MARCEL RODRIGUES DE BARROS

PALOMA BASTOS FONZAR

MOBILEAUTH – SISTEMA DE AUTENTICAÇÃO MÓVEL PARA OPENID

Trabalho de Formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Bacharelado em Engenharia Elétrica com Ênfase em Computação.

São Paulo
2011

MARCEL RODRIGUES DE BARROS

PALOMA BASTOS FONZAR

MOBILEAUTH – SISTEMA DE AUTENTICAÇÃO MÓVEL PARA OPENID

Trabalho de Formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Bacharelado em Engenharia Elétrica com Ênfase em Computação.

Área de Concentração:
Sistemas Digitais

Orientadora: Profª. Dra. Tereza Cristina Melo de Brito Carvalho

São Paulo
2011

AGRADECIMENTOS

Agradecemos a nossa orientadora, Profª. Dra. Tereza Cristina Melo de Brito Carvalho, por ter se disponibilizado a orientar este trabalho mesmo com seu pouco tempo disponível.

Aos nossos co-orientadores, Charles Christian Miers e Rony Sakuragui, pelas profundas contribuições neste trabalho desde o momento da concepção inicial até as revisões finais deste documento.

Agradecemos a todos os nossos professores, que contribuíram direta ou indiretamente com os conhecimentos necessários à realização do projeto. Em especial, ao Prof. Dr. Marcos Simplício Jr., por ter contribuído com seus conhecimentos sobre segurança de sistemas.

Agradecemos aos nossos pais e família, por nos darem a oportunidade de receber uma boa educação e por nos incentivarem mesmo nos momentos mais difíceis da graduação. Ainda, por nos fornecerem um ambiente de conforto e apoio que nos sustentou durante os últimos cinco anos.

Finalmente, agradecemos a todos os nossos amigos, em especial à Lorenna, Helena, Guilherme e Bruna, pela paciência durante este último ano e pelo apoio que nos deram, compreendendo nossa ausência em diversas situações. Agradecemos também por nos fornecerem os momentos de descontração necessários toda vez que precisávamos de um pouco de descanso.

RESUMO

Com o aumento contínuo da quantidade e diversidade de serviços oferecidos no ambiente Web, começaram a surgir tecnologias de gerenciamento de identidade que visam à centralização da identidade digital de um usuário. Essas tecnologias tem o objetivo de facilitar a utilização de serviços eliminando a necessidade da criação de contas diferentes para cada serviço na *Web*. Por outro lado, essas tecnologias trazem consigo preocupações em relação à segurança da identidade, uma vez que, neste cenário, um roubo de credenciais é muito mais prejudicial, já que com uma única identidade poder-se-ia acessar todos os serviços de um usuário. Com isso, este trabalho tem como objetivo melhorar a segurança de sistemas de gerenciamento de identidade por meio da proposta de um módulo remoto de autenticação integrado ao provedor de identidade centralizada, que funcione como uma autenticação redundante para complementar o tradicional uso das credenciais composto por nome de usuário e senha. Adicionalmente deseja-se que o módulo de autenticação não represente um obstáculo ao uso da tecnologia, para isto o mesmo foi desenvolvido no formato de uma aplicação para *smartphones*, dispositivos que se mostram cada vez mais utilizados por usuários de telefones celulares. Com essa preocupação pretendemos ilustrar a questão do *trade-off* entre segurança e usabilidade por meio de opções distintas de autenticação fornecidas pelo módulo e a análise dos respectivos impactos na forma de uso dos serviços.

Palavras-chave: segurança da informação, usabilidade, Internet, provedor de identidade

ABSTRACT

With the continuous increase in diversity and quantity of services found in the Web environment, alternatives in centralized identity management start to rise. These technologies aim to facilitate the services usage by eliminating the need of creating different accounts to each desired purpose. However, they bring questioning about the security of the identity, since, in this scenario, the credentials theft is much more harmful, as the attacker would be able to grant access to all user services with the unique identity. This work goal is to reduce the vulnerability described by proposing a remote authentication module integrated to the identity provider, which works as a redundant authentication needed to complete the traditional usage of username and password credentials. Additionally, the work intends that the authentication module does not act as an obstacle to the usage of identity management systems; to this end, it was developed as an application for smartphones, devices that are becoming more and more adopted by mobile phone users. With this concern, the work aims to illustrate the trade-off between security and usability by providing different authentication options and analyzing their impact on the way the services are used.

Keywords: information security, usability, Internet, identity provider

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3DES	*Triple Data Encryption Standard*
AES	*Advanced Encryption Standard*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
HTTPS	*HyperText Transfer Protocol Secure*
IMEI	*International Mobile Equipment Identity*
IMSI	*International Mobile Subscriber Identity*
OP	*OpenID Provider*
RP	*Relying Party*
SGI	Sistema de Gerenciamento de Identidades
SIM	*Subscriber Identity Module*
SREG	*Simple Registration Extensions*
TLS	*Transport Layer Security*
URL	*Uniform Resource Locator*

SUMÁRIO

1 INTRODUÇÃO

Com o aumento contínuo da quantidade e diversidade de serviços oferecidos no ambiente *Web*, começaram a surgir tecnologias de gerenciamento de identidade que visam à centralização da identidade digital de um usuário. Essas tecnologias tem o objetivo de facilitar a utilização de serviços eliminando a necessidade da criação de contas diferentes para cada serviço na *Web*. Por outro lado, essas tecnologias trazem consigo preocupações em relação à segurança da identidade, uma vez que, neste cenário, um roubo de credenciais é muito mais prejudicial, já que com uma única identidade poder-se-ia acessar todos os serviços de um usuário.

Diversos autores reconhecem a relação entre o aumento da segurança e a redução da usabilidade do sistema e vice-versa, como uma preocupação necessária a projetos de sistemas que buscam a aceitação no meio em que se inserem (YAN *et al.*, 2004)(YEE, 2004). Com isso em vista faz-se necessária uma análise sobre as consequências da adoção de um modelo que permita que uma mesma credencial conceda acesso a diversos serviços. O aumento da usabilidade desse sistema é visível em diversos sentidos, mas os riscos ocasionados por sua adoção inspiram especial atenção.

Adotando uma abordagem bastante aceita para o risco de uma aplicação computacional que afirma que o mesmo é composto pelo produto das ameaças a um sistema, pelas suas vulnerabilidades e pelo custo de uma violação de segurança[1] (ELKY, 2006), conclui-se que à medida que cresce o valor das informações gerenciadas assim como a ameaça às mesmas devem-se reduzir as vulnerabilidades para impedir o crescimento do risco do sistema. Faz-se clara então a necessidade de um aumento de segurança no cenário de gerenciamento identidades únicas para serviços na *Internet*, pois com a possibilidade de acesso aos múltiplos serviços com uma única credencial, crescem as possíveis ameaças e o valor do conjunto de informações acessível por meio deste.

Os sistemas de gerenciamento de identidade apresentam a mesma estrutura de acesso para todos os serviços associados a uma mesma conta (DHAMIJA; DUSSEAULT, 2008).

---

[1] Essa relação é comumente expressa pela equação: Risco = Vulnerabilidades x Ameaças x Custos.

Esse modelo trata serviços que possuam diferentes níveis de risco distintos de forma idêntica, estabelecendo características de segurança e usabilidade estáticas que apresentam boa aderência a alguns serviços, mas a outros não.

Esse trabalho enxerga que um tratamento diferenciado para serviços distintos associados a uma mesma identidade digital é o caminho adequado para uma melhor aderência das características de segurança e usabilidade às necessidades de cada usuário. Através de um módulo de autenticação redundante criar-se-á um mecanismo inicial para melhoria de segurança, que por sua vez possui a possibilidade de diferentes disposições das características de usabilidade e segurança de acordo com as diferentes necessidades de cada serviço.

1.1 Objetivo e Escopo

O objetivo do projeto de formatura com o tema MobileAuth – Sistema de Autenticação Móvel para *OpenID* é especificar e desenvolver um protótipo de um mecanismo de autenticação adicional para um provedor de identidade *OpenID*. A proposta do grupo é implantar uma aplicação em um smartphone que se comunique com o provedor de identidade para completar a autenticação e garantir que apenas o dono das informações consiga realizar o acesso.

Como um objetivo secundário do projeto deseja-se ilustrar o *trade-off* "Segurança VS. Usabilidade" por meio da utilização de diferentes opções de segurança para confirmar o acesso.

1.2 Organização do texto

O capítulo 2 deste documento contém uma breve descrição de conceitos de segurança da informação e visa introduzir o leitor aos conceitos e técnicas comumente utilizadas para o projeto de sistemas seguros. Por sua vez, o capítulo 3 descreve sistemas de gerenciamento de identidades focando na tecnologia *OpenID,* ainda no capítulo 3 é

abordado o *trade-off* entre usabilidade e segurança através de um breve comparativo com o capítulo 2. Estes dois últimos capítulos fornecem o embasamento para a proposta do projeto, que é especificada no capítulo 4. Já o capítulo 5 apresenta detalhes de desenvolvimento da proposta exposta no capítulo anterior, trazendo ao leitor detalhes sobre as características internas do sistema e dos algoritmos associados à proposta. Por fim, o capítulo 6 apresenta as considerações finais sobre o sistema, discutindo o atendimento aos objetivos iniciais, as contribuições e inovações, bem como os trabalhos futuros possivelmente relacionados ao projeto.

2 SEGURANÇA DA INFORMAÇÃO

Um sistema seguro é aquele que fornece informações íntegras somente a usuários autorizados, no momento em que elas são pedidas, através de requisições válidas e identificadas, não permitindo que essas informações sejam recebidas, observadas ou alteradas por terceiros não autorizados (NBR ISO/IEC 17799, 2001).

Existem diversas maneiras de constituir um sistema seguro, porém é importante ressaltar que um aspecto comum destes diversos métodos consiste na existência de um ou mais segredos confiáveis de posse dos interlocutores, que em sua ausência impeçam que terceiros acessem as informações que se deseja proteger.

Há diferentes aspectos que caracterizam a segurança de um sistema de computadores. Estes aspectos dizem respeito às diferentes características de troca de informações que devem ser protegidas e há grande divergência em relação à taxonomia que deve ser adotada. Tais aspectos podem ser denominados serviços de segurança (STALLINGS, 2002).

Esta seção visa descrever brevemente os serviços de segurança apresentando os conceitos de segurança relacionados que serão necessários à compreensão do sistema a ser proposto. Também serão apresentados métodos e ferramentas que buscam a implantação dos serviços de segurança em sistemas diversos. A partir dessas informações pretende-se fornecer o embasamento teórico e tecnológico necessários à discussão do cenário abordado neste trabalho.

2.1 Serviços de segurança

Uma nomenclatura bastante aceita para a definição da segurança de sistemas é a descrição dos serviços básicos de segurança que esse sistema fornece. A seguir estão listados e brevemente descritos quatro serviços básicos de segurança (STALLINGS, 2002).

- Confidencialidade:

Refere-se à garantia de que qualquer informação armazenada num sistema de computação ou transmitida via rede seja revelada, acessada e/ou manipulada somente por usuários devidamente autorizados.

- Integridade:

Refere-se à possibilidade de verificar a consistência da informação contida nos dados, impedindo que seja alterada indevidamente ou acidentalmente de maneira imperceptível.

- Legitimidade:

Garantia de que os recursos de um sistema não sejam utilizados por entidades não autorizadas ou de forma não autorizada. Este serviço pode ser dividido em dois sub-serviços.

a. Autenticidade: a identidade alegada por um usuário é verificável e intransferível.

b. Irretratabilidade: nem o remetente nem o destinatário das informações podem negar a sua transmissão, recepção ou posse.

- Disponibilidade:

Garantia de que os usuários legítimos não sejam impedidos indevidamente ou acidentalmente de acessarem as informações e os recursos do sistema aos quais tem permissão de uso.

2.2 Ferramentas e métodos

Existem diversos métodos e ferramentas que são comumente adotadas para o desenvolvimento dos serviços de segurança descritos na seção anterior. A seguir encontram-se breves abordagens das metodologias e ferramentas mais relevantes para a solução do problema abordado neste trabalho.

2.2.1 HTTPS

HTTPS é um protocolo desenvolvido para estabelecer conexões seguras de modo a prover um canal de comunicação do tipo HTTP que satisfaça alguns requisitos de segurança. O HTTPS foi viabilizado inicialmente através do protocolo SSL v1 (*Secure Socket Layer*) que foi sendo aprimorado (v2 e v3) para incluir novas capacidades e resultou no TLS (*Transport Layer Security*) que atualmente está na versão 1.3.

Sendo assim, atualmente o HTTPS é o uso do protocolo HTTP com a camada TLS de segurança[2], que tem como objetivo proporcionar um canal de comunicação seguro sobre uma conexão inicialmente insegura (DIERKS; RESCORLA, 2006).

É importante ressaltar que o HTTPS parte de duas premissas para fornecer seus serviços de segurança. Primeiramente, as partes que desejam se comunicar devem ser capazes de utilizar algoritmos de encriptação eficientes para a geração de suas chaves privadas. Uma das partes também deve estar certificada por um certificado digital emitido por uma autoridade certificadora de confiança (e.g. Verisign, Microsoft, etc...).

2.2.2 Chaves Simétricas

A criptografia com chaves simétricas consiste na utilização de uma mesma chave para realizar tanto a encriptação quanto a decriptação de uma mensagem. É aplicado um algoritmo criptográfico utilizando a chave que gera um texto cifrado. O mesmo algoritmo, porém em modo inverso, aplicado ao texto cifrado usando a mesma chave, recupera o texto claro (STALLINGS, 2002).

Este tipo de criptografia garante a confidencialidade das informações, assumindo que apenas remetente e destinatário possuam a chave comum capaz de decifrar a mensagem. Para garantir a segurança da criptografia simétrica, é preciso que o sistema resista a dois tipos de ataque: criptoanálise e força bruta. A resistência à criptoanálise

[2] Vide anexo 1.

diz respeito a características do algoritmo de criptografia utilizado, enquanto a resistência ao ataque por força bruta diz respeito à utilização de uma chave de tamanho suficiente para que o esforço de testar todas as combinações possíveis de caracteres não seja viável.

Os algoritmos mais utilizados para criptografia simétrica são o 3DES (*Triple Data Encryption Standard*) e o AES (*Advanced Encryption Standard)*, que utilizam, respectivamente, chaves de até 192 e 256 bits.

2.2.3 Chaves Assimétricas

A criptografia através de chaves assimétricas se refere ao uso de chaves distintas para a encriptação e decriptação de dados. As chaves são denominadas pública e privada, e possuem um relacionamento matemático. É importante ressaltar que esse relacionamento não deve possibilitar o descobrimento de uma chave a partir da outra e que toda mensagem cifrada com uma chave somente pode ser decifrada pelo seu par correspondente.

Uma chave privada deve permanecer secreta, enquanto a pública pode ser divulgada livremente sem comprometer a segurança da comunicação (STALLINGS, 2002).

O uso de chaves assimétricas possibilita dois serviços básicos de segurança. A autenticação de ambas as partes pode ser feita através do envio de mensagens codificadas com a chave privada de cada parte, que só podem ser decodificadas pela chave pública, garantindo assim a autenticidade do emissor. Por outro lado, a confidencialidade pode ser garantida através da codificação com a chave pública, o que permite a decodificação somente pelo possuidor da chave privada correspondente.

2.2.4 Certificados Digitais

Com o uso de chaves assimétricas o problema da autenticação recai sobre a autenticação de uma chave pública, ou seja, deve-se garantir que uma chave pública foi realmente gerada pelo emissor esperado.

