Armazenamento Seguro de Credenciais e Atributos de Usuários em Federação de *Clouds*

Luciano Barreto[1], Leomar Scheunemann[1] , Joni da Silva Fraga[1], Frank Siqueira[2]

[1]Departamento de Automação e Sistemas – (DAS)
[2]Departamento de Informática e Estatística – (INE)
Universidade Federal de Santa Catarina

{luciano.barreto,leomar.scheunemann}@posgrad.ufsc.br,{joni.fraga,frank.siqueira@ufsc.br}

Abstract. *The use of cloud computing and cloud federations has been the focus of studies. Many of these infrastructures delegate their authentication to Identity Providers. Once these services are available through the Internet, concerns about the confidentiality of user credentials and attributes are high. The main focus of this work is the security of the credentials and user attributes in authentication infrastructure, exploring secrets sharing techniques and using clouds federation as a base for the storage of this information.*

Resumo. *O uso de provedores de cloud e formações de federações tem sido foco de estudos há alguns anos. Muitas destas infraestruturas delegam as autenticações de seus usuários a Provedores de Identidades. Por se tratarem de serviços disponíveis através da Internet, a preocupação com o sigilo das credenciais e atributos de usuários é constante. O foco principal deste trabalho é a segurança das credenciais e atributos de usuários em infraestruturas de autenticação, explorando técnicas de compartilhamento de segredos e utilizando federações de clouds como base para o armazenamento destas informações.*

1. Introdução

O agrupamento de provedores de serviços de *cloud* em associações tem sido apresentado na literatura como solução para o compartilhamento de recursos ociosos. Vários termos têm sido usados para caracterizar estas associações de provedores de *clouds*: federação de *clouds*, *Inter-cloud* [Grozev and Buyya 2012], *Multi-cloud* [Grozev and Buyya 2012] e *Cross-Cloud* [Celesti et al. 2010]. Embora as diferenças de nomenclatura e mesmo que muitas das propostas para a formação destas redes de confiança entre *clouds* tenham diferentes focos, o objetivo final é sempre o uso de serviços de diversos provedores de *clouds* para atender a demanda de clientes e os *SLAs* (*Service-Level Agreement*) acordados. Uma tendência em sistemas distribuídos de larga escala como *clouds,* federações de *clouds*, grades computacionais e redes colaborativas em geral é o uso de infraestruturas que realizem o gerenciamento de identidades [Jøsang et al. 2005]. Estas infraestruturas normalmente integram políticas e tecnologias, permitindo às diferentes organizações que participam destes grandes sistemas terem aplicações que ultrapassem seus domínios locais (domínios administrativos ou de políticas), perfeitamente credenciadas e autorizadas pelas diferentes políticas locais.

Vários modelos de gerenciamento de identidades são identificados na literatura (dentre os quais destacamos o centralizado, identidades federadas e *user-centric* [Jøsang

et al. 2005]) e, em muitos destes modelos, a autenticação não é mais realizada nos provedores de serviços (*SPs*), mas sim em autoridades independentes e confiáveis de autenticação, chamadas de Provedores de Identidades. Estes Provedores de Identidades (autoridades de autenticação) na verdade passaram a ser os pontos centrais na aplicação de políticas de segurança em sistemas distribuídos. Mas, também por serem serviços que ficam disponíveis via Internet, estes *IdPs* estão sujeitos a ataques que podem resultar em intrusões, o que seria catastrófico para a segurança das informações e recursos.

O objetivo deste artigo é apresentar uma abordagem que faz uso de provedores de *cloud* para armazenar as informações de usuário (credenciais e atributos) e vem como uma proposta de solução para a autenticação em federações de provedores de *cloud*, mas não restrita a somente estes ambientes. O uso de *clouds* no armazenamento destas informações de usuário exige um conjunto de mecanismos para fornecer segurança e disponibilidade destas informações, mesmo considerando a falta de controle absoluto sobre estes recursos e serviços delegados. Mas a vantagem desta abordagem está na flexibilidade do acesso às informações permitindo que usuários possam fazer os seus *logins* a partir de um *IdP* de uma federação de *clouds* e ter acessos a recursos em diferentes partes desta federação. O restante deste texto está organizado da seguinte forma: na seção 2 apresentamos o modelo de sistema, na seção 3 a arquitetura proposta. A seção 4 foca nos protocolos e os resultados da abordagem são apresentados na seção 5. Por fim, na seção 6 é apresentada a literatura relacionada e na seção 7 as conclusões do artigo.

2. Modelo de Sistema

2.1 Caracterização de Federação de *Clouds*

O agrupamento de provedores de *clouds* através de redes de confiança formadas com o objetivo de atender um vasto número de usuários coloca também grandes desafios na autenticação e autorização destes usuários junto aos provedores nestas federações. A Figura 1 ilustra uma destas associações que visam o compartilhamento de recursos. Nesta figura, explicitamos basicamente o aspecto dos controles de autenticação, centrando sempre estas funções em provedores de identidades (*IdPs*).

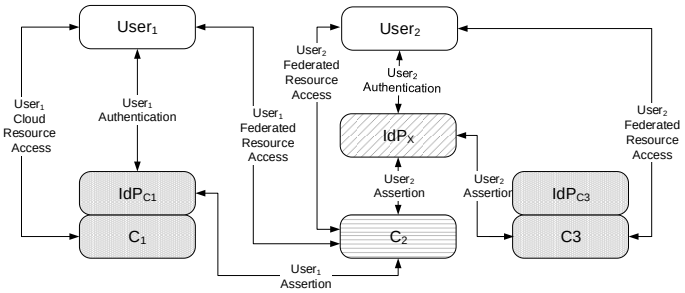

Figura 1 – Autenticação e Autorização em Federação de Clouds

Os provedores de *clouds* (*Clouds* C_1, C_2 e C_3) da Figura 1 apresentam formas diferentes em seus controles de autenticação. Os provedores C_1 e C_3 possuem *IdPs* internos ($IdPc_1$ e $IdPc_3$) para executar suas funções de autenticação e a *cloud* C_2, ao contrário, delega suas autenticações de usuários a um *IdP* externo (IdP_X) com o qual mantém relações de confiança. Considerando o arranjo citado, o usuário $User_1$ é mostrado efetuando a sua autenticação junto ao *IdP* de C_1 ($IdPc_1$). Este usuário pode só ter conta

