informática e Estatística, Universidade Federal de Santa Catarina.

Hess, F. (2003). Efficient identity based signature schemes based on pairings. In *Selected Areas in Cryptography*, pages 310–324. Springer.

Itani, W., Kayssi, A., and Chehab, A. (2009). Privacy as a service: Privacy-aware data storage and processing in cloud computing architectures. In *International Conference on Dependable, Autonomic and Secure Computing*, pages 711–716. IEEE.

Jansen, W. and Grance, T. (2011). Guidelines on security and privacy in public cloud computing. *NIST special publication*, pages 800–144.

Jung, T., Li, X.-Y., Wan, Z., and Wan, M. (2013). Privacy preserving cloud data access with multi-authorities. In *IEEE INFOCOM*.

Kate, A., Huang, Y., and Goldberg, I. (2012). Distributed key generation in the wild. *IACR Cryptology ePrint Archive*, 2012:377.

Lynn, B. (Novembro, 2013). The pairing-based cryptography (pbc) library. *Available on http://crypto. stanford. edu/pbc*.

Padilha, R. and Pedone, F. (2011). Belisarius: Bft storage with confidentiality. In *Network Computing and Applications (NCA), 2011 10th IEEE International Symposium on*, pages 9–16. IEEE.

Pearson, S., Shen, Y., and Mowbray, M. (2009). A privacy manager for cloud computing. In *Cloud Computing*, pages 90–106. Springer.

Plank, J. S., Simmerman, S., and Schuman, C. D. (2008). Jerasure: A library in c/c++ facilitating erasure coding for storage applications-version 1.2. *University of Tennessee, Tech. Rep. CS-08-627*, 23.

Ruj, S., Nayak, A., and Stojmenovic, I. (2011). Dacc: Distributed access control in clouds. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pages 91–98. IEEE.

Singh, Y., Kandah, F., and Zhang, W. (2011). A secured cost-effective multi-cloud storage in cloud computing. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 619–624. IEEE.

Yang, K., Jia, X., and Ren, K. (2012a). Dac-macs: Effective data access control for multi-authority cloud storage systems. *IACR Cryptology ePrint Archive*, 2012:419.

Yang, K., Liu, Z., Cao, Z., Jia, X., Wong, D. S., and Ren, K. (2012b). Taac: Temporal attribute-based access control for multi-authority cloud storage systems. *IACR Cryptology ePrint Archive*, 2012:651.

Zhou, L., Varadharajan, V., and Hitchens, M. (2011). Enforcing role-based access control for secure data storage in the cloud. *The Computer Journal*, 54(10):1675–1687.

 ©2014 SBC — Soc. Bras. de Computação