Para a realização desse procedimento utilizam-se certificados digitais. Certificados digitais são agregados de dados armazenados segundo um formato padronizado, e.g. X.509[3], emitidos por um órgão de confiança, que atesta a autenticidade de uma informação (FARRELL; ADAMS, [S.d.]).

2.2.5 Resumos Criptográficos

Um resumo criptográfico, ou *hash*, consiste em um resumo de tamanho fixo de uma mensagem qualquer. Ao contrário da criptografia simétrica e assimétrica, para obter um resumo criptográfico não é necessário utilizar uma chave, ou seja, o resultado depende exclusivamente da mensagem. O *hash* é resultado de uma função de *hash* que deve apresentar algumas características particulares (STALLINGS, 2002):

- Resistência à primeira inversão: Dado um resumo criptográfico, é inviável encontrar uma mensagem que submetida à função de *hash* resulte neste resumo.
- Resistência à segunda inversão: Dados uma mensagem e seu resumo criptográfico, é inviável encontrar uma mensagem diferente que gere o mesmo resumo.
- Resistência a colisões: É inviável encontrar duas mensagens diferentes quaisquer que apresentem resumos criptográficos iguais.

[3] Detalhes do padrão X.509 podem ser encontrados no Anexo 2.

As funções de *hash* são algoritmos auxiliares aos sistemas criptográficos, fornecendo integridade à troca de mensagens quando estas são enviadas com seu resumo anexado (STINSON, 2002).

2.3 Segurança versus Usabilidade

Durante o projeto e desenvolvimento de sistemas que possuam requisitos de segurança, é importante prever, observar e minimizar os efeitos colaterais possivelmente causados pela adoção dos métodos supracitados. Sistemas que tentam aumentar seu nível de segurança tendem a diminuir a usabilidade dos mesmos (YEE, 2004). Um exemplo simples são os sistemas de serviços de bancos *online*, que por lidarem com transações financeiras, apresentam requisitos de segurança como prioridades em seus sistemas. Esses sistemas, em sua maioria, se mostram de difícil utilização devido às inúmeras etapas de autenticação necessárias para completar uma operação.

É, portanto, necessário que os projetos de sistemas definam requisitos tanto de segurança quanto de usabilidade de forma a identificar as suas prioridades, encontrando o equilíbrio que levará a aceitação do sistema por parte dos usuários.

2.4 Considerações Finais

Este capítulo apresentou os conceitos relevantes sobre segurança da informação assim como os métodos de implantação de serviços de segurança mais utilizados. Este arcabouço de soluções fornece subsídios para a implantação de sistemas seguros se combinados de maneira adequada. Todavia, como dito anteriormente, as soluções criptográficas apresentadas se baseiam na existência de um ou mais segredos confiáveis entre os interlocutores. Desta forma, de acordo com esses métodos, o sistema deixa de ser seguro caso haja comprometimento do segredo confiável de uma das partes envolvidas.

A premissa da existência de um segredo confiável e de que o mesmo não será acessado por terceiros, se mostra especialmente forte e de difícil cumprimento quando estes segredos se tratam de senhas geradas por interlocutores humanos e que devem ser mantidas em sigilo pelos mesmos. Evidencia-se, portanto, a importância de sistemas que minimizem o impacto causado pelo comprometimento dos segredos confiáveis.

O próximo capítulo descreve sistemas de gerenciamento de identidade, uma abordagem de sistema voltada à usabilidade, que, como já discutido, tende a agravar vulnerabilidades de segurança.

3 SISTEMAS DE GERENCIAMENTO DE IDENTIDADE E OPENID

Este capítulo tem como objetivo apresentar os sistemas de gerenciamento de identidade, em especial o sistema *OpenID*, dando enfoque às suas características de usabilidade e os benefícios que a abordagem traz a usuários do ambiente *Web*. A introdução de conceitos como a de ambientes *Web* centrados em usuários é essencial à compreensão deste trabalho.

3.1 Sistemas de Gerenciamento de Identidade

Um sistema de gerenciamento de identidades (SGI) tem o objetivo de fornecer uma forma de gerenciar identidades de usuários através de diferentes serviços durante o ciclo de vida de uma identidade (TRACY, 2008).

Uma aplicação deste tipo de sistema é o gerenciamento de identidades dentro de organizações como empresas ou universidades, sendo uma forma de controlar o acesso aos seus serviços de acordo com mudanças de cargos, no caso de empresas, ou com chegadas de novos alunos e saídas de alunos formados, no caso das universidades (TRACY, 2008).

Com o crescimento da gama de serviços disponíveis no ambiente *Web*, e com a maior demanda de novos usuários a cada dia, se fez necessário gerenciar as identidades *online* dos usuários nesses serviços.

Na *Web*, o modelo mais comum de gerenciamento de identidades é do tipo *Service Provider Centric*, ou seja, centrado no provedor de serviço (POPE; JOSANG, 2005). Neste modelo, o provedor de serviços é responsável por armazenar e gerenciar as informações de todos os seus usuários. Desta forma, se um novo usuário desejar utilizar tal serviço, esse deve fornecer um conjunto de informações para criar uma nova identidade naquele provedor de serviço. Da mesma forma, caso este usuário deseje agora acessar um outro serviço, deverá repetir o processo para o novo provedor. A Figura 1 mostra este modelo, chamado de *site-centric.*

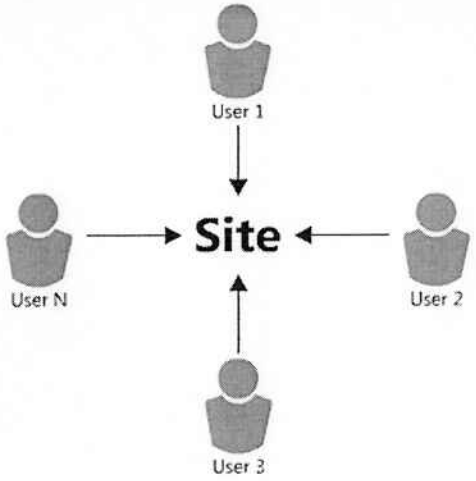

Figura 1 - Ambiente centrado em sites (*site-centric*)

Visando fornecer mais flexibilidade ao usuário e eliminar a necessidade da criação de múltiplas contas para os serviços que este utiliza, surge o modelo de gerenciamento de identidade *user-centric*, ou seja, centrado no usuário. Baseado no modelo de identificação do mundo físico, onde cada indivíduo possui uma identidade que é apresentada para diferentes fins, o modelo *user-centric* propõe que o usuário possua identidades que possam ser gerenciadas de modo centralizado e apresentadas para os serviços que desejar utilizar (POPE; JOSANG, 2005), eliminando assim a necessidade de memorizar e gerenciar contas para todos os serviços utilizados. A Figura 2 ilustra este cenário.

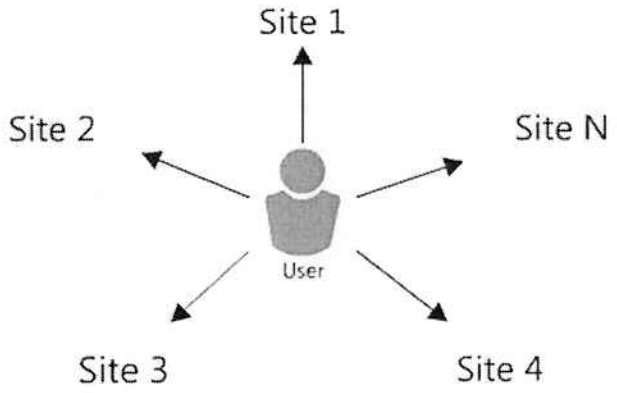

Figura 2 - Ambiente centrado no usuário (*user-centric*)

3.2 Tecnologias

Existem diversas tecnologias que se propõem a fornecer sistemas de gerenciamento de identidade, entre elas pode-se citar: XRI/XRD (LABALME; LINDELSEE; WACHOB, 2005), ID/WSF Liberty Alliance (ALLIANCE, 2003), Shibboleth (MORGAN *et al.*, 2004), Microsoft CardSpace (MALINEN, [S.d.]), Facebook Connect (STONE, 2008) e *OpenID* (RECORDON; FITZPATRICK, 2007).

Dentre as tecnologias citadas, as mais utilizadas no ambiente *Web* são o Facebook Connect e o *OpenID.* As outras tecnologias não são popularmente adotadas devido à complexidade, padrões fechados e problemas de usabilidade (DHAMIJA; DUSSEAULT, 2008).

Dentre esses dois padrões mais populares optou-se pelo uso do sistema *OpenID* , que apresenta como vantagem o fato de ser um padrão aberto além de sua adoção estar em crescimento (KISSEL, 2009).

3.3 OpenID

O *OpenID*[4] é um protocolo de autenticação que visa o fortalecimento da identidade digital do usuário, colocando em prática, assim, a abordagem de gerenciamento de identidades *user-centric.*

A identidade digital *OpenID* deve ser obtida em um provedor de identidade no ambiente *Web*. Neste provedor, uma conta é criada com as informações do usuário, tais como nome, idade, e-mail e endereço, e a esta conta é associada uma URL (e.g.: www.exemplo.com/username) com a qual o usuário poderá acessar os serviços que suportem *OpenID*. Como exemplos de provedores de identidade na *Web*, pode-se citar

[4] www.openid.net

Google[5] e Yahoo[6]. A Figura 3 ilustra o exemplo de um serviço no qual o usuário pode utilizar sua conta em diferentes provedores de identidade para se autenticar.

Figura 3 - Serviço *Web* (*http://sonora.terra.com.br*) que permite a utilização de diferentes provedores para o acesso

3.3.1 Componentes

A arquitetura do *OpenID* possui três entidades principais que são: o navegador do usuário, localizado no cliente; o provedor de identidade *OpenID* (OP); e a *Relying Party* (RP), que reside nos servidores do serviço acessado (RECORDON; FITZPATRICK, 2007).

- Provedor de Identidade: O provedor de identidade é o serviço responsável por fornecer um identificador *OpenID* e prover a autenticação *OpenID* com os serviços que a requisitarem.
- Navegador do Usuário: O processo de autenticação é iniciado no navegador do usuário, onde este tentará acessar um serviço que possua suporte ao *OpenID*. O serviço se comunica com o provedor de identidade *OpenID* requisitando a autenticação e as informações iniciais serão inseridas por meio do navegador.
- Relying Party: A *Relying Party*, ou parte confiável, é o serviço que o usuário deseja acessar, o qual necessita da autenticação *OpenID* e obtenção dos

[5] www.google.com

[6] www.yahoo.com

atributos referentes à identidade digital do usuário. Este serviço pode ter naturezas diversas, podendo ser desde um blog até um site de notícias ou compras.

3.3.2 Fluxo de Comunicação

A comunicação entre os componentes da arquitetura *OpenID* está representada na Figura 4 e descrito a seguir (SAKURAGUI, 2011):

1. Usuário insere seu identificador OpenID.
2. A Relying Party usa o identificador para encontrar as informações necessárias para a requisição de autenticação.
3. A Relying Party compartilha um segredo com o Provedor OpenID.
4. O usuário é redirecionado ao Provedor OpenID. Esta mensagem possui um parâmetro que indica a localização para onde o usuário deve ser redirecionado após a autenticação.
5. O usuário se autentica no Provedor OpenID.
6. O Provedor OpenID redireciona o usuário para a Relying Party, enviando um conjunto de parâmetros da identidade do usuário e uma assinatura digital.
7. A Relying Party verifica a validade da assinatura digital e garante o acesso ao serviço.

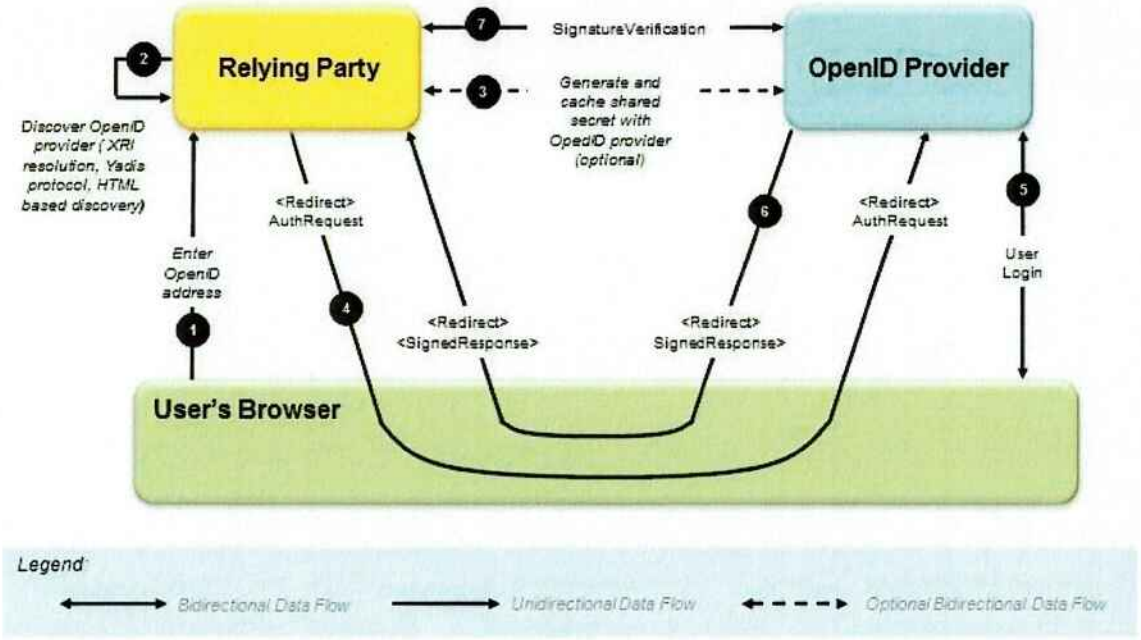

Figura 4 - Fluxo de comunicação entre componentes OpenID

3.4 Considerações sobre Sistemas de Gerenciamento de Identidade

Neste capítulo, apresentaram-se características de sistemas de gerenciamento de identidade, focando no modelo *user-centric* e expondo com mais detalhes a tecnologia *OpenID*. Como visto, este modelo se mostra fortemente focado em usabilidade, em oposição ao desenvolvimento de sistemas voltados à segurança como exposto no capítulo 1.

O próximo capítulo irá abordar e detalhar as vulnerabilidades de segurança decorrentes do foco em usabilidade proposto por este tipo de sistema. A partir dessa análise, será elaborada uma proposta de solução que busca conciliar melhorias nos aspectos de segurança com o avanço em usabilidade trazido pelos sistemas de gerenciamento de identidade centrados em usuário.

4 O SISTEMA MOBILEAUTH

Neste capítulo, serão abordadas vulnerabilidades dos sistemas apresentados no capítulo anterior com base em conceitos de segurança discutidos no capítulo 1. A partir dessa abordagem, será apresentada a proposta do sistema MobileAuth. Com base na proposta, são apresentados os componentes do sistema. A seguir é detalhada a comunicação entre esses componentes. Por fim são apresentados os casos de uso e requisitos do sistema.

4.1 Vulnerabilidades Abordadas

Como já discutido no capítulo 1, sistemas que tem como prioridade a usabilidade podem prejudicar os atributos de segurança do mesmo. No caso dos sistemas de gerenciamento de identidades baseados no modelo *user-centric*, especialmente a tecnologia *OpenID*, essa vulnerabilidade se revela por meio da centralização das informações no usuário, o que torna todos os serviços do usuário vulneráveis mediante um comprometimento da identidade *OpenID* do mesmo.