na *cloud* $Cloud_{C1}$ e, diante disto, uma vez autenticado tem acesso a recursos do provedor C_1. Como uma federação de *clouds* é construída sobre relações de confiança, este mesmo usuário ($User_1$) tem também acesso em recursos de C_2, uma vez que esta última confia nas autenticações de C_1. Ou seja, os dois provedores de *cloud* (C_1 e C_2) mantêm relações de confiança. Nesta mesma figura, $User_2$ se autentica em IdP_X (um *IdP* externo que C_2 confia) e com isto acessa recursos de duas *clouds* da federação (C_2 e C_3).

2.2 Armazenamento de Credenciais e Atributos de Usuário

Nossa proposta está baseada na existência de diversos *IdPs*. Porém, diferentemente das abordagens convencionais, onde os atributos de usuários são armazenados localmente, em nossa proposta estes atributos não são armazenados diretamente nos *IdPs*, ou seja, as *clouds* não mantêm em seus *IdPs* as informações de seus próprios clientes. A ideia é dispor estas informações, de forma segura em uma base de dados construída sobre recursos distribuídos pela federação.

O armazenamento seguro em *clouds* tem como solução usual a encriptação dos dados, o que garante a confidencialidade dos mesmos. As chaves criptográficas usadas nestas encriptações são protegidas normalmente com o uso de técnicas de compartilhamento de segredos [Schoenmakers 1999; Shamir 1979]. As informações criptografadas podem ser replicadas e armazenadas em recursos de várias *clouds* de modo a garantir a disponibilidade das mesmas em ambientes onde podem ocorrer ações maliciosas ou mesmo falhas. Porém, existem soluções com um menor custo computacional que fazem uso de técnicas de dispersão de informação (*Information Dispersal Algorithms*: *IDA* [Rabin 1989]), baseadas em códigos de correção de erro (*erasure codes)*. Estas transformações *IDA* particionam os dados de uma informação em um conjunto de n partes (ou blocos) de tal maneira que t partes (com $t<n$) são suficientes para que se recupere a informação original. Com $t-1$ ou menos partes os dados, como um todo, não podem ser recuperados. No entanto, o uso destas técnicas de recuperação de erros não garante a confidencialidade das informações; é necessário a encriptação das mesmas.

Em um esquema de criptografia de limiar, chamado de compartilhamento de segredo, a partir de uma transformação, um segredo S dá origem a n *shadows* (partes). Com isto, cada uma destas partes do segredo não permite qualquer inferência sobre o segredo S. Esta informação só pode ser reconstruída a partir da obtenção de um conjunto mínimo de t partes ($0 < t \leq n$). Desta forma, com um conjunto de k partes do segredo, com $k < t$, não se consegue recuperar informação alguma de S. Os primeiros esquemas de compartilhamento de segredo propostos na literatura [Shamir 1979] definiam compartilhamentos de segredo baseados em interpolação polinomial e geometria plana (intersecção de retas). Porém, alterações destas partes poderiam acabar por comprometer a recuperação de S. Para lidar com cenários onde as partes estão sob a ação de entidades maliciosas que agem ativamente contra os protocolos de restauração dos segredos, foram criados os mecanismos de compartilhamento de segredo verificável [Schoenmakers 1999], em que cada fragmento de um segredo fornecido a um combinador pode ter sua validade verificada.

Vários sistemas, que por armazenarem arquivos em *clouds* baseados em transformações *IDA* e seus códigos de correção de erro, precisam que seus arquivos de informações sejam criptografados para garantir a confidencialidade dos mesmos. As

chaves usadas na encriptação das informações também precisam ser protegidas e neste caso são aplicados esquemas de compartilhamento de segredo. No nosso esquema, as informações (credenciais e atributos de usuários) a serem armazenadas tem um tamanho pequeno (poucos *bytes)* e não se torna proibitivo a aplicação de esquemas de compartilhamento de segredo diretamente nestas informações.

Na nossa abordagem, portanto, aplicamos somente um esquema de compartilhamento de segredo sobre as informações de usuário que decompõe as mesmas em *n* partes. Estas partes são então distribuídas entre *n* diferentes entidades (uma por cada uma das *n* entre as *m clouds* do sistema). As informações são recuperadas a partir de no mínimo *t* partes que estão armazenadas em *n* das *m clouds* do sistema, onde $0 < t < n < m$. Nas experimentações, usamos dois esquemas de compartilhamento de segredo: um não verificável [Shamir 1979] e um verificável [Schoenmakers 1999].

2.3 Premissas Assumidas

O fato de colocarmos os nossos dados pessoais em recursos gerenciados por terceiros envolve uma preocupação com a confidencialidade e a integridade destes dados. Alguns autores assumem que provedores de *cloud* mantêm um comportamento honesto, mas curioso (*honest-but-curious*). Este tipo de comportamento limita as ações de entidades de uma *cloud* a apenas leitura de informações em dados armazenados nos recursos da *cloud*. Em nossa abordagem, assumimos premissas menos restritivas. Cada provedor de *cloud* pode ser confiável, mas entidades maliciosas podem agir internamente devido à grande disponibilidade dos recursos de uma *cloud* a acessos externos ou mesmo internos. Estas entidades podem formar conluios para quebrar segredos. Os segredos dos usuários são mantidos, em nossa abordagem, quando o número de *clouds* sofrendo intrusões e falhas no sistema não ultrapassa o limiar *f* que definimos como sendo de valor *t-1* ($f = t - 1$). Nesta situação nossos protocolos funcionam corretamente e os segredos das informações não são quebrados e podem ser seguramente transformados.

