

UNIVERSIDADE FEDERAL DE GOIÁS  
INSTITUTO DE INFORMÁTICA

DANIEL MELO

# **Uso de Webservices Criptográficos a partir de Dispositivos Móveis**

**Avaliação de viabilidade**

Goiânia  
2016

UNIVERSIDADE FEDERAL DE GOIÁS  
INSTITUTO DE INFORMÁTICA

**AUTORIZAÇÃO PARA PUBLICAÇÃO DE DISSERTAÇÃO  
EM FORMATO ELETRÔNICO**

Na qualidade de titular dos direitos de autor, **AUTORIZO** o Instituto de Informática da Universidade Federal de Goiás – UFG a reproduzir, inclusive em outro formato ou mídia e através de armazenamento permanente ou temporário, bem como a publicar na rede mundial de computadores (*Internet*) e na biblioteca virtual da UFG, entendendo-se os termos “reproduzir” e “publicar” conforme definições dos incisos VI e I, respectivamente, do artigo 5º da Lei nº 9610/98 de 10/02/1998, a obra abaixo especificada, sem que me seja devido pagamento a título de direitos autorais, desde que a reprodução e/ou publicação tenham a finalidade exclusiva de uso por quem a consulta, e a título de divulgação da produção acadêmica gerada pela Universidade, a partir desta data.

**Título:** Uso de Webservices Criptográficos a partir de Dispositivos Móveis – Avaliação de viabilidade

**Autor(a):** Daniel Melo

Goiânia, 13 de Dezembro de 2016.

---

Daniel Melo – Autor

---

Marcelo Akira Inuzuka – Orientador

DANIEL MELO

# **Uso de Webservices Criptográficos a partir de Dispositivos Móveis**

**Avaliação de viabilidade**

Dissertação apresentada ao Programa de Pós-Graduação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Mestre em Computação.

**Área de concentração:** Otimização.

**Orientador:** Prof. Marcelo Akira Inuzuka

Goiânia  
2016

DANIEL MELO

# **Uso de Webservices Criptográficos a partir de Dispositivos Móveis**

## **Avaliação de viabilidade**

Dissertação defendida no Programa de Pós-Graduação do Instituto de Informática da Universidade Federal de Goiás como requisito parcial para obtenção do título de Mestre em Computação, aprovada em 13 de Dezembro de 2016, pela Banca Examinadora constituída pelos professores:

---

**Prof. Marcelo Akira Inuzuka**  
Instituto de Informática – UFG  
Presidente da Banca

---

**Prof. <Nome do membro da banca>**  
<Unidade acadêmica> – <Sigla da universidade>

---

**Profa. <Nome do membro da banca>**  
<Unidade acadêmica> – <Sigla da universidade>

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador(a).

**Daniel Melo**

Graduado na Universidade Federal de Goiás como Bacharel em Sistemas de Informação no ano de 2016, tendo desenvolvido como Trabalho de Conclusão de Curso um estudo de viabilidade do fornecimento de webservices criptográficos seguindo o padrão OpenPGP a partir de dispositivos móveis.

Dedido este trabalho aos meus pais, Maria de Lourdes de Oliveira Melo e Antonio Melo Lima, a quem eu amo, que com seu esforço, carinho e dedicação, construíram as bases de todas as capacidades que me permitem concluir este curso.

---

## Agradecimentos

---

<Texto com agradecimentos àquelas pessoas/entidades que, na opinião do autor, deram alguma contribuição relevante para o desenvolvimento do trabalho.>

<Epígrafe é uma citação relacionada com o tópico do texto>

**<Nome do autor da citação>,  
<Título da referência à qual a citação pertence>.**



---

## Resumo

---

Melo, Daniel. **Uso de Webservices Criptográficos a partir de Dispositivos Móveis**. Goiânia, 2016. 59p. Dissertação de Mestrado. Instituto de Informática, Universidade Federal de Goiás.

Apesar de a criptografia ser conhecida há muitos anos o desenvolvimento das ferramentas e técnicas não tem sido suficiente para assegurar a sua adoção por grandes grupos de usuários. Neste trabalho propomos o uso de web services capazes de oferecer recursos de criptografia a partir de dispositivos móveis, oferecendo uma alternativa para que o usuário final possa realizar o transporte e a gestão de seus recursos de segurança com mais simplicidade. Essa abordagem também busca facilitar a integração à outras ferramentas que desejem usar tais recursos. Essas capacidades são desenvolvidas mantendo o controle dos recursos criptográficos com o usuário final.

### Palavras-chave

Criptografia, PGP, Webservices, Dispositivo Móvel

---

## Abstract

---

Melo, Daniel. **Using Cryptographic Web services from Mobile Devices**. Goiânia, 2016. 59p. MSc. Dissertation. Instituto de Informática, Universidade Federal de Goiás.

Although the encryption be known for many years the development of tools and techniques have not been sufficient to ensure its adoption by large groups of users. In this paper we propose the use of web services capable of offering encryption capabilities from mobile devices, offering an alternative so the end user can perform transport and management of its security features with more simplicity. This approach also aims to facilitate the integration with other tools that want to use such resources. These capabilities are developed maintaining the end user in total control of his cryptography resources.