O problema de segurança de um sistema centrado no usuário no caso do comprometimento da senha de acesso está ilustrado na Figura 5. No possível cenário de haver um comprometimento de identidade, quando em um ambiente *site-centric*, somente aquele serviço será comprometido, já que as identidades para os diversos serviços são independentes, como já discutido no capítulo 3. Por outro lado, caso aconteça a mesma situação em um ambiente *user-centric*, como a identidade do usuário é a mesma para todos os serviços, além de ser o centro de controle de acesso aos mesmos, o comprometimento do segredo de acesso implica diretamente no comprometimento de todos os serviços associados àquele usuário.

Aliado à maior abrangência do segredo de acesso à conta mantido pelo usuário, a própria característica construtiva deste segredo é pouco confiável, pois, como já exposto, senhas geradas por usuários apresentam limitações em sua complexidade e tamanho devido à necessidade de memorização. Assim, grande parte dos usuários opta por senhas consideradas fracas, o que facilita desde ataques de força bruta até ataques

baseados em informações pessoais obtidas por outros meios que são comumente utilizadas para compor senhas de fácil memorização.

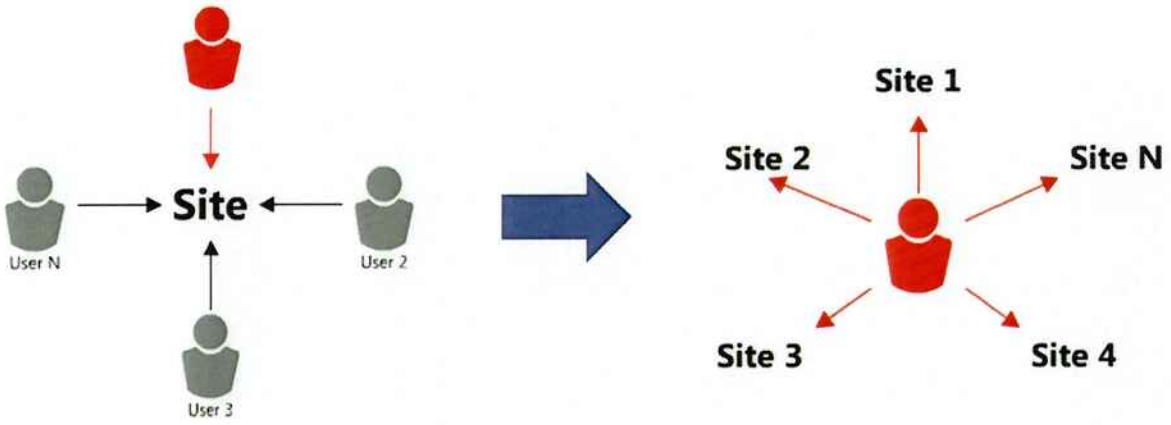

Figura 5 - Comprometimento de identidade nos casos *site-centric* e *user-centric*

4.2 Proposta de sistema

Com base nas situações expostas, propõe-se uma solução que fortaleça a segurança de acesso do ambiente *OpenID*. Deseja-se, portanto, construir um sistema que mantenha a usabilidade e o enfoque aos usuários relacionados com os sistemas de gerenciamento de identidades *user-centric*, porém de forma a inserir resistências a ataques de obtenção indevida de senha.

É importante destacar que o sistema a ser proposto, apresenta uma aderência a cenários de segurança não crítica, ou seja, cenários em que o valor da obtenção da informação é em geral pequeno. Em outras palavras, não está dentro do escopo desse trabalho reduzir as vulnerabilidades do sistema frente a ataques criptográficos elaborados, que são geralmente organizados para a obtenção de valores monetários, independente da identidade do alvo (e.g. ataques contra sistemas bancários).

O sistema MobileAuth se encaixa portanto em cenários de gerenciamento de identidade para *blogs*, contas de e-mail, redes sociais e outros sistemas em que a relevância das informações armazenadas pelo serviço está ligada à identidade do usuário.

O sistema MobileAuth estende a arquitetura *OpenID* padrão, mostrada na Figura 6, por meio da adição de uma aplicação móvel, instalada em um smartphone, que deverá complementar a autenticação do usuário. A nova arquitetura está ilustrada na Figura 7.

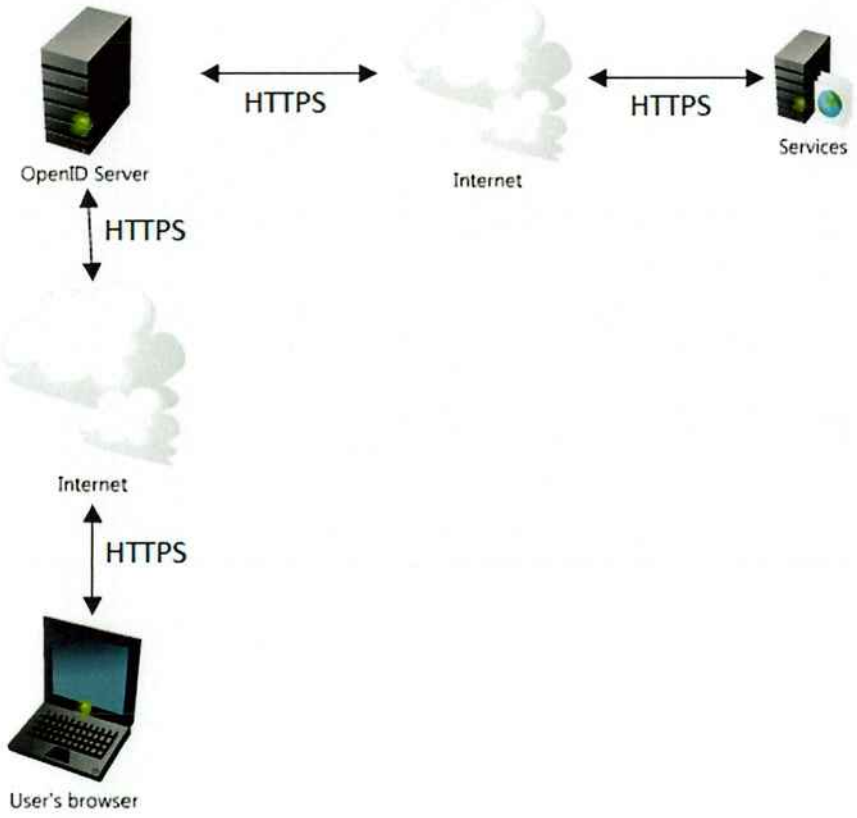

Figura 6 - Arquitetura OpenID padrão

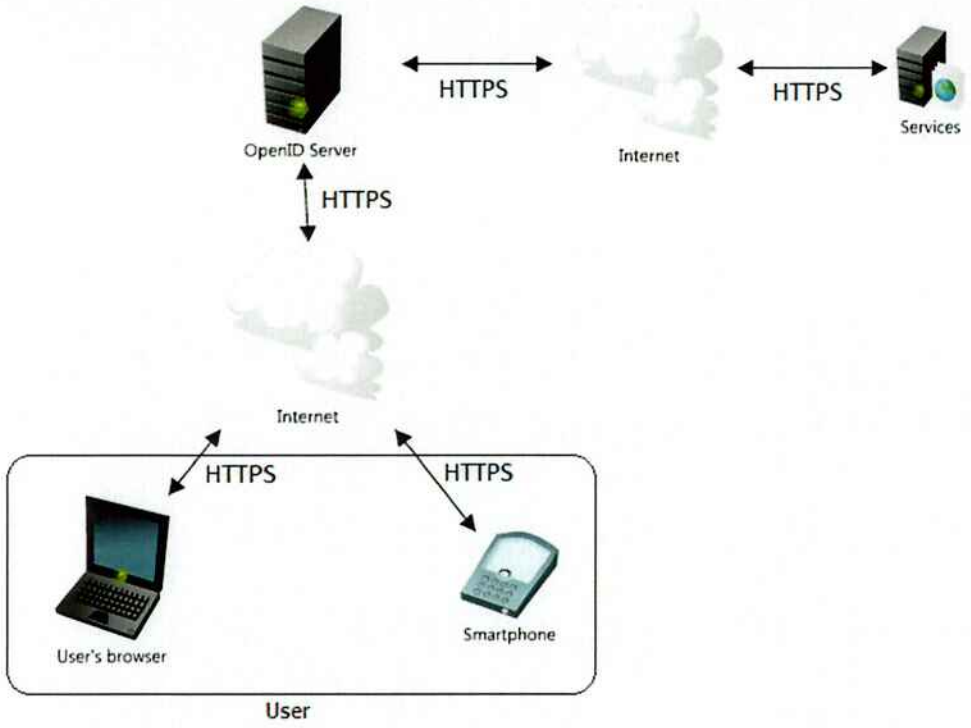

Figura 7 - Arquitetura MobileAuth

Na arquitetura proposta, o usuário passa a interagir com o sistema por dois meios, além de utilizar o seu navegador *Web* como na arquitetura padrão, agora será utilizado um *smartphone* que será responsável por enviar informações adicionais que agirão como uma autenticação adicional do usuário com o provedor de identidade.

4.3 Componentes

Além dos três componentes principais da tecnologia OpenID, no MobileAuth adiciona-se um componente: o *smartphone* do usuário, como já comentado. É relevante destacar que os componentes da arquitetura padrão *OpenID* utilizados para o desenvolvimento do sistema MobileAuth já foram implantados em outros projetos, portanto cabe a este trabalho especificar e desenvolver o novo módulo, bem como realizar sua integração com a arquitetura já existente. A seguir encontra-se a descrição dos componentes no novo sistema.

4.3.1 Navegador do usuário e Relying Party

Como foi visto na seção 3.3.1, dentre os componentes da arquitetura *OpenID* padrão estão o navegador do usuário e a *relying party*.

O navegador do usuário serve como interface para a comunicação entre usuário e sistema. É nele que o usuário iniciará a autenticação de sua identidade, que fará o disparo para o provedor de identidade realize todas as ações necessárias para garantir o acesso ao usuário. No novo sistema, o navegador ainda possui a mesma função de simplesmente ser o meio de interação entre usuário e serviços ou provedor de identidade. Portanto, no MobileAuth este item se mantém inalterado.

Da mesma forma, a relying party também não sofrerá alterações no novo sistema. A RP consiste no serviço que o usuário deseja acessar, a qual deve se comunicar com o provedor de identidade *OpenID* para solicitar a autenticação do usuário. Como o fluxo de comunicação *OpenID*, apresentado na seção 3.3.2 se mantém inalterado, consequentemente não haverá modificações na RP e esta continuará se comunicando com o provedor de identidade da maneira convencional.

4.3.2 Provedor de Identidade

O provedor de identidade *OpenID* é o responsável por emitir os identificadores e prover a autenticação com os serviços, como já mencionado no capítulo 3.

Como discutido, será utilizado um provedor de identidade já existente para a realização do projeto do sistema MobileAuth, desenvolvido pelo Laboratório de Arquitetura e Redes de Computadores da Escola Politécnica. Atualmente este provedor suporta autenticação por meio de nome de usuário e senha, e armazena as informações de identidade de usuário do padrão SREG, são elas:

- Nome;
- Nome completo;
- Data de nascimento;

- Gênero;
- Linguagem;
- E-mail;
- Telefone;
- Página web;
- Endereço (Endereço, Código Postal, Cidade, Estado, País);
- Fuso horário.

O provedor será modificado para que se possa associar um *smartphone* a uma conta *OpenID*, e para realizar todos os procedimentos de comunicação entre aplicativo e servidor. Detalhes de funcionalidades serão descritos em seções posteriores.

4.3.3 Aplicação Móvel

Foi desenvolvida uma aplicação para um *smartphone* que será a responsável pela confirmação de acesso de um usuário ao provedor de identidade. Esta escolha de plataforma se deve primeiramente ao caráter pessoal e intransferível que os aparelhos celulares culturalmente assumem.

A mobilidade dessa plataforma também exerce papel fundamental nessa escolha para o sistema MobileAuth, pois ao estabelecer que a autenticação adicional será feita mediante a posse de um dispositivo, é essencial garantir que essa necessidade não represente um incômodo para o usuário, o que comprometeria a usabilidade e aceitação do sistema. Por fim, destaca-se o crescimento do uso de dispositivos móveis para acesso a *Internet*, consolidando as possibilidades de aceitação deste trabalho (FLOSI, 2011).

A aplicação armazenará dados do *smartphone* do usuário, de forma que a conta *OpenID* só consiga ser acessada por um único *smartphone* e único *SIM Card*. Para isso serão usadas duas informações:

- IMEI: código de 15 dígitos único para cada dispositivo móvel.
- IMSI: código de 15 dígitos único para cada *SIM Card*.

Esse procedimento tem por objetivo associar não só o aparelho do usuário, mas também o *SIM Card* que determina sua conta de acesso ao serviço de telecomunicações, fortalecendo assim o caráter pessoal e intransferível da autenticação adicional proposta.

A aplicação fornecerá três níveis de autenticação de usuário, baseados nas três informações necessárias para tornar um acesso seguro (JAIN; FLYNN; ROSS, 2007):

- O que o usuário possui.
- O que o usuário sabe.
- O que o usuário é.

Portanto, a aplicação móvel fornecerá os níveis de autenticação baixo, médio e alto:

- Nível baixo: O nível mais baixo de segurança se baseará na simples posse do *smartphone* com o *SIM Card* definidos no momento em que o *smartphone* é associado à conta do usuário no provedor de identidade.
- Nível médio: Neste nível, o usuário deverá inserir uma senha também definida no momento da associação.
- Nível alto: O nível mais alto deve consistir em uma informação mais forte do que a simples inserção de uma senha, como no nível anterior, portanto uma proposta para este nível é a utilização de sistemas de identificação biométrica.

Para o nível alto de autenticação, é utilizada a biblioteca *Gesture* do Android, exposta na seção 3.2. Como a proposta consiste na utilização de identificação biométrica, primeiramente foi analisada a possibilidade de utilizar reconhecimento de impressões digitais, porém devido à existência de poucos dispositivos com suporte a este tipo de reconhecimento e ao alto preço dos mesmos, essa possibilidade foi deixada de lado. Para substituir o reconhecimento de digitais, surgiu a nova possibilidade da utilização de assinaturas utilizando a tela sensível ao toque do *smartphone*, porém não havia disponível nenhuma biblioteca que tratasse esse reconhecimento e não seria viável realizar o desenvolvimento dessa função, devido ao tempo disponível e ao escopo do trabalho. Portanto, decidiu-se utilizar para o nível alto a inserção de um gesto, que representa uma assinatura biométrica.

A escolha dos níveis de segurança será feita no momento do primeiro acesso a determinado serviço. O provedor de identidade solicitará que o usuário defina qual nível deverá ser usado para o serviço em particular. O usuário deverá necessariamente escolher uma das três opções para o serviço, não sendo possível a não utilização da aplicação móvel em qualquer acesso. O acesso à conta no provedor de identidade para gerenciamento será sempre realizado com o nível alto de autenticação.

As informações cadastradas dos níveis de segurança, as informações referentes ao *smartphone* e informações de segredos confiáveis deverão ser armazenadas em arquivos armazenados na área do smartphone reservada à aplicação, ou seja, em uma área da memória que não pode ser acessada por outras aplicações.

Essa modelagem de sistema com níveis distintos de autenticação permite uma maior versatilidade quanto às relações entre usabilidade e segurança. Ao utilizar o mesmo mecanismo de gerenciamento de identidade para todos os seus serviços, o *OpenID* impõe o uso de um determinado nível de usabilidade e segurança, que pode ser adequado a um determinado serviço, mas se mostrar insatisfatório para outro.

A existência dos três níveis de segurança permite que o usuário opte por um enfoque em segurança para um serviço que este julgue pertinente, ao mesmo tempo permite que o usuário escolha uma abordagem voltada a usabilidade para um serviço que considere de segurança não crítica.