3. Arquitetura Proposta

A arquitetura para a autenticação em uma federação de *clouds* é ilustrada na Figura 2. Os provedores de identidades (*IdP*), para implementarem os serviços de autenticação, usam as especificações *OpenID Connect* e, como não armazenam os atributos dos usuários, ficam reduzidos a poucas funcionalidades. Os protocolos *OpenID* trabalham sobre uma *cache.* A recuperação das informações de usuário diretamente das diversas *clouds* apresenta restrições de desempenho devido a questões de rede e ao próprio custo dos algoritmos envolvidos. A *cache* é então justificada pela manutenção dos atributos e credenciais de usuários mais recentes e frequentes, evitando sempre que possível a execução dos protocolos de combinação e a recuperação das informações de usuários.

Os atributos e credenciais de usuários que são inseridos nesta *cache* são recuperados da base de dados distribuída na federação de *clouds*. Para tanto, nos *IdPs*, são necessárias as funcionalidades da camada *Secret Sharing* (Figura 2), cujos algoritmos executam a função de combinar as partes dos segredos espalhados pelas *clouds* e, a partir desta combinação, tornam disponível os atributos de usuários na *cache*. Para que esta camada de compartilhamento de segredos (*Secret Sharing*) possa obter as partes de um segredo referente a um usuário na federação de *clouds*, é necessário o uso de *client stubs*

dos diversos bancos de dados usados no armazenamento dos mesmos na federação de *clouds*. Na Figura 2, as *stubs* são apresentadas como parte da camada *Database Client*.

A camada *Account Management* (Figura 2) permite que registros com credenciais e atributos de usuários sejam criados ou alterados em base de dados distribuída na federação. O *Account Management* faz uso mais amplo da camada *Secret Sharing*. Além da combinação de partes de um segredo, algoritmos de geração de partes de um segredo são fornecidos nesta camada. Por sua vez, a camada *Database Client* fornece os acessos (*read*/*write*) aos serviços de base de dados das diversas *clouds* da federação no armazenamento e na recuperação das informações de usuários durante a criação de suas contas. A Figura 2 mostra ainda os provedores de *cloud* da federação. Nas demanda de recursos, as verificações de autorização são baseadas em asserções de autenticação recebidas de *IdPs* da federação. Com isto, as *clouds* devem manter as listas dos *IdPs* com os quais possuem relações de confiança e seus respectivos certificados. Baseadas nestas listas, estas *clouds* podem validar as asserções de autenticação emitidas pelos *IdPs* da federação. Em relação ao sistema de autenticação, estes provedores de *cloud* tornam disponíveis os serviços de base de dados que armazenam as partes de segredos referentes às credenciais e atributos de usuários.

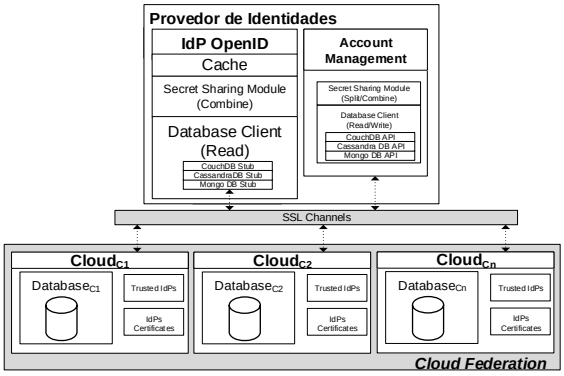

Figura 2 – Organização do Sistema de Autenticação

4. Protocolos da Abordagem

Assumimos duas possibilidades na criação de contas: na primeira, a interface *Account Management* é acionada por pessoal credenciado da *cloud* e é necessária a presença física do usuário; na alternativa, o próprio usuário faz o seu registro através da Internet.

Na criação de conta com a exigência de presença física, o cadastro de usuários é efetuado segundo os conceitos de segurança definidos em [Burr et al. 2006], onde a validade da identidade assegurada é de nível *LoA 4* (*Level of Assurance 4*). Para este nível, o usuário deve comparecer junto ao agente credenciado pela *cloud*, com os documentos necessários para garantir a veracidade de suas informações em seu registro. O agente então, com base nestes documentos, introduz os dados do usuário no sistema. Na criação de contas por acesso remoto à interface da camada *Account Management*, o usuário deve fornecer um certificado de chave pública assinado por autoridade certificadora oficial (*AC* de uma *PKI* oficial), ou ainda, por uma *AC* reconhecida pelos provedores de *cloud*. Nesta alternativa, o usuário fornece ou altera seus atributos e credenciais, devidamente assinados por sua chave privada (*LoA 3*). Uma descrição do protocolo de criação de conta de usuário (segundo o nível *LoA 3*) é ilustrada na Figura 3.

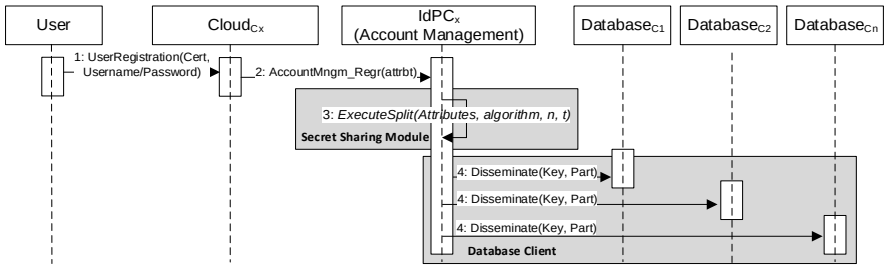

Figura 3 – Registro de Usuários

Em um primeiro passo, o usuário preenche o formulário de criação de conta através de um *IdP* da federação, introduzindo suas informações para o seu registro no sistema (passo 1, na Figura 3). Os dados passam por uma validação local (verificação da correção do certificado do usuário e também das assinaturas sobre os atributos passados pelo usuário). Uma vez validadas estas informações de usuário, as mesmas são passadas à camada *Secret Sharing* para que sejam geradas as partes do segredo e disseminadas entre as *clouds*, na base de dados distribuída que vai alimentar os *IdPs* do sistema. Este processo envolve algumas definições como: a escolha do esquema de compartilhamento de segredo, o número de partes do segredo e o limiar (número mínimo de partes) para recuperação dos segredos, dentre outros (passo 2). Após a geração das partes do segredo a partir das informações do usuário (passo 3), a distribuição destas se dá através das *stubs* apropriadas da camada *Database Client* (passos 4 na Figura 3).