4.4 Comunicação

Embora o MobileAuth tenha como objetivo estender a segurança do sistema para casos em que a senha de usuário está potencialmente comprometida, é importante que técnicas de segurança sejam utilizadas na comunicação entre a aplicação móvel e o provedor de identidade, impedindo que a informação seja interceptada ou modificada. Para garantir a segurança na comunicação, serão utilizadas criptografia simétrica, protocolo HTTPS com certificado digital e criptografia assimétrica, descritos no capítulo 1.

O MobileAuth gera as chaves simétricas a partir de dados pessoais (IMEI, IMSI) e números pseudo-aleatórios.

Para o certificado digital, devido à restrição de recursos, e do caráter demonstrativo do projeto, será utilizada uma entidade certificadora particular que, consequentemente, só será reconhecida pelos módulos utilizados na demonstração. O certificado utilizará o padrão X.509[7], utilizado pelas autoridades certificadoras, e será instalado tanto no navegador do usuário como em seu smartphone.

O método proposto, apresentado a seguir, foi fortemente inspirado no protocolo TLS (Transport Layer Security), que utiliza um canal de criptografia assimétrica para a derivação de chaves simétricas entre os interlocutores (MICROSOFT TECHNET, [S.d.]). Essa escolha se dá por simplicidade de implantação e pelo fato de que o uso de um modelo já existente e fortemente aceito possibilita a construção de um sistema criptograficamente sólido.

4.4.1 Etapas da comunicação

A comunicação do sistema MobileAuth ocorre entre o Provedor OpenID e a Aplicação Móvel.

O fluxo da comunicação entre os dois componentes visto de forma macroscópica, pode ser representado em quatro passos, que podem ser observados na Figura 8.

1. Ao ser ativada, a aplicação móvel envia uma requisição inicial para o Provedor *OpenID.*

2. Em seguida, ocorre o fluxo de autenticação do TLS[8], disparado pela requisição inicial, garantindo a segurança do canal de comunicação.

---

[7] Vide anexo 3.

[8] A descrição do protocolo TLS é apresentada no anexo 1.

3. A partir disso, é realizada a troca das mensagens necessárias para a definição de qual procedimento será realizado: cadastro ou autenticação.

4. Como última etapa é realizada a confirmação de autenticidade para a confirmação do procedimento. As sub-seções seguintes apresentarão as etapas desta confirmação.

Figura 8 - Fluxo geral de comunicação

4.4.1.1 Segredo Compartilhado

Antes de definir-se o método de confirmação de autenticidade da aplicação móvel, é preciso definir a geração do segredo compartilhado necessário a esta etapa.

Devido à proposta dos níveis de segurança apresentados na seção 4.3.3, é necessário obter dois segredos diferentes. Isso se deve ao primeiro nível de segurança especificado, a confirmação ilustrando o caso da senha com uma informação que o usuário possui. Para este nível de segurança, como não temos nenhuma informação de senha associada, não existe uma informação para usar como chave para proteger o arquivo, fazendo necessário manter o arquivo em texto claro.

Como a partir do segundo nível é necessária a inserção de uma senha, esta pode ser utilizada para proteger o arquivo onde se encontra o segredo compartilhado.

A adoção da separação dos dois segredos também ilustra a menor segurança no caso do primeiro nível, já que o segredo é mantido aberto, enquanto nos outros o segredo se encontra cifrado.

4.4.1.2 Método de Confirmação de Autenticidade

Para a verificação de autenticidade da confirmação de acesso enviada pelo smartphone foi especificado um método de troca de mensagens baseado no fluxo do TLS.

É utilizada a criptografia simétrica para realizar um teste de autenticidade. A partir de uma chave simétrica previamente compartilhada entre as duas partes do sistema, é realizado o seguinte teste de verificação:

1. O servidor produz um número pseudo-aleatório.
2. O servidor encripta o número gerado utilizando como chave um segredo previamente compartilhado entre as duas partes da comunicação. Esse segredo é gerado no momento em que se associa um *smartphone* à conta do usuário, ou seja, na primeira vez que provedor de identidade e aplicação móvel se comunicam.
3. O servidor envia a mensagem encriptada por meio de uma resposta a uma requisição pendente da aplicação móvel.
4. Ao receber esta resposta, a aplicação móvel a decripta utilizando como chave o segredo compartilhado.
5. Ao número obtido da decriptação é somado o valor 1.
6. A aplicação encripta o resultado da soma.
7. A aplicação envia uma nova requisição ao servidor com o resultado da encriptação.
8. O servidor decripta a mensagem recebida e verifica se o número resultante corresponde ao valor gerado por ele anteriormente e acrescido de um.

Utilizando este método, garante-se que a confirmação de acesso só é realizada pela aplicação móvel devidamente associada à conta do usuário, que possui o segredo compartilhado correto e necessário para realizar as etapas de criptografia.

4.5 Casos de Uso e Requisitos do Sistema

Neste capítulo serão descritas as funcionalidades do sistema por meio da apresentação de casos de uso e requisitos.

4.5.1 Casos de Uso

Nesta seção estão apresentados os casos de uso que contemplam as funcionalidades que o sistema deve cumprir.

Caso de Uso 1 – Cadastro: Primeira etapa

Descrição: Este caso de uso descreve a primeira etapa do cadastro de um usuário no provedor de identidade.

Pré-Condição: Não há.

Sequência de Eventos:

1. Usuário acessa o Provedor de Identidade *OpenID* por meio de seu navegador e clica no *link* de cadastro.
2. Usuário preenche formulário de cadastro.
3. Sistema fornece ao usuário instruções para *download* da aplicação móvel.

Pós-Condição: Os dados do usuário e a chave identificadora estão armazenados no sistema.

Caso de Uso 2 – Cadastro: Segunda etapa

Descrição: Este caso de uso descreve a segunda etapa do cadastro de um usuário no provedor de identidade.

Pré-Condição: Usuário já realizou a primeira etapa do cadastro no provedor.

Sequência de Eventos:

1. Usuário faz *download* e instala a aplicação móvel.
2. A aplicação obtém e armazena as informações do smartphone.
3. A Aplicação fornece instruções de uso.
4. Usuário insere senha referente ao nível médio de segurança.
5. Aplicação armazena esta informação.
6. Usuário insere seu gesto, que representa uma assinatura biométrica, na tela sensível ao toque para o nível alto de segurança.
7. Aplicação armazena esta informação.
8. Aplicação envia requisição ao Provedor de Identidade para realizar associação do smartphone à conta do usuário.
9. Provedor de Identidade associa o smartphone.
10. Provedor de Identidade notifica o usuário da conclusão do cadastro.

Pós-Condição: Efetuada associação do *smartphone* à conta do usuário e cadastro concluído.

Caso de Uso 3 – Autenticação em serviço

Descrição: Este caso de uso descreve a autenticação em um serviço.

Pré-Condição: Usuário cadastrado no Provedor de Identidade com aplicação móvel instalada em seu *smartphone.*

Sequência de Eventos:

1. Usuário insere seu identificador *OpenID* no serviço para autenticação.
2. Serviço redireciona usuário ao Provedor *OpenID.*
3. Usuário insere seu nome de usuário e senha.
4. Provedor notifica o usuário que a aplicação móvel deve ser acionada.

5. Aplicação móvel se comunica com o Provedor requisitando o nível de autenticação que deve realizar.
6. Provedor responde informando o nível de segurança necessário para o acesso a este serviço.
7. Usuário insere informações de autenticação na Aplicação Móvel.
8. Fluxo de confirmação de autenticidade.
9. Provedor libera acesso ao usuário.

Pós-Condição: Usuário autenticado no serviço desejado.

Extensões:

1. Primeira vez que usuário tenta acessar este serviço: Provedor fornece opções de nível de segurança para usuário escolher (Passo 4).
2. Usuário insere informações erradas de autenticação: Aplicação notifica o usuário e não envia requisição ao Provedor (Passo 7).
3. Aplicação não está autorizada e Provedor nega o acesso (Passo 9).

Caso de Uso 4 – Acesso ao Provedor de Identidade

Descrição: Este caso de uso descreve o acesso à conta de usuário no Provedor de Identidade.

Pré-Condição: Usuário cadastrado.

Sequência de Eventos:

1. Usuário acessa Provedor de Identidade por meio de seu navegador.
2. Usuário insere nome de usuário e senha.
3. Provedor notifica o usuário que a aplicação móvel deve ser acionada.
4. Aplicação envia requisição ao provedor solicitando o nível de segurança.
5. Provedor responde enviando o nível alto de segurança.

6. Usuário insere senha e gesto na aplicação móvel.
7. Fluxo de confirmação de identidade
8. Provedor exibe informações da conta do usuário.

Pós-Condição: Usuário tem acesso à sua conta.

Extensões:

1. Usuário insere informações erradas de autenticação: Aplicação notifica o usuário e não envia requisição ao Provedor (Passo 6).
2. Aplicação não está autorizada e Provedor nega o acesso (Passo 8).

Caso de Uso 5 – Atualização de informações no Provedor de Identidade

Descrição: Este caso de uso descreve a atualização de informações da conta do usuário no Provedor de Identidade.

Pré-Condição: Usuário cadastrado e autenticado previamente no Provedor utilizando a aplicação móvel.

Sequência de Eventos:

1. Usuário altera informações de um conjunto de dados de sua conta e clica em Salvar.
2. Provedor conclui a atualização das informações.

Pós-Condição: Informações de conta de usuário alteradas.

Caso de Uso 6 - Atualização de informações na aplicação móvel

Descrição: Este caso de uso descreve a atualização de informações da aplicação móvel.

Pré-Condição: Usuário cadastrado e com a aplicação instalada no *smartphone*.

Sequência de Eventos:

1. Usuário escolhe opção de alteração de informações de autenticação na aplicação móvel.
2. Aplicação exibe opções de alteração: senha ou gesto.
3. Usuário escolhe a opção desejada.
4. Aplicação solicita que usuário insira a informação atual para realizar a atualização.
5. Usuário insere a informação.
6. Aplicação exibe campo para inserção da nova informação.
7. Usuário insere nova informação.

Pós-Condição: Informação de autenticação alterada.

Caso de Uso 7 – Perda do *smartphone*

Descrição: Este caso de uso descreve o procedimento caso o usuário perca seu *smartphone*.

Pré-Condição: Usuário cadastrado.

Sequência de Eventos:

1. Usuário acessa o Provedor de Identidade e clica no link de perda de celular.
2. Provedor notifica o usuário que será enviado um e-mail e solicita a confirmação da ação.
3. Usuário confirma a ação.
4. Provedor envia um e-mail para o usuário contendo um link que deve ser acessado para completar o congelamento da aplicação móvel.
5. Usuário acessa o link informado.
6. Provedor exibe um código para acesso alternativo à conta e manda o mesmo por e-mail.

Pós-Condição: Uso da aplicação móvel congelado.

Extensão:

1. Código expira após cinco dias caso não seja associado um novo smartphone neste período (Passo 4).

Caso de Uso 8 – Revogação

Descrição: Este caso de uso descreve o procedimento para a revogação da conta do usuário em caso de desistência do serviço.

Pré-Condição: Usuário cadastrado e autenticado no provedor de identidade.

Sequência de Eventos:

1. Usuário clica no *link* de revogação de conta.
2. Provedor notifica usuário que a ação não pode ser desfeita e solicita que a aplicação móvel seja acionada para a confirmação.
3. Aplicação móvel notifica Provedor que está pronta.
4. Provedor responde informando o nível de segurança alto para realizar a confirmação.
5. Usuário confirma inserindo as informações de autenticação do nível alto.
6. Provedor exclui conta do usuário.
7. Usuário pode desinstalar aplicação móvel de seu *smartphone.*

Pós-Condição: O usuário não possui conta no sistema.

Caso de Uso 9 – Acesso alternativo aos serviços

Descrição: Este caso de uso descreve o acesso no caso de conta com funcionalidades limitadas por perda do *smartphone.*

Pré-Condição: Usuário solicitou congelamento do uso da aplicação móvel.

Sequência de Eventos:

1. Usuário insere seu identificador *OpenID* no serviço para autenticação.
2. Serviço redireciona usuário ao Provedor *OpenID*.
3. Usuário insere seu nome de usuário e senha.
4. Provedor solicita a inserção do código de acesso alternativo.
5. Usuário insere o código.
6. Provedor libera acesso ao serviço.

Pós-Condição: Usuário autenticado no serviço desejado.

Caso de Uso 10 – Alteração de *smartphone*

Descrição: Este caso de uso descreve o procedimento para alterar o *smartphone* associado à conta do usuário.

Pré-Condição: Usuário cadastrado e autenticado no provedor de identidade.

Sequência de Eventos:

1. Usuário seleciona opção de alteração de smartphone associado.
2. Provedor solicita confirmação da ação.
3. Usuário confirma.
4. Provedor notifica o usuário sobre a ativação da aplicação móvel no novo *smartphone.*
5. Usuário realiza cadastro da aplicação no novo smartphone.
6. O provedor associa o novo smartphone à conta.

Pós-Condições: O *smartphone* associado à conta foi substituído.

4.5.2 Requisitos Funcionais

Esta seção apresenta os requisitos funcionais para o provedor de identidade e para a unidade móvel de autenticação.

4.5.2.1 Provedor de identidade

1. Permitir associação de um *smartphone* a uma conta;
2. Gerar chaves únicas a partir de IMEI, IMSI e número aleatório;
3. Permitir que o usuário escolha um nível de segurança para cada serviço acessado;
4. Armazenar informação de nível de segurança relativo a cada serviço;
5. Responder requisições vindas do *smartphone*;
6. Verificar validade das mensagens vindas do *smartphone*;
7. Exibir informações de conta de usuário relativas a informações pessoais, serviços, níveis de segurança e *smartphone* associado;
8. Permitir alteração das informações de conta do usuário;
9. Permitir alteração de *smartphone* associado
10. Fornecer procedimento de autenticação alternativa no caso de congelamento da aplicação móvel;
11. Fornecer código para autenticação alternativa no caso de congelamento da aplicação móvel;
12. Exibir mensagens de notificação ao usuário;
13. Permitir encerramento de conta de usuário;
14. Enviar *e-mails* de notificação;
15. Liberar o acesso.

4.5.2.2 Unidade móvel de autenticação

1. Disparar requisições *Web* de forma segura;
2. Apresentar uma tela de instruções de uso;
3. Armazenar as informações sigilosas do usuário de forma segura;
4. Permitir que o usuário cadastre senha e gesto;
5. Permitir que o usuário modifique senha e gesto;
6. Permitir que o usuário autorize pedidos de confirmação;
7. Enviar mensagem ao provedor de identidade comprovando a autenticidade do usuário.

4.5.3 Requisitos Não-Funcionais

Os requisitos não-funcionais que o sistema deve cumprir são:

1. O *smartphone* deve possuir sistema operacional Android 2.1 ou mais recente;
2. O *smartphone* deve possuir tela de 2.8 polegadas com resolução de 240 x 320 pixels, devido a restrições;
3. O *smartphone* deve possuir tela sensível ao toque; devido à necessidade de inserção de gesto;
4. O *smartphone* deve suportar acesso à *Internet*;
5. A comunicação deve ser realizada utilizando HTTPS.

Devido à falta de maiores recursos disponíveis, a aplicação móvel será desenvolvida para o *smartphone* de um dos membros do grupo, impondo restrições de tamanho da tela e versão do sistema operacional.

4.6 Considerações sobre o sistema MobileAuth

A especificação do sistema MobileAuth foi desenvolvida de acordo com os objetivos iniciais descritos no capítulo 1. Através da inserção de um módulo adicional de autenticação pretende-se reduzir os riscos associados à adoção de um sistema de gerenciamento de identidade para o uso de serviços na *Internet*. Esse módulo apresenta, como descrito nesse capítulo, diferentes níveis de segurança para que o usuário opte pela relação entre usabilidade e segurança que considere mais adequada.