Outro protocolo de nossa abordagem é descrito pelo processo de autenticação dos usuários do sistema, apresentado na Figura 4. Este procedimento é iniciado quando o usuário acessa um provedor de *cloud* para requisitar recursos (passo 1, Figura 4) que, por sua vez, verifica se este usuário mantém alguma sessão de autenticação já válida. Caso não tenha, a *cloud* estabelece comunicação com o *IdP* (passo 2). Se este *IdP* possui asserção de autenticação do usuário, no caso do mesmo já estar autenticado e acessando recursos em outra *cloud* (passo 3), a asserção correspondente é então enviada ao provedor de *cloud* requisitante (passo 4). Caso esta asserção de autenticação também não exista no *IdP* contatado, o usuário é direcionado para este provedor de identidades para que sua autenticação se concretize. O processo de autenticação é iniciado com o *IdP* solicitando ao usuário suas credenciais de autenticação (passo 5). Após a inserção destas credenciais (passo 6), o *IdP* faz consulta para verificar se as informações de *login* do usuário estão em sua *cache* (passo 7).

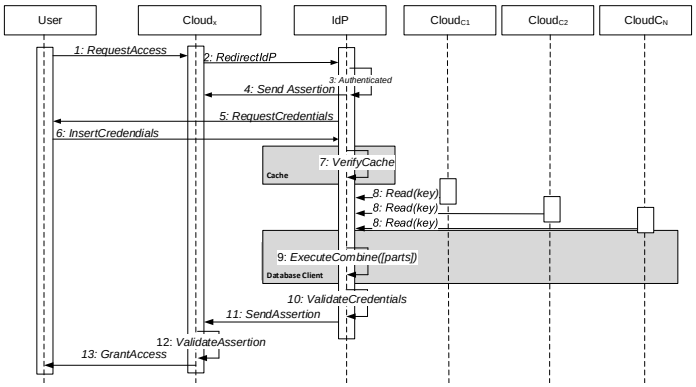

Figura 4 – Autenticação de Usuário

Se estas informações estão presentes, o *IdP* confronta com as credenciais fornecidas, validando o usuário com a geração de uma asserção (passo 10). Caso não estejam disponíveis na *cache*, o *IdP* deve buscar as informações do usuário na base de dados distribuída entre *clouds* da federação (passo 8). A busca de informações do usuário na federação faz uso da camada *Database Client* para a recuperação das partes de segredo que foram distribuídas entre *clouds* da federação. Com *t* partes do mesmo segredo, a camada *Secret Sharing* faz a combinação e recupera as credenciais e atributos do usuário correspondente, tornado disponíveis as mesmas na cache do *IdP* (passo 9).

De posse destas informações, o *IdP* faz as validações necessárias sobre as credenciais informadas pelo usuário (Passo 10) e em caso afirmativo envia para o provedor de *cloud* uma cópia da asserção gerada no processo de autenticação (passo 10). A Figura 4 mostra também a recepção da asserção de autenticação pelo provedor de *cloud* que efetua a validação da mesma com as verificações dos *nonces*, data de validade da asserção e assinaturas (passo 12). É importante ressaltar que o *IdP* em questão deve estar na lista de *IdPs* confiáveis do provedor de *cloud* (possui o certificado de chave pública do *IdP*). Com a asserção válida, o provedor de *Cloud* inicia o tratamento da requisição do usuário e finaliza o processo de autenticação (Passo 13).

5. Protótipo e Resultados

O sistema desenvolvido permite a autenticação de usuários e a geração de asserções de autenticações, que podem ser utilizadas por provedores de *cloud* de toda a federação, como forma de atestar a autenticidade do usuário. A Figura 5 descreve uma visão das ferramentas utilizadas na implementação do protótipo do sistema de autenticação. Na camada *Secret Sharing* que provê as funcionalidades de compartilhamento de segredos, foram implementados os esquemas de *Shamir* e *PVSS*. Os testes foram automatizados através da ferramenta *Apache JMeter*, de onde foram retirados os resultados apresentados nos gráficos das próximas seções.

O serviço provê somente a autenticação de usuários, pois o registro de usuário, como citado no texto, é executado por extensão (a interface *Account Management*), em pilha de protocolos própria a partir dos provedores de identidades. Os atributos e credenciais dos usuários registrados no sistema são armazenados em diferentes provedores de *cloud* que oferecem serviços de base de dados. Atualmente utilizamos serviços de bases de dados gratuitas, sem garantias de seus serviços (não garantem a qualidade do canal de comunicação, a disponibilidade das informações, etc.). Porém, para testar a viabilidade da proposta estes serviços de bases de dados se mostraram suficientes.

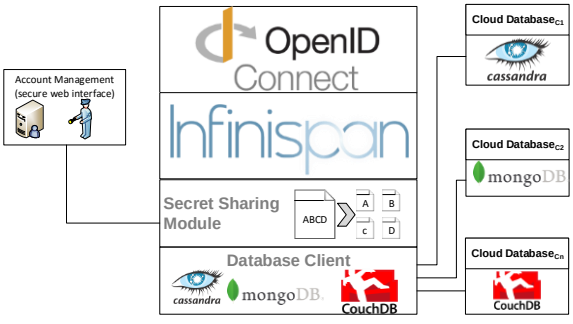

Figura 5 – Tecnologias utilizadas

5.1 Testes e Resultados

Os testes realizados com o protótipo do sistema de autenticação proposto envolveram as operações de registro de usuários, autenticação de usuários e geração das asserções de autenticação correspondentes. Para a realização destes testes, 1000 registros de usuários contendo as credenciais (usuário e senha) e atributos (nome, sobrenome, endereço, data de nascimento, profissão, etc.) foram gerados e armazenados utilizando os dois algoritmos de compartilhamento de segredos. Diferentes parâmetros *n* e *t* foram utilizados (representados nos gráficos como *Algorithm Parameters*).