Este capítulo apresentou de forma macroscópica como o sistema MobileAuth pretende alcançar os objetivos propostos através da descrição dos componentes do sistema e suas funcionalidades. Foram apresentados também aspectos da comunicação entre os componentes descritos de forma a possibilitar o estabelecimento de um relacionamento de confiança mútua entre o provedor de identidade e a aplicação móvel de autenticação.

No próximo capítulo serão apresentados detalhes internos dos componentes aqui descritos, com especial enfoque nos fluxos relativos ao cadastro e a autenticação da aplicação móvel com o provedor de identidade.

5 PROTÓTIPO DESENVOLVIDO

Este capítulo tratará do desenvolvimento do projeto do sistema MobileAuth, especificado no capítulo 4. Serão apresentadas as tecnologias auxiliares e as etapas de desenvolvimento, envolvendo desde fluxogramas de comunicação até pseudocódigos para representar os algoritmos.

Primeiramente serão apresentadas a tecnologias utilizadas com especial enfoque nas plataformas Java e Android que representam a base do desenvolvimento do sistema. Em seguida, este capítulo abordará uma visão macroscópica dos fluxos de comunicação entre os dois módulos do projeto, ou seja, o provedor de identidades e o *smartphone* do usuário. Posteriormente, são detalhados características de cada módulo do sistema. Por fim, são apresentadas as principais funcionalidades desenvolvidas.

5.1 Tecnologias Utilizadas

Como discutido no capítulo anterior, há dois módulos que devem ser desenvolvidos: provedor de identidade e aplicação móvel. O provedor de identidade será alterado a partir de um projeto já existente, portanto as tecnologias utilizadas serão legadas de tal projeto. Este provedor de identidade foi desenvolvido em linguagem Java[9], com páginas Web em formato JSP10, utilizando como intermediário o framework Struts11.

O desenvolvimento da aplicação móvel será realizado para o sistema operacional móvel Android12, como especificado nos requisitos não funcionais, na seção 4.5.3. Atualmente, o desenvolvimento de aplicações para Android só está disponível para a linguagem Java,

[9] www.java.com

[10] http://www.oracle.com/technetwork/java/javaee/jsp/index.html

[11] http://struts.apache.org/

[12] http://www.android.com/

o que acaba por facilitar o desenvolvimento já que o outro módulo do sistema também deverá ser desenvolvido nessa linguagem.

A seguir, serão brevemente descritas as tecnologias mencionadas, além de apresentar as APIs adotadas para desenvolver a aplicação Android.

5.1.1 Java

O Java é uma linguagem de programação de alto nível com paradigma de orientação a objetos. Foi desenvolvida pela *Sun Microsystems*, comprada em 2009 pela *Oracle Corporation*, em 1995. É a linguagem de programação mais utilizada atualmente, segundo estatísticas ("TIOBE Software: Tiobe Index", [S.d.]) tendo como maior concorrente a linguagem C.

Entre as características da linguagem, pode-se destacar (NOURIE, [S.d.]):

- Sintaxe simples;
- Orientação a objetos;
- Distribuição;
- Suporte a *multithread*;
- Dinamismo
- Independência de arquitetura;
- Portabilidade;
- Alta performance;
- Robustez;
- Segurança.

O Java é uma linguagem de propósito geral, podendo ser usado para desenvolver aplicações *desktop*, *web* ou sistemas embarcados.

Adicionalmente à adoção da linguagem Java, são utilizadas no projeto as tecnologias JSP e *Struts:*

- JSP: Sigla para *Java Server Pages*, o JSP é uma tecnologia para criação de páginas dinâmicas para o ambiente *Web*. A tecnologia permite mesclar código HTML com trechos de código Java para a criação de conteúdo dinâmico, como acesso a bancos de dados e leitura de parâmetros. A forma como uma página JSP deve ser codificada separa o conteúdo lógico da descrição HTML, o que proporciona flexibilidade em relação a alterações no design sem afetar a lógica ("JavaServer Pages Overview", [S.d.]).
- Apache Struts: Framework para a criação de aplicações *Web* em Java baseado no modelo MVC (*Model-View-Controller*). O *Struts* tem por objetivo separar os componentes do modelo MVC para facilitar a construção de uma aplicação *Web* ("The Apache Software Foundation", [S.d.]).

5.1.2 Android

O Android é um sistema operacional para dispositivos móveis desenvolvido pela Google, que fornece para desenvolvedores um pacote de software base para desenvolvimento de aplicações que possam ser executadas em celulares com o sistema. O pacote contém uma série de classes que envolvem funcionalidades tais como sincronização com serviços Google, utilização de GPS, câmera, acelerômetro e conexões a *Internet*, interação com teclas e tela do dispositivo, armazenamento de dados, entre outros. Também é oferecido um emulador para execução e depuração das aplicações antes de executá-las no dispositivo propriamente (SAHA, 2008).

5.1.3 Bibliotecas

Para o desenvolvimento da aplicação móvel, foram adotadas APIs (*Application Programming Interface*) para auxiliar no cumprimento das funcionalidades desejadas. As duas bibliotecas principais são:

- Apache HTTP: Biblioteca oferecida pela organização Apache[13] que permite lidar com requisições HTTP. Com esta biblioteca é possível criar requisições especificando seu tipo (e.g.: POST), adicionar parâmetros à requisição, executá-la e obter sua resposta e parâmetros da mesma.
- Android Gesture: Biblioteca oferecida junto ao *kit* de desenvolvimento do Android que permite tratar movimentos de gestos feitos na tela sensível ao toque dos dispositivos.

5.2 Fluxos de Comunicação

Esta seção apresentará uma visão de alto nível dos principais fluxos entre os dois módulos do MobileAuth. Os fluxos detalhados serão: fluxo de cadastro e fluxo de autenticação, este último é dividido entre fluxo de acesso à conta e fluxo da *relying party*. É importante ressaltar que cada um desses fluxos é uma extensão do fluxo geral de comunicação definido na seção 4.4.1

A descrição destes fluxos exibe a troca de mensagens necessária para o estabelecimento da aplicação móvel como autenticação adicional, como proposto no capítulo 4. Os detalhes internos dos módulos serão expostos nas próximas seções.

5.2.1 Fluxo de Cadastro

O primeiro fluxo de comunicação presente no MobileAuth é o fluxo do cadastro da aplicação móvel no provedor de identidades, é importante ressaltar que um pré-requisito para a ocorrência deste fluxo é a existência de uma conta no provedor de identidades, que embora seja criada de forma semelhante à que ocorre no sistema *OpenID* padrão será detalhada na seção 5.5.1.

[13] http://www.apache.org/

O fluxo de cadastro é o estabelecimento de um dispositivo móvel como uma entidade confiável atrelada a uma conta de usuário existente. Este fluxo parte de uma situação de confiança unilateral onde a aplicação móvel confia no provedor de identidade devido à existência de um certificado digital válido. Essa situação deve ser convertida em uma relação de confiança bilateral, onde o provedor confia na aplicação devido à existência devido à existência de uma informação confiável de posse da mesma.

A Figura 9 ilustra a troca de informações entre os dois módulos. Inicialmente, a aplicação móvel envia uma mensagem ao provedor de identidade com as informações da conta do usuário já criada e com informações sobre o próprio dispositivo. As mensagens seguintes se referem ao estabelecimento do segredo confiável entre aplicação e provedor. O segredo é enviado pelo provedor e a aplicação o notifica do recebimento. A seguir deve ser realizado o fluxo de confirmação de autenticidade descrito na seção 4.4.1.1 para verificar se a aplicação realmente recebeu o segredo correto.

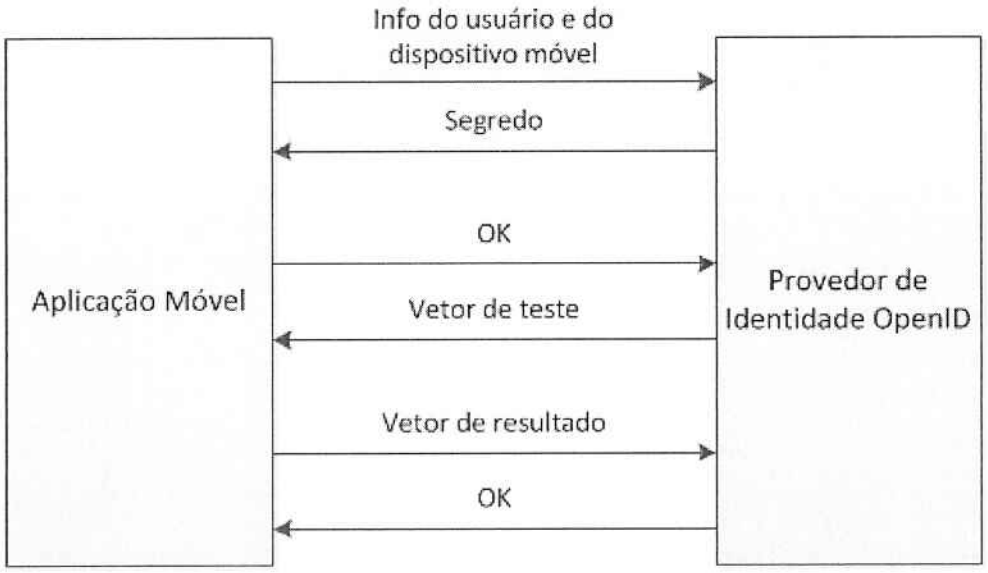

Figura 9 - Fluxo de cadastro da aplicação no provedor de identidade

5.2.2 Fluxo de Autenticação

O fluxo de autenticação acontece todas as vezes que o usuário deseja acessar sua conta ou um serviço. Neste caso, o pré-requisito necessário é já ter completado o fluxo de cadastro descrito na seção anterior.

A Figura 10 - Fluxo de autenticaçãoFigura 10 apresenta a troca de informações entre os dois módulos. Primeiramente, a aplicação envia uma mensagem ao provedor de identidade contendo informações do usuário necessárias para que o provedor identifique qual conta aquela aplicação está tentando autenticar. O provedor responde a mensagem informando qual é o nível de segurança que deve ser satisfeito para a última requisição de acesso pendente realizada pelo navegador do usuário. Após a operação de autenticação na aplicação, esta responde com uma mensagem de confirmação e é iniciado o processo de verificação de autenticidade, especificado na seção 4.4.1.1

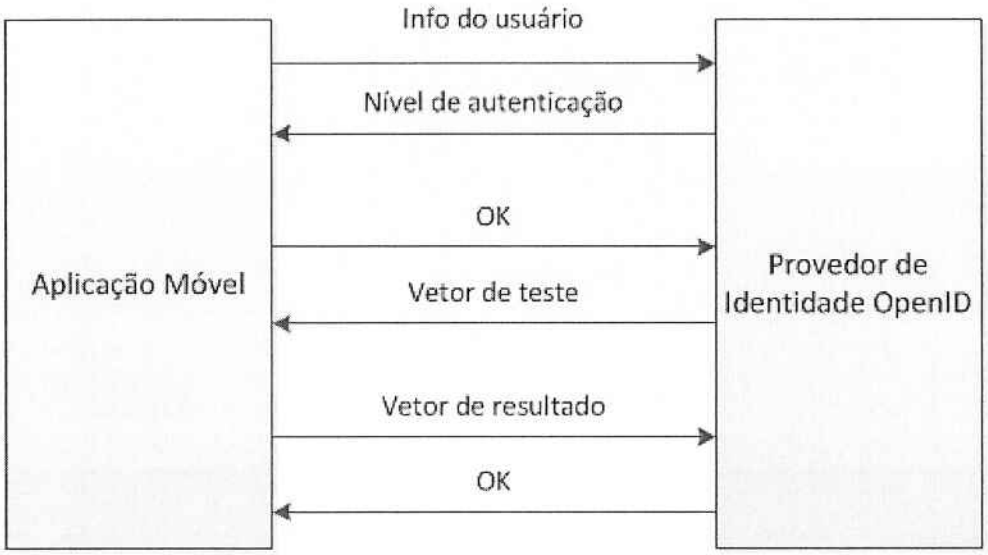

Figura 10 - Fluxo de autenticação

5.2.3 Comunicação Segura

Para o desenvolvimento da comunicação entre módulos apresentadas de forma a garantir os serviços básicos de segurança descritos na seção 2.1, estabelecendo uma relação inicial de confiança unilateral, onde a aplicação móvel confia no provedor de identidade, como já citado anteriormente, foi utilizado o protocolo HTTPS. Esse protocolo permite em primeiro lugar que, a partir da existência de um certificado digital válido, a aplicação garanta a legitimidade do provedor. Em segundo lugar, o uso do HTTPS garante a integridade e confidencialidade das mensagens trocadas entre as duas partes envolvidas.

5.3 *Provedor de Identidade OpenID*

Esta seção apresentará detalhes do funcionamento do provedor de identidade. Vale ressaltar que será dado enfoque às alterações realizadas no provedor já existente[14] que foi utilizado como base para este trabalho.

5.3.1 Estrutura Geral

O provedor de identidade apresenta de forma macroscópica uma estrutura em camadas de software, ilustrada na Figura 11.

A camada mais superior consiste na interface *Web* do sistema, que consiste num repositório de páginas *Web* responsáveis por receber dados de interação do usuário com o sistema e disparar as ações adequadas. As ações disparadas são enviadas para a camada imediatamente inferior, a camada de lógica do sistema, para serem tratadas e posteriormente visualmente apresentadas.

A camada de lógica do sistema recebe as ações geradas pela cama de interface e é responsável por trata-las adequadamente, gerando um resultado a ser devolvido para a camada superior, ou disparando novas ações de lógica até obter-se o resultado. Durante as ações lógicas, podem ser feitas chamadas de escrita ou leitura de dados à camada de persistência, caso seja detectada a necessidade da persistência de algum dado ou do acesso de algum dado persistido anteriormente.

A camada de persistência consiste na camada mais inferior do sistema, que realiza os acessos ao banco de dados para escrever ou ler os dados necessários.

[14] http://idp.larc.usp.br

Figura 11 - Estrutura Geral do Provedor de Identidade

5.3.2 Envio e Recebimento de Mensagens

O envio e recebimento de mensagens são realizados na camada de interface *Web* da Figura 11 apresentada na seção anterior.

As páginas JSP realizam o tratamento de mensagens *Web* por meio do acesso direto aos objetos:

- *request*: Objeto do tipo HttpServletRequest, que contém as informações relativas à requisição HTTP recebida pelo servidor de aplicação *Web*.
- *response*: Objeto do tipo HttpServletResponse, que contém informações relativas à resposta da requisição HTTP, que será enviada para o servidor de aplicação ao fim da execução daquela página.

Esses objetos permitem que sejam obtidos parâmetros das requisições e inseridos parâmetros nas respostas, estabelecendo assim os requisitos básicos para o uso do protocolo HTTPS como canal de comunicação entre os módulos.

5.3.3 Alterações para Envio e Recebimento de Informaç

Com base nos fluxos de comunicação descritos, foram criadas no provedor de identidade novas páginas *Web* que satisfaçam o recebimento e envio de informações dos fluxos. Nesta seção estão especificados os conteúdos das mensagens que são recebidas e enviadas por essas novas interfaces em cada um dos fluxos. Ressalta-se que os detalhes de geração dos dados contidos nessas mensagens serão abordados posteriormente.