5.1.1 Testes sobre o Tempo de Registro de Usuários

Neste teste foi medido o tempo necessário para a geração das partes de segredo pelos algoritmos de compartilhamento e também no armazenamento dos dados resultantes nas diversas *clouds* da federação, representando o registro de usuários através da interface *Account Management*. A Figura 6 apresenta os tempos de registro das implementações dos esquemas de *Shamir* e do *PVSS*, respectivamente.

Diante dos resultados apresentados na Figura 6 podemos observar que o tempo de processamento na geração das partes de segredo (operação de *Split* na camada *Secret Sharing*) para o algoritmo *PVSS* é menor se comparado com o algoritmo *Shamir*. Em ambos os casos, o tempo de processamento está relacionado com o método empregado para se encontrar o valor do módulo a ser usado nas operações (aritmética modular) dos esquemas de segredo. No *PVSS* este módulo é definido pelo tamanho da chave criptográfica usada (512 *bits* em nossa simulação), enquanto no método de *Shamir* o mesmo é assumido como o tamanho dos segredos considerados (em nossa simulação os segredos são de aproximadamente 256 *bytes*).

O tempo de envio dos dados para as bases de dados é ligeiramente menor quando utilizadas as implementações do esquema de *Shamir*. Isto se deve ao fato de que a quantidade de dados gerados por este algoritmo é menor que a quantidade de dados gerados no esquema *PVSS*. São enviadas com o *PVSS* mais informações para o armazenamento, como as provas para verificação da correção das partes do segredo.

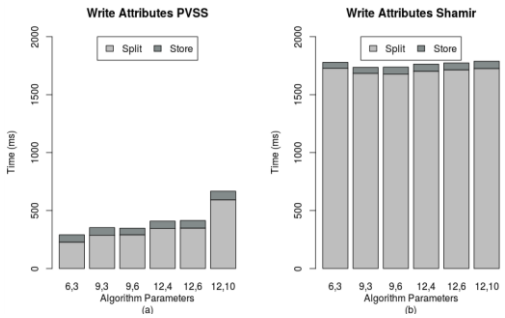

Figura 6 – Tempos de Registro

5.1.2 Testes sobre o Tempo de Consulta (Tempo de Autenticação)

Os tempos de consulta (referente ao tempo correspondente a execução de *querying* por dados pelo *IdP* e a disponibilidade dos mesmos em *cache*) corresponde praticamente ao tempo para a autenticação de um usuário no sistema (desconsiderando o tempo de verificação em *cache*). Na obtenção destes tempos, consideramos primeiro uma pequena

carga de utilização (10 usuários simultâneos) e, posteriormente, uma carga de utilização bem mais significativa (100 usuários simultâneos). A Figura 7 apresenta os tempos de consulta envolvendo também as diferentes configurações de protocolos de compartilhamento de segredo (o *PVSS* e o *Shamir*).

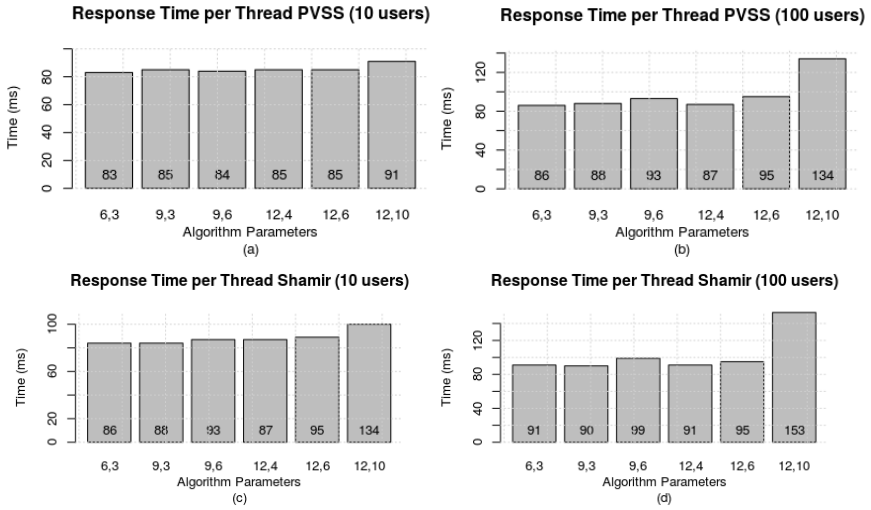

Figura 7 – Tempo de Autenticação

É possível perceber que este tempo de *querying* tem pequenas variações quando consideramos dez (10) usuários simultâneos (Figura 7.a e Figura 7.c). O mesmo não acontece nos testes de cem (100) usuários simultâneos (Figura 7.b e Figura 7.d). As leituras nas diferentes bases de dados das *clouds* acontecem de forma paralela, de modo que o tempo de *querying* será sempre determinado pela *cloud* que responde mais lentamente. No caso de muitos usuários simultâneos, o acesso às informações começa a apresentar baixo desempenho pelos problemas já comentados sobre a contenção nos canais. Apesar de não presentes nos gráficos, o tempo de execução dos protocolos de *Secret Sharing* (por operação) tem seus valores em uma razão menor que 1 *ms* nos casos com 10 usuários e aproximadamente 4,6 *ms* nos casos com 100 usuários.

5.1.3 Testes de Vazão de Autenticações

Estes testes tiveram como objetivo verificar a vazão (operações de autenticação por segundo) do protótipo desenvolvido. A Figura 8 apresenta os resultados obtidos. É possível perceber que os valores de vazão começam a diminuir significativamente a partir de aproximadamente 70 usuários concorrentes no processo de autenticação no sistema. Após a leitura dos resultados percebemos que o aumento do número de usuários simultâneos acaba impactando fortemente no desempenho do sistema.