Para o fluxo de cadastro, apresentado na seção 5.2.1, as interfaces *Web* devem inicialmente receber:

- Nome de usuário: Valor contendo *username* relativo à conta cadastrada no provedor de identidade.
- Senha: Valor contendo a senha da conta cadastrada.
- IMEI: Valor contendo o código de 15 dígitos relativo ao dispositivo móvel do usuário.
- IMSI: Valor contendo o código de 15 dígitos relativo ao *SIM Card* usado no dispositivo móvel do usuário.

Depois de realizado o processamento dessas informações, deve ser enviado o valor do segredo compartilhado que agirá como informação confiável de posse da aplicação móvel. A partir desse momento é realizado o fluxo de verificação de autenticidade.

O fluxo de verificação de autenticidade é disparado por uma requisição vazia enviada pela aplicação móvel para a interface web adequada, que indica sua disponibilidade para realizar o teste de autenticidade. A partir do recebimento desse aviso é produzido um vetor de testes relacionado ao segredo comum entre provedor e aplicação móvel que é enviado para a aplicação.

Por fim, a interface está preparada para o recebimento de um vetor de resultados do teste realizado. Com esse vetor será definido se a aplicação móvel é realmente confiável e representa uma autenticação adicional válida para a conta de usuário determinada anteriormente.

Para o fluxo de autenticação, descrito na seção 5.2.2, inicialmente a interface *Web* deve receber os parâmetros:

- Nome de usuário.
- IMEI.
- IMSI.

Como resposta, deve enviar um número de 1 a 3 representando o nível de autenticação desejado. A partir disso, ocorre o mesmo fluxo de verificação de autenticidade descrito acima.

Foram criadas diversas páginas *Web* para receberem requisições com os parâmetros necessário e enviarem respostas também com parâmetros necessários. Como discutido na seção 5.3.1, a interface *Web* dispara as chamas a lógica do sistema necessárias para o processamento e obtenção desses parâmetros.

5.3.4 Algoritmo de Verificação de Autenticidade

Com a interface *Web* preparada para receber e enviar as requisições necessárias, podemos apresentar a maneira como é realizado o fluxo de verificação de autenticidade, apresentado na seção 4.4.1.1.

Como já discutido, a primeira troca de informações entre aplicação móvel e provedor de identidade acontece no momento em que se associa um *smartphone* à conta do usuário, na etapa de cadastro. É nessa passagem que devem ser definidos os dois segredos compartilhados entre as duas partes do sistema, que posteriormente serão responsáveis pelos processos de autenticação requisitados.

A Figura 12 ilustra o fluxograma da geração dos segredos compartilhados.

1. Inicialmente, é utilizada uma função de geração de números pseudo-aleatórios para a obtenção de um número de 32 bytes. O comprimento do número aleatório foi escolhido de forma que o esforço para descobri-lo seja grande, caso um invasor tenha obtido as informações de IMEI e IMSI do *smartphone* do usuário.

2. Ao número obtido, é concatenado o IMEI do *smartphone*. O IMEI é composto por 15 dígitos, ou seja, 15 bytes.

3. Ao resultado, é concatenado o IMSI do *smartphone*, também com 15 bytes.

4. Nesta etapa é feita a separação dos dois segredos. É concatenado o dígito "1" para obter o primeiro valor, e o dígito "2" para obter o segundo. Os dígitos "1" e "2" representam um byte cada, portanto ao fim desta etapa, cada vetor contém 63 bytes.

5. Este resultado a uma função de *Hash*. No caso, é usada a função SHA-2 (*Secure Hash Algorithm*), que fornece um resumo de tamanho 256 bits. É importante ressaltar que apesar do vetor submetido à função de *Hash* diferir em apenas um dígito, o resultado obtido para os dois resumos criptográficos são completamente diferentes.

6. Como o resultado obtido na etapa anterior é composto por 256 bits, selecionamos apenas os 128 primeiros.

7. Por fim temos dois segredos diferentes de tamanho 128, que serão utilizados como chave do algoritmo criptográfico no método de verificação de autenticidade.

Em posse dos segredos compartilhados, já é possível realizar o método de verificação de autenticidade, que é o mesmo tanto para confirmação de cadastro, quanto para autenticações, como já discutido na seção 5.2. O fluxograma do algoritmo pode ser visto na Figura 13.

1. Primeiramente é gerado um número aleatório de 16 bytes. Este tamanho foi escolhido baseado no tamanho do bloco do algoritmo de criptografia AES que será utilizado nas etapas posteriores.

2. O número aleatório é submetido à criptografia utilizando o algoritmo criptográfico AES com um dos segredos compartilhados usados como chave. Como a mensagem a ser cifrada tem o tamanho de um bloco para o AES, não é necessária a utilização de um modo de operação, sendo feita apenas a encriptação pura.

3. O resultado obtido da criptografia é enviado para a aplicação móvel como resposta de uma requisição pendente.

4. O resultado de verificação obtido na aplicação é recebido no provedor em uma requisição.

5. A mensagem recebida é decifrada com o AES usando o segredo compartilhado novamente como chave.

6. O provedor verifica se o valor da decifração corresponde ao valor esperado. Caso corresponda, o provedor pode confiar na aplicação móvel.

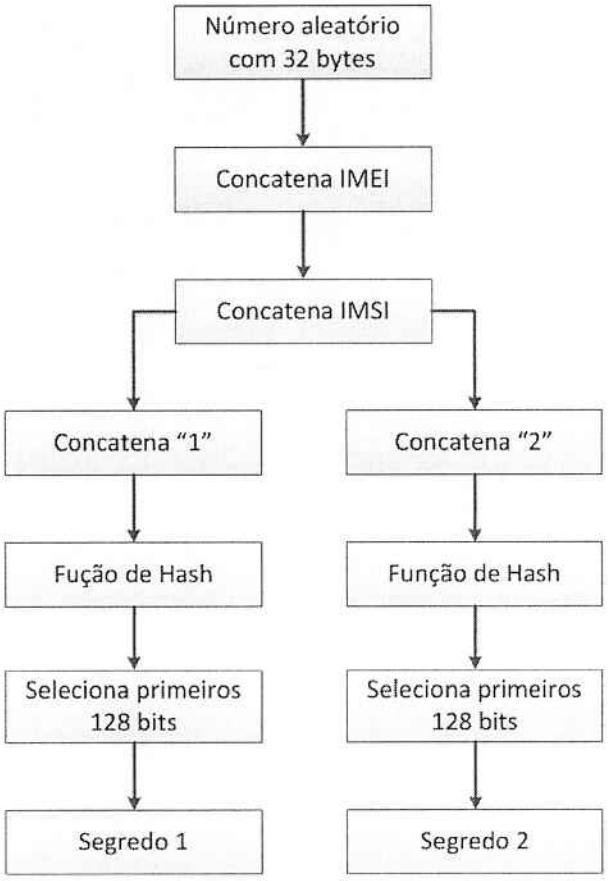

Figura 12 - Geração dos segredos compartilhados

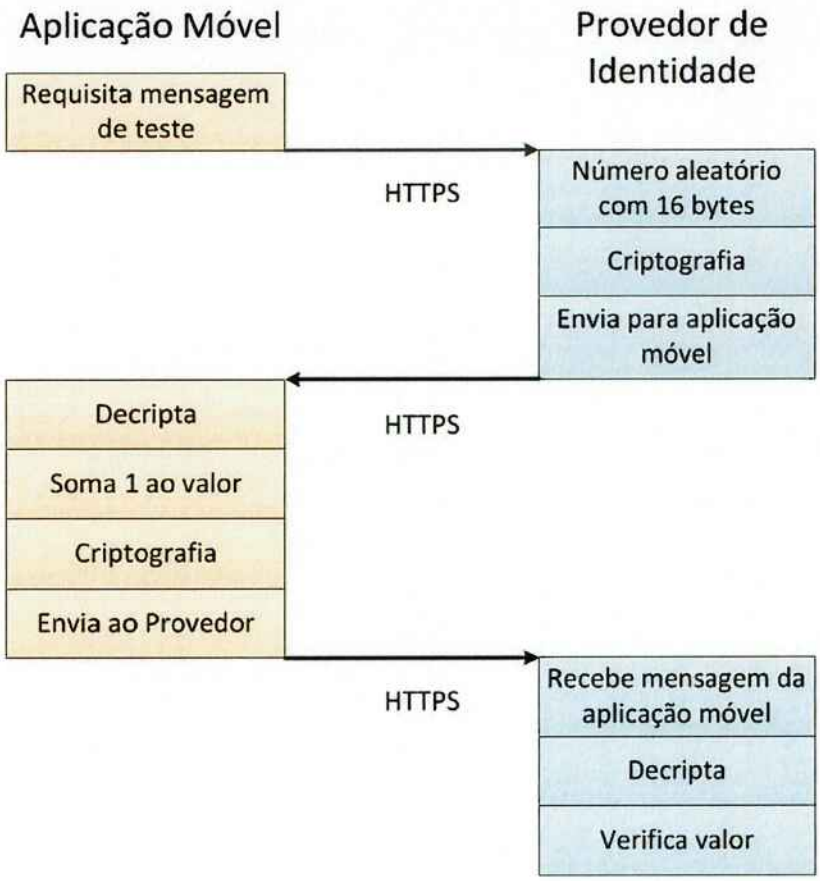

Figura 13 - Algoritmo de verificação de autenticidade no provedor de identidade

O algoritmo de verificação deve ser combinado com o algoritmo de verificação correspondente realizado na aplicação móvel, que será apresentado na seção 5.4.4.

5.3.5 Alterações na Estrutura do Banco de Dados

Para o funcionamento da aplicação móvel do MobileAuth junto ao provedor de identidade, as tabelas no banco de dados deste precisaram de adições de campos relacionados à aplicação móvel. As tabelas do provedor de identidade base que são relevantes são:

- User: Armazena dados de um usuário.
- Auth: Armazena dados relacionados a autenticação de um usuário.
- Site: Representa um site acessado por um usuário.

A Tabela 1 mostra as entradas da tabela User no banco de dados, antes das modificações realizadas para a adequação junto ao sistema MobileAuth. Apenas os dados relevantes para o projeto estão expostos.

Entrada	Tipo	Informação
username	String	Nome de usuário
creation_date	timestamp	Data de criação da conta

Tabela 1- Tabela User original

Já na Tabela 2, pode-se ver os campos adicionado à tabela User. Os campos *imei* e *imsi* foram inseridos para armazenar as informações do *smartphone* relacionado à conta do usuário. Já o campo *pendingRP* armazena a URL do último serviço para o qual há a intenção de liberação de acesso.

Entrada	Tipo	Informação
username	String	Nome de usuário
creation_date	timestamp	Data de criação da conta
imei	String	IMEI do *smartphone* relacionado à conta
imsi	String	IMSI do *smartphone* relacionado à conta
pendingRP	String	O serviço para o qual há uma requisição de autenticação pendente

Tabela 2 - Tabela User modificada

Na Tabela 3 podemos ver a descrição da tabela Auth do sistema original, enquanto as adições podem ser vistas na Tabela 4. Os campos *mobile_hashkey1* e *mobile_hashkey2* armazenam os segredos compartilhados entre provedor de identidade e aplicação móvel, enquanto o campo *mobile_status* permite identificar se a aplicação móvel daquele usuário está ativa ou congelada. O campo *mobileExpiration_date* representa a data máxima em que o acesso do usuário está permitido antes de necessitar se autenticar na

aplicação móvel novamente. Por fim, os campos *access_code* e *date* armazenam o código utilizado no caso de acesso alternativo e a data até a qual o acesso alternativo é permitido.

Entrada	Tipo	Informação
username	String	Nome de usuário
password	String	Senha
creation_date	timestamp	Data de criação da conta
expiration_date	timestamp	Data de expiração da conta
email	String	*E-mail* do usuário

Tabela 3 - Tabela Auth original

Entrada	Tipo	Informação
username	String	Nome de usuário
password	String	Senha
creation_date	timestamp	Data de criação da conta
expiration_date	timestamp	Data de expiração da conta
email	String	*E-mail* do usuário
mobile_hashkey1	String	Segredo compartilhado com a aplicação móvel 1
mobile_hashkey2	String	Segredo compartilhado com a aplicação móvel 2
mobile_status	String	*Status* do uso da aplicação móvel
mobileExpiration_date	timestamp	Data de expiração da autenticação móvel
access_code	String	Código para acesso alternativo

access_date	timestamp	Data de expiração do acesso alternativo

Tabela 4 - Tabela Auth modificada

Por fim, a tabela Site está ilustrada na Tabela 5, antes da modificação, e na Tabela 6, após a modificação. O campo *conter* representa o número de vezes que aquele serviço foi acessado, já o campo *mobile_option* indica o nível de segurança escolhido para aquele serviço, enquanto o campo *confirmation* indica se o acesso àquele serviço já foi autorizado utilizando a aplicação móvel.

Entrada	Tipo	Informação
site_id	String	Identificador do serviço
user_username	String	Nome de usuário
realm	timestamp	URL do serviço
expiration_date	timestamp	Data de expiração do acesso ao *site*
last_access	String	Data do último acesso ao *site*
creation_date	String	Data do primeiro acesso ao *site*

Tabela 5 - Tabela Site original

Entrada	Tipo	Informação
site_id	String	Identificador do serviço
user_username	String	Nome de usuário
realm	timestamp	URL do serviço
expiration_date	timestamp	Data de expiração do acesso ao *site*
last_access	String	Data do último acesso ao *site*
creation_date	String	Data do primeiro acesso ao *site*

counter	int	Número de vezes que o site foi acessado
mobile_option	int	Opção de nível de segurança para esse serviço
confirmation	String	Representa a confirmação da aplicação móvel

Tabela 6 - Tabela Site modificada

5.4 Aplicação Móvel

Está seção apresenta características de desenvolvimento da aplicação móvel responsável por garantir a autenticação adicional necessária ao sistema MobileAuth. Será apresentada, como na seção anterior, uma abordagem *top-down* do sistema desenvolvido, portanto será apresentada inicialmente a estrutura geral da aplicação móvel, seguida pelos detalhes de envio e recebimento de mensagens. Depois, é apresentada a sequência de interfaces para cada tipo de operação realizada na aplicação. A seguir, é detalhado o algoritmo de verificação de autenticidade. Por fim, são apresentados detalhes do armazenamento de informações relevantes na memória do dispositivo.

5.4.1 Estrutura Geral

A estrutura geral da aplicação móvel também pode ser representada em camadas e pode ser vista na Figura 14.

A camada mais superior consiste na camada de interface, onde o usuário interage com a aplicação móvel inserindo as informações necessárias para as operações desejadas por meio da tela sensível ao toque do dispositivo móvel.

Na camada abaixo estão as atividades, que representam a lógica do programa. Uma atividade, no desenvolvimento para Android, significa uma classe associada a uma tela de interface, que recebe informações de preenchimento de campos ou ações de botões e

dispara a lógica necessária para tratar essas entradas. Ao final do processamento das entradas, uma atividade transita para outra que exibe uma nova tela.

A camada mais inferior consiste na memória do dispositivo, onde dados podem ser lidos ou escritos. O acesso a memória é realizado pela camada de lógica, ou seja, pelas atividades. Os arquivos de uma aplicação são armazenados em um diretório oculto na memória do dispositivo, o qual só pode ser acessado pela aplicação que o criou, evitando assim que aplicações maliciosas tentem obter as informações contidas nos arquivos.

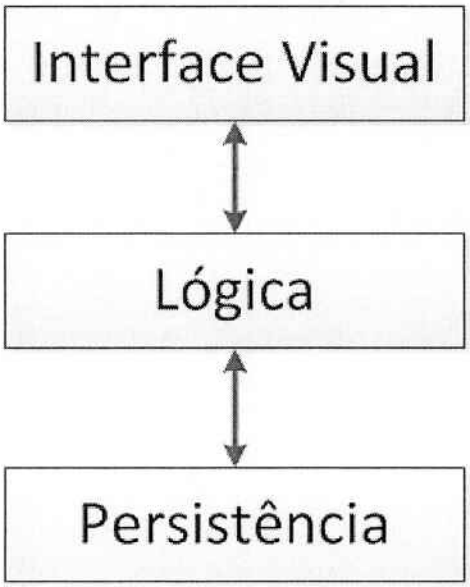

Figura 14 - Estrutura geral da aplicação móvel

5.4.2 Envio e Recebimento de Mensagens

No caso da aplicação móvel, não se tem disponível uma camada capaz de tratar requisições e respostas de requisições como no caso do provedor de identidade, discutido na seção 5.3.2, portanto esse tratamento é realizado na camada de atividades.

Para auxiliar no envio e recebimento de requisições, é utilizada aqui a biblioteca Apache HTTP, apresentada na seção 5.1.3. Com a utilização desta biblioteca, podem-se instanciar objetos das seguintes classes:

- HttpClient: Representa um cliente HTTP responsável por executar requisições.

- HttpPost: Representa uma requisição do tipo POST, onde é especificado o destino da requisição além dos seus parâmetros.
- HttpResponse: Representa a resposta a uma requisição HTTP, obtida quando o cliente executa a requisição.

Combinando esses três objetos básicos, além de outros auxiliares, são geradas as requisições com os parâmetros que devem ser enviados ao provedor de identidade e são obtidas as respostas a essas requisições, também com seus parâmetros e outros atributos.

5.4.3 Interfaces Visuais Principais

Estão apresentadas nesta seção as principais interfaces visuais da aplicação móvel.

A Figura 15 apresenta a tela principal da aplicação MobileAuth. Nesta tela o usuário pode decidir entre quatro opções de botão. O primeiro botão, *Connect*, é o botão acionado para realizar a autenticação adicional com o provedor de identidade. No botão *Settings* o usuário pode escolher entre mudar as configurações de sua aplicação móvel. O botão *Instructions*, quando selecionado, exibe instruções de como utilizar o MobileAuth. Por fim, o botão *Close* encerra a aplicação.

Na Figura 16 podem ser vistas as telas referentes aos níveis de autenticação: confirmação, senha e gesto.

Figura 15 - Tela principal da aplicação móvel

Figura 16 - Níveis de autenticação

5.4.4 Algoritmo de Autenticação

Com o detalhamento do envio e recebimento das mensagens detalhado na seção 5.4.2 e das interfaces visuais mais importantes na seção anterior, já é possível descrever o algoritmo de verificação de autenticidade. A Figura 17 mostra o fluxograma para esse algoritmo.

1. A aplicação móvel envia uma requisição do tipo POST para o provedor de identidade, solicitando que este responda com o vetor de teste. A requisição é montada utilizando os objetos da biblioteca Apache HTTP, como já mencionado.

2. Ao receber a resposta à requisição, a aplicação submete o valor recebido ao algoritmo criptográfico AES utilizando como chave um dos segredos compartilhados obtidos no cadastro da aplicação no provedor de identidade.

3. É realizada a soma do valor resultante da decriptação com o número 1.

4. O resultado da soma é encriptado, utilizando novamente o algoritmo AES e o segredo compartilhado como chave.

5. É criada uma nova requisição, tendo como parâmetro o resultado da criptografia, e enviada ao provedor.

6. A aplicação aguarda o provedor de identidade responder com a confirmação que o procedimento teve sucesso.

Figura 17 - Algoritmo de verificação de autenticidade na aplicação móvel

5.4.5 Armazenamento de informações

Como já discutido na seção 5.4.1, arquivos podem ser armazenados na memória do dispositivo móvel, em um diretório visível apenas a aplicação que o criou, de forma que outras aplicações não possam acessar ou modificar esse arquivo.

Para a aplicação MobileAuth, é necessário armazenar arquivos que contenham as informações de conta do usuário e, principalmente, os segredos compartilhados que serão responsáveis pela confirmação de autenticação.

Devido à existência de dois segredos diferentes, como discutido na seção 4.4.1.1, são mantidos dois arquivos na memória do dispositivo.

O primeiro arquivo guarda duas informações: o nome de usuário e o segredo compartilhado necessário ao primeiro nível de autenticação. Como já discutido, esse arquivo não é cifrado devido à inexistência de uma informação que seja usada como chave.

Já no segundo arquivo é guardada apenas a informação do segundo segredo compartilhado. Este arquivo é cifrado com o algoritmo AES, utilizando como chave a informação de senha do segundo nível de autenticação. Apesar dos arquivos não serem visíveis a outras aplicações do dispositivo, o segundo arquivo é cifrado visando evitar a obtenção da informação contida nele por meio de acesso forçado a memória do dispositivo.

5.5 Desenvolvimento das Funcionalidades

Tendo conhecimento das características de desenvolvimento dos dois módulos do sistema e com base nos casos de uso e requisitos funcionais definidos na seção 4.5 são expostos nesta seção o desenvolvimento das principais funcionalidades do sistema.

5.5.1 Criação de Conta

Como comentado anteriormente, a ação de criação de uma conta no provedor de identidade é semelhante à criação no provedor original utilizado como base do trabalho. Serão detalhados os passos com base no que foi apresentado nas seções anteriores.

1. Após a inserção dos dados do usuário na interface *Web* de cadastro são criadas entradas no banco de dados em todas as tabelas, incluindo as três apresentadas: *User*, *Auth* e *Site*.

2. São realizadas as seguintes atribuições aos novos atributos da tabela *Auth*:

    a. mobile_hashkey1 = "idle"

    b. mobile_kashkey2 = "idle"

c. mobile_status= "inexistent"

Os valores "*idle*" nos campos referentes ao segredo compartilhado significam que os segredos ainda não foram criados e compartilhados. Já o valor "*inexistent*" no campo referente ao *status* da aplicação representa que ainda não foi associado um *smartphone* àquela conta.

3. O usuário é redirecionado para uma página *Web* que o informa que para completar o cadastro deve realizar *download* e execução da aplicação móvel.

5.5.2 Associação de Aplicação Móvel

A ação de associação de aplicação móvel deve completar a etapa de cadastro apresentada na seção anterior, para isso são executados os seguintes passos:

1. No momento em que o usuário utilizar a aplicação pela primeira vez, deve inserir na interface o nome de usuário e a senha que foram previamente cadastradas no provedor de identidade.
2. A atividade relacionada àquela interface obtém os valores dos campos de texto preenchidos e obtém os valores de IMEI e IMSI do dispositivo.
3. É criada uma requisição com o uso da biblioteca Apache HTTP, onde são inseridos os quatro parâmetros obtidos no passo anterior. Essa requisição é executada.
4. Ao chegar no provedor de identidade, são obtidos os quatro parâmetros da requisição. O provedor verifica, primeiramente, se o nome de usuário e senha estão corretos para alguma das contas no servidor. Neste momento, são inseridos os seguintes valores na tabela *User*:
    a. imsi = parâmetro IMSI recebido na requisição.
    b. imei = parâmetro IMEI recebido na requisição.
    c. pendingRP = "None", representando que não há ainda um serviço pendente.
5. O provedor envia uma resposta à requisição recebida, inserindo como parâmetro o segredo compartilhado, gerado como especificado na seção 5.3.4. Nesse

momento, o campo *mobile_status* recebe o valor "waiting1", o que representa que está esperando pela confirmação do primeiro segredo compartilhado.

6. A aplicação móvel obtém o parâmetro e envia uma nova requisição ao servidos, sem parâmetros, solicitando o valor para o teste de autenticidade.
7. O provedor gera um número aleatório e realiza a encriptação do mesmo como especificado na seção 5.3.4 e envia como resposta à requisição.
8. A aplicação móvel obtém o parâmetro da resposta e realiza a operação necessária para a verificação de autenticidade, especificada na seção 5.4.4. É criada uma nova requisição com o resultado do teste como parâmetro e é enviada ao provedor.
9. O provedor obtém o parâmetro da requisição e verifica se o valor recebido corresponde ao valor esperado, que ficara armazenado na sessão do usuário. Sendo verdadeiro, o provedor insere no campo *mobile_hashkey1* da tabela *Auth* o valor do segredo compartilhado que foi gerado no passo 5, insere no campo *mobile_status* o valor "confirmed1", representando a confirmação do primeiro segredo, e envia a resposta à requisição contendo a mensagem de *status* "200 OK".
10. Ao receber a resposta, a aplicação cria o primeiro arquivo necessário, armazenando o nome de usuário e o segredo compartilhado. Após o armazenamento, a aplicação exibe a interface de inserção de senha para o nível médio de segurança.
11. A senha é obtida e a aplicação monta uma nova requisição, inserindo dessa vez como parâmetro apenas o nome de usuário, que é lido no arquivo que acaba de ser criado no passo 10. A requisição é enviada ao provedor.
12. O provedor verifica as informações do usuário e gera o segundo valor de segredo compartilhado. Dessa vez, o campo *mobile_status* recebe o valor "waiting2", analogamente ao que acontece no passo 5.
13. Os passos 6 a 9 são repetidos, dessa vez sendo armazenado o segredo no campo *mobile_hashkey2* e inserindo o valor "confirmed2" no campo *mobile_status*.
14. Ao receber a confirmação, a aplicação cria um novo arquivo contendo o segundo segredo compartilhado e o submete a cifração utilizando o algoritmo AES e a senha cadastrada pelo usuário como chave do mesmo.

15. A aplicação transita para a interface de cadastro de gesto. Ao obter o objeto de gesto, a aplicação o armazena em um diretório interno. Nesse momento é criada a última requisição referente a esta etapa, enviando como parâmetro novamente o nome de usuário.
16. Ao receber esta requisição, o provedor de identidade insere o valor "confirmed" no campo *mobile_status* e responde a requisição com a mensagem "200 OK".
17. A aplicação transita para a tela principal.

5.5.3 Acesso ao Provedor de Identidade

Após a associação do *smartphone* já é possível realizar o acesso à conta no provedor de identidade, este processo acontece da seguinte maneira:

1. No provedor, ao receber os parâmetros de acesso de nome de usuário e senha e após a verificação da validade dos mesmos, ocorre o redirecionamento para a página principal da conta de usuário.
2. Ao entrar no código referente a essa página, o provedor verifica o campo *mobileExpiration_date* da tabela *Auth* no banco de dados. Caso o horário atual tenha ultrapassado o horário armazenado neste campo, há um redirecionamento para uma página de notificação, que solicita que o usuário conecte sua aplicação móvel.
3. Ao utilizar a aplicação, esta cria uma requisição vazia e a envia ao provedor, esta requisição representa a solicitação do nível de autenticação.
4. O provedor recebe o parâmetro de nome de usuário e verifica o campo *pendingRP* na tabela *User*. No caso do acesso direto à conta do provedor de identidade, este campo conterá o valor "None". O provedor interpreta este valor, e como não há serviço pendente, responde a requisição enviando um parâmetro de nível de autenticação com o valor 3, representando o nível alto de autenticação.
5. A aplicação interpreta esta resposta e transita para a tela de inserção de gesto. Ao capturar o gesto, a aplicação utiliza a biblioteca *Gesture* para verificar se o gesto inserido corresponde ao gesto cadastrado pelo usuário. Em caso positivo, a aplicação agora transita para a tela de inserção se senha. A aplicação obtém a

senha inserida e tenta decriptar o arquivo do segredo compartilhado, caso seja possível, significa que a senha está correta, se estiver incorreta a aplicação não será capaz de realizar a decifração.

6. Com o arquivo decriptado, é lido o valor do segredo compartilhado e armazenado em uma variável auxiliar. A aplicação envia agora uma requisição ao provedor de identidade solicitando o valor para a realização do teste de autenticidade.
7. O provedor gera o número aleatório, o cifra e envia como resposta a requisição.
8. A aplicação realiza a operação necessária e envia uma nova requisição ao provedor, inserindo como parâmetro o resultado de seu teste de autenticidade.
9. O provedor recebe a requisição e faz a verificação. Caso o resultado esteja correto, o campo *mobileExpiration*_date é atualizado com o horário atual acrescido de 10 minutos e o acesso à conta é liberado. O provedor responde então a requisição com uma mensagem de confirmação "200 OK".

5.5.4 Acesso a Serviços

Com a associação do *smartphone* à conta do usuário já realizada, também pode-se solicitar o acesso a um serviço. Aqui, o procedimento é muito semelhante ao procedimento detalhado na seção anterior, portanto omitiremos os passos que realizem as mesmas ações, explicitando apenas as diferenças no acesso a um serviço.

1. A requisição de acesso começa no *site* do serviço que o usuário deseja acessar, onde o usuário insere seu identificador *OpenID*.
2. Nesta etapa acontece o fluxo de redirecionamentos do protocolo *OpenID*, já implementado pelo provedor de identidades utilizado como base para este trabalho, portanto o detalhamento da implementação desse fluxo não faz parte do escopo desta seção.
3. O fluxo *OpenID* redireciona o usuário à interface de *login* do provedor de identidade.
4. A ação responsável por realizar o *login* percebe que esta acontecendo um fluxo de autenticação em um serviço, esta verificação é legada do provedor utilizado. Com essa interpretação, o provedor obtém a URL do site que está tentando obter o

acesso e armazena essa informação no campo *pendingRP* da tabela *User*. Já na tabela *Site*, é incrementado o campo *counter*, que representa o número de vezes que aquele serviço foi acessado por aquele usuário.

5. Após a inserção das informações no banco de dados, o provedor realiza um redirecionamento para a página de autorização de serviço. Antes de renderizar esta página, é verificado o campo *counter* do banco de dados. Se este possuir o valor "1" significa que esta é a primeira vez que o usuário tenta acessar aquele serviço, portando são exibidas na página as opções de níveis de segurança para que o usuário selecione. Caso contrário essas opções não são exibidas.
6. No momento em que o usuário insere as informações e solicita a permissão para o serviço, o provedor de identidade o notifica que deve ser utilizada a aplicação móvel.
7. Ao ativar a aplicação móvel, esta envia uma requisição ao provedor solicitando o nível de autenticação que deve ser realizado, enviando como parâmetro o nome de usuário.
8. O provedor verifica na tabela *User* o campo *pendingRP* e verifica na tabela *Site* o campo *mobile_option* contendo o nível de segurança para aquele serviço. Caso seja o primeiro acesso àquele serviço, deve ser adotado o nível de autenticação mais alto. O provedor responde a requisição enviando o nível necessário.
9. A aplicação interpreta a resposta e transita para uma das telas de confirmação, dependendo do nível desejado.
10. A partir daqui ocorre o teste de verificação de autenticidade, já discutido exaustivamente nas seções anteriores.
11. Ao fim do teste, o provedor insere no campo *confirmation* da tabela *Site* o valor "confirmed" e redireciona o programa para o fim do fluxo de permissão de serviço.

5.5.5 Acesso Alternativo

Aqui são detalhadas as duas etapas da funcionalidade de acesso alternativo: a requisição de acesso alternativo, e o acesso propriamente dito. A requisição do acesso alternativo ocorre segundo os seguintes passos:

1. O provedor de identidade coleta informações de nome de usuário, senha e e-mail do usuário em uma página referente à solicitação de acesso alternativo após a confirmação do usuário por essa opção.
2. É gerado um código alfanumérico de 10 caracteres e armazenado no campo *access_code* da tabela *Auth*, além de inserir no campo *access_date* a data atual acrescida de 5 dias, ou seja, 120 horas. Já no campo *mobile_status* da tabela *Auth*, é inserido o valor "*frozen*", que representa o congelamento do uso da aplicação móvel.
3. O provedor envia um e-mail para o usuário informando o código de acesso alternativo.