Porém, em uma análise preliminar foi possível constatar que o problema está relacionado à implementação das estruturas de dados compartilhadas entre as *threads*. Uma implementação mais criteriosa das estruturas de dados compartilhadas e o uso de bases de dados com um maior *QoS* poderia apresentar melhora nestes valores, assim como amenizar o efeito de gargalo no caso de múltiplos usuários.

5.1.4 Testes com o uso de *cache*

Em nossos testes anteriores, sempre levamos em conta que os atributos de usuários não estavam em *cache*. Por outro lado, a Figura 9 apresenta a vazão de autenticações com o

uso de *cache,* porém com somente dados de um pequeno número de usuários. Nestes testes, foram considerados 10% dos registros de usuários presentes na *cache* para consulta. A escolha do valor de 10% de registros em *cache* se justifica diante do número potencial de usuários de uma federação de *clouds*. As autenticações que fazem uso destes 10% de registros em *cache* não executam o algoritmo de compartilhamento de segredos (camada *Secret Sharing*) e tampouco a busca das partes na federação. Com isto, foi possível perceber o acréscimo da vazão de autenticações em relação aos testes onde as informações não estavam em *cache.* Quando o esquema de *Shamir* é usado nesta camada *Secret Sharing*, este aumento é de aproximadamente 13%. Com o PVSS o acréscimo na vazão foi de aproximadamente 15%.

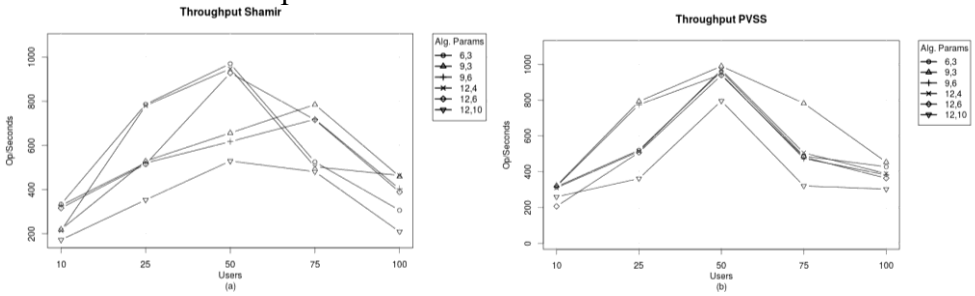

Figura 8 – Vazão de autenticações (operações por segundo)

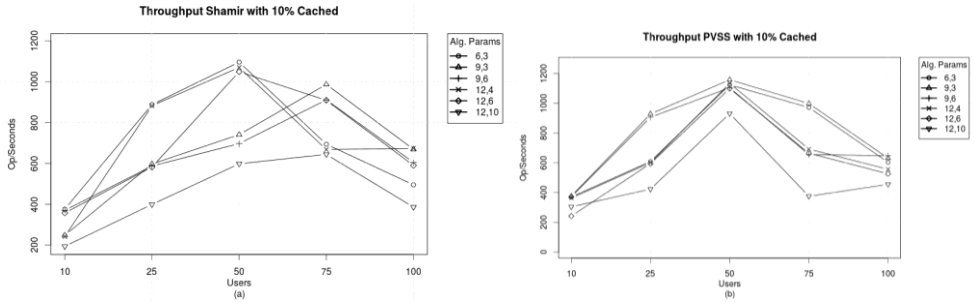

Figura 9 – Vazão com o uso de *cache* (operações por segundo)

5.1.5 Testes dos algoritmos com a presença de intrusões

Nesta seção buscamos avaliar as implementações nos casos onde intrusos podem alterar as partes do segredo comprometendo a montagem dos atributos. Para este fim, partes dos segredos foram alteradas, simulando uma intrusão e o comprometimento de partes dos segredos nas bases de dados distribuídas nas clouds da federação. Nestes testes, foi considerado o número máximo de intrusões (n - t intrusões), ou seja, todos os registros de segredos distribuídos na federação de clouds terão n - t de suas partes corrompidas.

5.1.6 Vazão dos algoritmos com intrusões

Nesta seção, são apresentados os resultados de vazão de autenticações, considerando os algoritmos de *Shamir* e *PVSS* na camada *Secret Sharing* e a presença de intrusões nas *clouds* da federação. O algoritmo PVSS, por se tratar de um esquema de compartilhamento verificável, permite que as partes de um segredo quando recebidas sejam verificadas. As verificações das partes usam as provas geradas no momento do processo de particionamento do segredo [Schoenmakers 1999].

De forma distinta ao protocolo *PVSS*, a abordagem de *Shamir* para o compartilhamento de segredo não apresenta nenhum recurso que permita a verificação da integridade das partes. Basta uma parte corrompida sendo usada para que o algoritmo retorne um valor inadequado. A alternativa para recuperar o segredo é fazer o uso de combinações de *t* partes dentre as *n* disponíveis (C_n^t), até que se consiga *t* partes corretas. Para fins de análise, buscamos então definir dois limites para estes valores: o primeiro foi chamado de *Best Case* que corresponde ao uso de *t* partes corretas na primeira iteração do algoritmo de *Shamir*, com o mesmo retornando de imediato o segredo; o segundo limite identificamos como *Worst Case*, onde somente na última interação do algoritmo ocorre a combinação com *t* partes corretas para conseguir o segredo desejado. A Figura 10 mostra resultados para estas vazões quando o algoritmo utilizado é o *PVSS*. Podemos perceber a partir destes resultados que os valores de vazão (em presença de intrusões) sofrem uma diminuição de aproximadamente 50% quando comparados com os casos sem intrusões. Os testes de vazão realizados utilizando o esquema de *Shamir* na camada *Secret Sharing* são mostrados na Figura 11. Neste caso, a vazão de autenticações apresenta valores de aproximadamente 55% do valor do mesmo teste sem intrusões em *Best Case* e valores de vazão extremamente baixos (menores que uma operação por segundo) em *Worst Case*.