A partir desse momento, o usuário não conseguirá mais utilizar sua aplicação móvel para se autenticar e o acesso será realizado da seguinte maneira:

1. Ao receber as informações de nome de usuário e senha, o provedor verifica o campo *mobile_status* e encontra o valor "*frozen*". Este valor é interpretado fazendo o provedor ser redirecionado a uma página onde deve inserir seu código de acesso alternativo.
2. O provedor obtém o código inserido e verifica se é o mesmo contido no campo *access_code*. Posteriormente verifica a data presente no campo *acess_date* e verifica se o horário atual ultrapassou o valor ali armazenado. Caso o código esteja correto e a data ainda não tenha sido ultrapassada, o acesso é liberado.

5.6 *Considerações sobre o Desenvolvimento do Sistema*

O desenvolvimento do sistema foi realizado com base na especificação detalhada no capítulo 4, seguindo os casos de uso e requisitos que foram propostos. Neste capítulo foram abordados os detalhes relativos à implantação das funcionalidades que o sistema deve cumprir, os algoritmos necessários às etapas destas funcionalidades e as alterações realizadas no provedor de identidade base para o correto funcionamento do sistema.

Ressalta-se a especial atenção dedicada ao desenvolvimento dos algoritmos de autenticação de forma a constituir um sistema com criptografia sólida utilizando-se das ferramentas e conceitos apresentados no capítulo 2.

No próximo capítulo são apresentadas as considerações finais acerca do sistema proposto e desenvolvido.

6 CONSIDERAÇÕES FINAIS

Neste capítulo serão apresentadas as considerações finais sobre o sistema MobileAuth e suas implicações. Mediante o cenário, os conceitos de segurança da informação e de provedores de identidade *user*-centric, e a especificação do sistema MobileAuth, será discutido primeiramente a aderência do sistema aos objetivos propostos, ressaltando-se as contribuições deste trabalho em relação aos sistemas padrão de gerenciamento de identidade.

Finalmente serão apresentados alguns possíveis trabalhos futuros, que constituiriam melhorias e complementações do projeto base aqui descrito.

6.1 Atendimento aos Objetivos

De acordo com os objetivos declarados na seção 1.1, pode-se dividir o sistema MobileAuth em duas vertentes. A primeira refere-se à constituição de uma autenticação adicional em um dispositivo móvel, reduzindo assim a vulnerabilidade do sistema frente a agressores que possuam a senha base do provedor de identidade.

A constituição do módulo de autenticação adicional se mostrou eficiente dado que impede a obtenção do acesso aos serviços associados a identidade *OpenID* a um indivíduo que não possua um *smartphone* cadastrado para realizar a autenticação adicional. Esse modelo cobre eficientemente a vulnerabilidade possivelmente causada pela escolha de uma senha fraca pelo usuário, fato que ocorre com frequência conforme descrito na seção 4.1.

A segunda vertente do sistema se refere ao objetivo de oferecer ao usuário diversas formas de combinar usabilidade e segurança por meio de diferentes níveis de segurança na autenticação móvel. Conforme especificado, o sistema MobileAuth por ser integrado à uma aplicação de *smartphone* não constitui um obstáculo significativo a adoção da tecnologia de gerenciamento de identidade *user-centric* dado o crescimento vertiginoso

da adoção desse tipo de dispositivo móvel no dia-a-dia dos usuários da *Internet*, conforme descrito na seção 4.3.3.

A partir dessa interação base constituída pela autenticação através do dispositivo móvel, o sistema MobileAuth apresenta uma gama de três opções para níveis de segurança, modelo este que se adere às necessidades de cada serviço. Ao contrário dos sistemas de gerenciamento de identidade padrão o sistema MobileAuth trata serviços distintos com relações de *trade-off* distintas.

6.2 Trabalhos Futuros

Dado a versatilidade do sistema MobileAuth, existem diversas vertentes possíveis melhorias e aumento de funcionalidades do mesmo. São possíveis desde alteração nas ferramentas, como uso de outros algoritmos criptográficos, até inclusão de novos níveis de segurança que ofereçam uma maior gama de opções para o comparativo segurança versus usabilidade.

Dentre as possíveis alterações tecnológicas destaca-se a adoção do padrão SHA-3 como algoritmo de geração de resumo criptográfico, que está atualmente em processo de definição com previsão de conclusão para os últimos meses do ano de 2012.

No que se refere à especificação de novos níveis de segurança, um ponto interessante de partida é o atual nível três da especificação do sistema MobileAuth. A substituição da API Gesture por um sistema mais rebuscado representa uma adição significativa na segurança e na usabilidade do sistema.

Sistemas como leitura de impressões digitais, reconhecimento facial ou de voz excluem a necessidade da memorização de uma senha adicional (vide nível dois de segurança) ao mesmo tempo em que restringem o acesso ao usuário cadastrado, dado à necessidade do reconhecimento biométrico.

Ressalta-se finalmente que a adição de níveis mais seguros constitui uma ampliação no escopo das aplicações que podem ser cobertas pelo sistema MobileAuth, i.e., aplicações de segurança crítica como operações bancárias podem se adequar ao sistema

MobileAuth expandido, empregando as vantagens do ambiente user-centric para esse tipo de serviço.

REFERÊNCIAS BIBLIOGRÁFICAS

ALLIANCE, L. **Introduction to the Liberty Alliance Architecture**. . [S.l: s.n.]. Disponível em: <http://www.projectliberty.org/> , 2003. Acesso em: 20 nov. 2011.

DHAMIJA, R.; DUSSEAULT, L. The Seven Flaws of Identity Management: Usability and Security Challenges. **IEEE Security & Privacy**, v. 6, n. 2, p. 24-29, abr 2008.

DIERKS, T.; RESCORLA, E. The transport layer security (TLS) protocol. **IETF RFC 4346**, 2006.

ELKY, S. **An Introduction to Information System Risk Management**. . [S.l: s.n.]. Disponível em: <http://www.sans.org/reading_room/whitepapers/auditing/introduction-information-system-risk-management_1204>. , 2006

FARRELL, S.; ADAMS, C. **Internet X.509 Public Key Infrastructure Certificate Management Protocols**. Disponível em: <http://tools.ietf.org/html/rfc2510>. Acesso em: 2 dez. 2011.

FLOSI. **comScore Reports July 2011 U.S. Mobile Subscriber Market Share - comScore, Inc.** Disponível em: <http://www.comscore.com/Press_Events/Press_Releases/2011/8/comScore_Reports_July_2011_U.S._Mobile_Subscriber_Market_Share>. Acesso em: 2 dez. 2011.

JAIN, A. K.; FLYNN, P.; ROSS, A. A. **Handbook of Biometrics**. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

JavaServer Pages Overview. [S.l: s.n.]. Disponível em: <http://www.oracle.com/technetwork/java/overview-138580.html>. Acesso em: 3 dez. 2011 , [S.d.]

KISSEL, B. **OpenID 2009 Year in Review | OpenID**. Disponível em: <http://openid.net/2009/12/16/openid-2009-year-in-review/>. Acesso em: 5 dez. 2011.

LABALME, F.; LINDELSEE, M.; WACHOB, G. **An Introduction to XRIs**. [S.l: s.n.]. Disponível em: <http://docs.oasis-open.org/xri/xri/V2.0/>. , 2005

MALINEN, J. **Windows CardSpace**. [S.l: s.n.]., [S.d.]

MICROSOFT TECHNET. **SSL/TLS in Detail**. Disponível em: <http://technet.microsoft.com/pt-br/library/cc785811(WS.10).aspx>. Acesso em: 2 dez. 2011.

MORGAN, R. L.; CANTOR, S.; CARMODY, S.; HOEHN, W.; KLINGENSTEIN, K. Federated Security: The Shibboleth Approach. **EDUCAUSE Quarterly**, v. 27, n. 4, p. 12-17, 2004.

NBR ISO/IEC 17799, A. **Tecnologia da informação - Código de prática para a gestão da segurança da informação**. [S.l: s.n.] , 2001

NOURIE, D. **The Java Technology Phenomenon**. [S.l: s.n.]. Disponível em: <http://docs.oracle.com/javase/tutorial/getStarted/intro/index.html>. Acesso em: 3 dez. 2011. [S.d.]

POPE, S.; JOSANG, A. **User Centric Identity Management**. [S.l: s.n.]. , 2005

RECORDON, D.; FITZPATRICK, B. **OpenID authentication 2.0-final**. [S.l: s.n.]. Disponível em: <http://openid.net/specs/openid-authentication-2_0.html>. , 2007

SAHA, A. K. **A Developer's First Look At Android**. [S.l: s.n.], 2008

SAKURAGUI, R. **GERENCIAMENTO DE IDENTIDADES COM PRIVACIDADE DO USUÁRIO EM AMBIENTE WEB**. . [S.l: s.n.], 2011

STALLINGS, W. **Cryptography and Network Security: Principles and Practice**. 3. ed. [S.l.]: Pearson Education, 2002.

STINSON, D. **Cryptography: Theory and Practice, Second Edition**. [S.l.]: {Chapman & Hall/CRC}, 2002.

STONE, B. Facebook Aims to Extend Its Reach Across the Web. **The New York Times**, 1 dez 2008.

TIOBE Software: Tiobe Index. Disponível em: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>. Acesso em: 3 dez. 2011.

TRACY, K. Identity management systems. **IEEE Potentials**, v. 27, n. 6, p. 34-37, dez 2008. **The Apache Software Foundation**. Disponível em: <http://struts.apache.org/>. Acesso em: 3 dez. 2011.

YAN, J.; BLACKWELL, A.; ANDERSON, R.; GRANT, A. Password memorability and security: empirical results. **IEEE Security & Privacy**, v. 2, n. 5, p. 25-31, out 2004.

YEE, K.-PING. Aligning Security and Usability. **IEEE SECURITY AND PRIVACY**, v. 2, p. 48-55, 2004.

ANEXO 1 – TRANSPORT LAYER SECURITY

O objetivo principal do protocolo TLS é prover privacidade e integridade dos dados entre dois pontos de comunicação entre aplicações. O protocolo é composto por duas camadas: o *TLS Record Protocol* e o *TLS Handshake Protocol.* O de nível mais baixo, construído em cima de algum protocolo de transporte confiável (e.g. TCP), está o *TLS Record Protocol.*

O *TLS Record Protocol* provê conexões seguras que possuem duas propriedades básicas:

- A conexão é privada. Criptografia simétrica é utilizada para encriptação de dados (e.g., AES, RC4, etc.). As chaves para essa encriptação simétrica são geradas de forma única para cada conexão e são baseadas em um segredo negociado por outro protocolo (como o TLS Handshake Protocol). O Record Protocol também pode ser usado sem encriptação.
- A conexão é confiável. O transporte da mensagem inclui uma checagem da integridade da mesma, envolvendo o uso de MAC com chaves. Funções de hash seguras (e.g., SHA-1, etc.) são usadas para o processamento do MAC. O Record Protocol pode operar sem um MAC, mas de modo geral só é utilizado nesse modo enquanto outro protocolo está usando o Record Protocol como um transporte para a negociação de parâmetros de segurança.

O *TLS Record Protocol* é usado para o encapsulamento de vários protocolos de nível mais alto. Um dos protocolos encapsulados, o *TLS Handshake Protocol,* permite o servidor e o cliente se autenticarem entre si e negociarem um algoritmo de encriptação e chaves criptográficas antes do protocolo do nível de aplicação enviar ou receber o primeiro byte de dados. O *TLS Handshake Protocol* provê conexão segura que possui três propriedades básicas:

- A identidade das partes pode ser autenticada usando criptografia assimétrica (e.g., RSA, DAS, etc.). Essa autenticação pode ser feita de forma opcional, mas de modo geral é necessária a pelo menos uma das partes.

- A negociação de uma segredo compartilhado é segura: O segredo negociado não está disponível para invasores, e em nenhuma conexão autenticada o segredo pode ser obtido, mesmo que um invasor possa se colocar no meio da conexão.
- A negociação é confiável: nenhum invasor pode modificar a comunicação envolvida na negociação sem que seja detectado pelas partes.

Na sequencia está descrito um exemplo completo autenticação, incluindo a autenticação do cliente, por meio do TLS usando certificados trocados entre as partes.

- O cliente manda uma mensagem de *ClientHello* especificando o protocolo TLS de versão mais alta que ele suporta, um número aleatório, uma lista de funções de encriptação sugeridas e métodos de compressão.
- O servidor responde com uma mensagem de *ServerHello*, contendo a versão do protocolo escolhido, um número aleatório, a função de encriptação escolhida e o método de compressão. O servidor pode também mandar um identificador de sessão como parte da mensagem, para realizar uma negociação resumida.
- O servidor envia a sua mensagem de certificado.
- O servidor requisita o certificado do cliente, para que dessa forma a conexão possa ser mutuamente autenticada, usando uma mensagem do tipo *CertificateRequest*.
- O servidor envia uma mensagem do tipo *ServerHelloDone*, indicando que a negociação inicial foi concluída.
- O cliente responde com uma mensagem contendo seu certificado digital.
- O Cliente manda uma mensagem do tipo *ClientKeyExchange*, que contém sua chave pública. Essa mensagem é encriptada usando a chave pública do servidor.
- O cliente envia uma mensagem do tipo *CertificateVerify*, que é uma assinatura sobre as mensagens de negociação anteriores usando a chave privada do cliente. Essa assinatura pode ser verificada utilizando a chave pública do cliente. Isso permite que o servido saiba que o cliente tem acesso a chave privada e consequentemente possui o certificado declarado.
- O cliente e o servidor usam os números aleatórios e as chaves públicas para produzir um segredo comum, chamado *master secret*. Todas as outras chaves para essa conexão são produzidas a partir desse segredo comum.

- O cliente envia uma mensagem do tipo *ChangeCipherSpec*, essencialmente dizendo ao servidor, "Tudo que eu disser a partir de agora será autenticado e encriptado."
- Finalmente o cliente manda uma mensagem encriptada de finalização, contendo dados para verificação da troca de mensagens.
- O servidor vai tentar desencriptar a mensagem de finalização do cliente e verificar os dados. Se a desencriptação falhar, a negociação é considerada mal sucedida e a conexão é desfeita.
- O servidor por sua vez também envia uma mensagem do tipo *ChangeCipherSpec*, dizendo para o cliente, "Tudo que eu disser a partir de agora será autenticado e encriptado."
- O servidor envia sua mensagem de Finalização.
- O cliente realiza o mesmo processo de desencriptação e verificação do servidor.
- Fase de aplicação: A partir desse ponto, a negociação está completa e o protocolo de aplicação está habilitado.

ANEXO 2 – PADRÃO DE CERTIFICAÇÃO X.509

O X.509 é um padrão da International Telecommunication Union - Telecommunication Standardization Sector (ITU-T) que define, entre outros procedimentos, formatos padrão para certificados de chave pública.

O modelo X.509 define um conjunto restrito de autoridades certificadoras que são responsáveis por assegurar a autenticidade de informações, que no caso abordado, são as chaves públicas das partes que desejam se comunicar de forma segura.

Esse modelo difere dos modelos de redes de confiança (web of trust models), em que qualquer página pode assinar certificadores e não só autoridades certificadoras pré-determinadas.

A estrutura do padrão X.509 está descrita abaixo:

- Certificado
    a. Versão
    b. Número de identificação
    c. Identificação do algortimo
    d. Emissor
    e. Validade
        i. Não antes que:
        ii. Não depois de:
    f. Proprietário
    g. Informações da chave pública do proprietário
        i. Algoritmo de chave pública
        ii. Chave pública
    h. Identificador único do emissor (opcional)
    i. Identificador único do proprietário (opcional)
    j. Extensões (opcional)
- Algoritmo da assinatura do certificado
- Assinatura do certificado