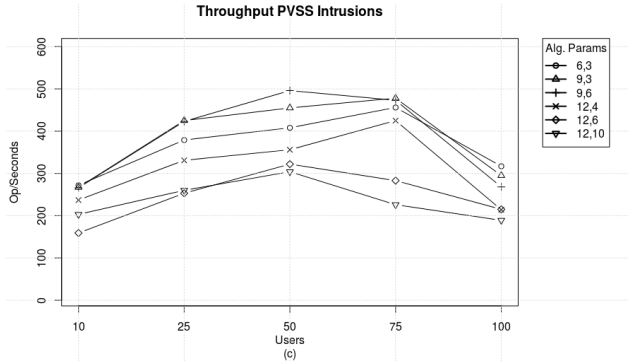

Figura 10 – Vazão com intrusões utilizando PVSS (operações por segundo)

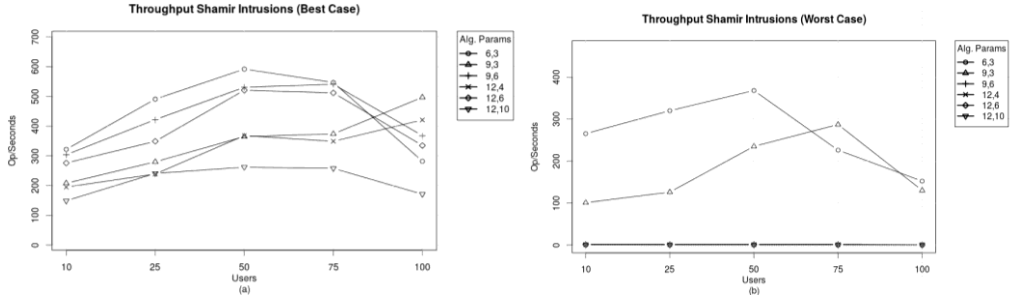

Figura 11 – Vazão com intrusões utilizando Shamir (operações por segundo)

6. Literatura Relacionada

Vários trabalhos na literatura tratam de atributos de usuários e autenticação em *clouds*. Mas, poucos destes coincidem nos seus objetivos com as preocupações que orientaram nossos trabalhos. Na sequência discutimos alguns destes trabalhos.

O projeto *SPICE* [Chow et al. 2012] descreve experiências com o objetivo de proteger a identidade de usuários que utilizam serviços em *clouds*. Os autores propõem a criação para cada usuário de um grupo de assinaturas [Boneh et al. 2004] que são

usadas para a autenticação do usuário nos diferentes provedores de *cloud*, onde cada assinatura gerada provê os atributos necessários para um determinado provedor. Outro *framework* semelhante, utilizando criptografia de grupo, é descrito em [Yan et al. 2009].

BlindIdM [Nuñez and Agudo 2014] discute os problemas de segurança relacionados ao envio de atributos de usuários para provedores de *cloud*. O objetivo principal da proposta é a proteção dos atributos de usuários, uma vez que os autores assumem que as *clouds* são ambientes com segurança limitada e que não podem controlar manipulação indiscriminada dos dados de seus usuários. Os autores então propõe a existência de um "*Blind Identity Provider*", que funciona como uma camada intermediária entre os provedores de identidades das *clouds* e os donos dos atributos.

Em Bertino [Bertino et al. 2009] a segurança dos atributos de usuários é garantida através do uso de duas técnicas: a primeira, onde o usuário e o consumidor da identidade (*cloud provider*) entram em acordo de forma a definir *metadados* para atributos de usuários. Estes *metadados* acordados entre as partes definem como as informações devem ser enviadas pelo usuário e como as mesmas devem ser interpretadas pelo provedor de *cloud*. Estes *metadados* são protegidos por assinaturas.

Estes trabalhos citados acima têm como objetivo facilitar aos usuários na manipulação de seus atributos e restringir a revelação dos mesmos nos acessos a provedores de *cloud*. A nossa abordagem, embora proteja os atributos de usuários, possui um foco diferente. No nosso caso, as *clouds* estão associadas em federação para fornecer serviços e recursos aos seus usuários. A autenticação do usuário, na nossa proposta, é feita no seu *IdP* de registro (na sua *cloud*) e é aceita pelas demais *clouds* da federação. O *BlindIdM* e a proposta apresentada em Bertino, também como a nossa abordagem, tentam evitar a disponibilidade de atributos do usuário nas *clouds*. Mas a ação dos mecanismos dos trabalhos citados está limitada às interações dos usuários no processo de autenticação nas *clouds*. Nós, em nossa abordagem, estamos propondo o armazenamento seguro destas informações em *clouds* da federação. E a autenticação em nosso sistema envolve provedores de identidades que tratam somente da autenticação de usuários em *clouds* desta federação.

Em [Angin et al. 2010], os autores apresentam uma proposta onde os atributos de usuário sempre estão de posse do usuário, nunca armazenados em um provedor de identidades. Isto garante melhorias no controle de que dados serão liberados para as *clouds* pois o usuário é quem define quais atributos liberar. Além disso, para manter atributos de forma segura, a abordagem utiliza assinatura do tipo *Zero-knowledge proof*, de forma que o usuário seja a própria entidade a certificar os atributos enviados**.** Uma abordagem semelhante foi apresentada [Böger et al. 2011] onde os atributos de usuário são mantidos em cartão (*smartcards*). A autenticação é feita via certificados e os atributos de usuário são enviados do *smartcard* para o provedor de serviço, via o *IdP* do sistema. São políticas *user-centric* que definem quais atributos liberar.

A literatura apresenta inúmeras experiências [Bessani et al. 2011; Sujana et al. 2013] na construção de memória segura em *clouds* ou em federação de *clouds*. Estes trabalhos fazem uso de esquemas de compartilhamento de segredo e de algoritmos de dispersão de informação para a memorização segura em *clouds*. Nestas abordagens, a confidencialidade das informações é conseguida com a encriptação das mesmas, usando uma chave simétrica gerada aleatoriamente. Em seguida, o arquivo cifrado é subdividi-

do a uma transformação *IDA*. Por sua vez, a chave criptográfica usada na proteção da informação passa por um processo de compartilhamento de segredo, gerando também partes relacionadas com a chave. Esta distribuição deve evitar que tanto as partes do arquivo como as partes da chave venham a ser enviadas aos mesmos provedores de modo a comprometer os seus segredos. A nossa abordagem é mais econômica no uso destas técnicas. Como os atributos e credenciais de usuários correspondem a registros de poucos bytes, na nossa abordagem podemos aplicar diretamente o compartilhamento de segredo sobre estas informações, evitando com isto o uso de técnicas de correção de erro e de criptografia para a confidencialidade destas informações.

7. Conclusão

Este artigo apresenta uma abordagem que faz uso de provedores de *cloud* para armazenar as informações de usuário (credenciais e atributos). Neste sentido, descrevemos um modelo de autenticação para uso em federações de provedores de *cloud*. Uma arquitetura foi desenvolvida para implementar um protótipo de nossas proposições que definem o modelo. Como ambientes de *clouds* apresentam segurança limitada, aplicamos técnicas de compartilhamento de segredo nas informações de usuário, permitindo a distribuição de partes destes segredos nas várias *clouds*. Testes com ou sem intrusões e usando diferentes esquemas de compartilhamento de segredo foram realizados para mostrar a eficiência de nossos mecanismos.

O modelo apresentado provê certa flexibilidade do acesso às informações permitindo que usuários possam fazer os seus *logins* a partir de um *IdP* de uma federação de *clouds* e ter acessos a recursos em diferentes partes desta federação. A abordagem de gerenciamento de identidades segue o modelo centralizado devido ao uso de protocolos do *OpenID*. Mas pela flexibilidade da disponibilidade das informações nas *clouds*, temos as mesmas características do gerenciamento de identidades federadas, porém sem a necessidade de relações de confiança entre *IdPs*.

Vale a pena ressaltar que, ainda que o processo de criação de registros apresente um desempenho menor do que os tempos de autenticação (até 10 vezes maiores), por serem menos frequentes, estas criações ou alterações de contas não chegam a influenciar no desempenho do sistema de autenticação.

O trabalho apresentado neste texto é o complemento de outra experiência descrita em [Barreto et al. 2013] onde os provedores de identidades são desenvolvidos com o uso extensivo da tecnologia de virtualização para isolar as informações de usuários. Intrusos não têm acesso a *cache* com registros de usuários que é mantida em compartimento isolado e distante da rede. O modelo de autenticação apresentado não se restringe a ambientes de federação de *clouds*. Qualquer *IdP* pode ser concebido usando a esta e fazer uso de alguns provedores de *cloud* que não formem associações para o compartilhamento de recursos como as citadas federações.

8. Agradecimentos

Agradecemos a CAPES pelo apoio financeiro com bolsa de doutorado e mestrado e bolsa CNPQ Processo [233648/2014-3].

Referências

Angin, P., Bhargava, B., Ranchal, R., et al. (oct 2010). An Entity-Centric Approach for Privacy and Identity Management in Cloud Computing. *2010 29th IEEE Symposium on Reliable Distributed Systems*, p. 177–183.

Barreto, L., Siqueira, F., Fraga, J. and Feitosa, E. (2013). An Intrusion Tolerant Identity Management Infrastructure for Cloud Computing Services. In *2013 IEEE International Conference On Web Services*.

Bertino, E., Lafayette, W., Paci, F. and Ferrini, R. (2009). Privacy-preserving Digital Identity Management for Cloud Computing. *Identity*, v. 32, p. 1–7.

Bessani, A., Correia, M., Quaresma, B. and Sousa, P. (2011). DEPSKY: Dependable and Secure Storage in a Cloud-of-Clouds. *European Systems Conference.*

Böger, D., Barreto, L., Fraga, J., et al. (2011). User-Centric Identity Management Based on Secure Elements. n. 590047.

Boneh, D., Boyen, X. and Shacham, H. (2004). Short Group Signatures. *Advances in Cryptology - CRYPTO 2004*, v. 3152, p. 227–242.

Burr, W. E., Dodson, D. F. and Polk, W. T. (2006). Electronic authentication guideline. *NIST Special Publication*, v. 800:63.

Celesti, A., Tusa, F., Villari, M. and Puliafito, A. (2010). How to Enhance Cloud Architectures to Enable Cross-Federation. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*.

Chow, S., He, Y., Hui, L. and Yiu, S. (2012). Spice–simple privacy-preserving identity-management for cloud environment. *Applied Cryptography and Network, p.*526–543.

Grozev, N. and Buyya, R. (2012). Inter-Cloud architectures and application brokering: taxonomy and survey. *Software: Practice and Experience*, p. 1–22.

Jøsang, A., Fabre, J., Hay, B., Dalziel, J. and Pope, S. (2005). Trust requirements in identity management. *CRPIT '44: Proceedings of the 2005 Australasian workshop on Grid computing and e-research*, p. 99–108.

Nuñez, D. and Agudo, I. (2014). BlindIdM: A privacy-preserving approach for identity management as a service. *International Journal of Information Security*, p. 199–215.

Rabin, M. O. (1989). Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, v. 36, n. 2, p. 335–348.

Schoenmakers, B. (1999). A Simple Publicly Verifiable Secret Sharing Scheme and its Application to Electronic Voting. *Advances in Cryptology (CRYPTO99)*, p. 148–164.

Shamir, A. (1979). How to share a secret. *Communications of the ACM*, p. 612–613.

Sujana, B., Tejaswini, P., Srinivasulu, G. and Karimulla, S. (2013). Secure Framework for Data Storage from Single to Multi clouds in Cloud Networking. v. 2, n. 2.

Yan, L., Rong, C. and Zhao, G. (2009). Strengthen cloud computing security with federal identity management using hierarchical identity-based cryptography. p. 167–177.