### Keywords

Cryptography, PGP, Web services, Mobile devices

---

# Sumário

---

Lista de Figuras	11
Lista de Tabelas	13
Lista de Algoritmos	14
Lista de Códigos de Programas	15
1 Introdução	16
2 Desafios da criptografia fim a fim	18
3 Uso do padrão OpenPGP para criptografia	20
4 Análise de formas de acesso à chaveiros criptográficos	21
4.1 Acesso através de linha de comando e bibliotecas	21
4.2 Token de Segurança Móvel	24
4.3 Considerações	26
5 Proposta de chaveiro pessoal móvel acessível por Webservices	27
5.1 Descrição da proposta	27
5.2 Operações dos Webservices	29
5.2.1 Autorização para utilizar os serviços	30
5.2.2 Criptografar um novo conteúdo	30
5.2.3 Consultar resultado da criptografia de conteúdo	31
5.2.4 Descriptografar uma mensagem PGP recebida	33
5.2.5 Consultar resultado da descriptografia de conteúdo	34
6 Experimento	37
6.1 Implementação dos serviços para o chaveiro PGP	38
6.2 Demonstração do consumo dos serviços	40
6.3 Análise de desempenho da solução	42
6.4 Análise de vulnerabilidades da solução	42
7 Trabalhos futuros	44
8 Conclusão	45
Referências Bibliográficas	46

A	Informações históricas do PGP	49
B	Demonstração de criação de chave e descryptografia com GnuPG	50
C	Detalhes do dispositivo móvel usado para testes	53
D	Detalhes de itens de configuração do aplicativo	54
E	Ambiente de desenvolvimento utilizado no experimento	55
F	Exemplo de cliente que consome os serviços criptográficos	56
G	Exemplo de um Apêndice	58
H	Exemplo de um Apêndice	59

---

## Lista de Figuras

---

4.1	Exemplo de operação de criptografia sendo realizada por linha de comando	22
4.2	Captura de tela a aplicação Enigmail, exibindo detalhes do recurso de descrição de criptografia	23
4.3	Diagrama com a proposta de funcionamento do enigmail com o módulo openpgp para Apache HTTP	24
4.4	Exemplos de diversos tipos de token, como aqueles em formato de pendrive, cartão criptográfico e token desconectado, gerador de código secreto com e sem entrada de senha pessoal	25
4.5	Representação da forma de acesso aos tokens conectados	25
5.1	representação de uma operação de criptografia com chaveiro em dispositivo móvel	28
5.2	ilustração de como a chamada de um serviço criptográfico aciona o Open KeyChain que, por sua vez, pede a senha do chaveiro do usuário para proceder	29
5.3	Captura de tela da aplicação Open KeyChain pedindo a senha da chave privada para o usuário alice@email.com	29
5.4	Funcionamento de uma requisição de encriptação com assinatura	30
5.5	Diagrama com o fluxo de consulta do resultado de uma operação de criptografia	32
5.6	Funcionamento de uma requisição de encriptação com assinatura	33
5.7	Diagrama com o fluxo de consulta do resultado de uma operação de criptografia	35
6.1	Captura de tela da aplicação OpenKey Chain, com duas chaves privadas de exemplo	37
6.2	Captura de tela da aplicação desenvolvida neste experimento.	38
6.3	Ilustração em alto nível das camadas envolvidas na implementação dos webservices em dispositivo móvel	39
6.4	Principais entidades que compõem a aplicação desenvolvida no experimento	39
6.5	Requisição de criptografia de conteúdo usando o webservice criptográfico	40
6.6	Consultando resultado de um pedido de criptografia com o requestId	41
6.7	Executando pedido de descrição de criptografia de conteúdo	41
6.8	Executando pedido de descrição de criptografia de conteúdo	42
B.1	Listagem de chaves presentes no chaveiro GnuPG	50
B.2	Comando e questionário para geração de nova chave com GnuPG	51
B.3	Resultado da geração da chave	51

B.4	Resultado da geração da chave	52
F.1	Captura de tela de aplicação experimental desenvolvida para fazer uso dos serviços criptográficos contidos no dispositivo móvel.	56

---

# Lista de Tabelas

---

---

## **Lista de Algoritmos**

---



---

## Lista de Códigos de Programas

---

5.1	Mensagem do pedido de criptografia	31
5.2	Resposta do pedido de criptografia	31
5.3	Resposta do pedido de criptografia em caso de processamento concluído	32
5.4	Resposta da consulta à pedido de criptografia ainda em andamento	33
5.5	Mensagem enviada para descriptografar uma mensagem OpenPGP	34
5.6	Exemplo de resposta do pedido de descriptografia	34
5.7	Exemplo de resposta em caso de processamento concluído	35
5.8	Exemplo de resposta em caso de processamento ainda em andamento	36

## Introdução

---

O uso crescente de ferramentas sociais para comunicação entre as pessoas em ambientes diversificados traz a necessidade da garantia de privacidade de forma efetiva e fácil de usar. A produção de informação é parte da vida das pessoas em muitos contextos em que convivem. Estes dados trafegam por vários meios desprotegidos, como a internet. As formas de proteger os dados e a privacidade de quem usa os recursos computacionais são frequentemente desconhecidas dos próprios usuários dos sistemas.

Ferramentas de comunicação tem alcançado grande público e compõem uma parte importante da troca de mensagens. Um exemplo atual é o Whatsapp, que conta com mais de 600 milhões de usuários. A ferramenta promete privacidade total nas versões mais recentes, dotadas de criptografia fim a fim, segundo a própria empresa [31]. Como se trata de uma aplicação proprietária, não é possível auditar se a implementação de fato segue o que é divulgado ao público. Sem a possibilidade de verificar o que é, de fato, realizado pela aplicação o controle sobre as pontas não está nas mãos dos usuários.

Assim, percebe-se a demanda de software criptográfico auditável, necessariamente de código aberto, que o usuário tenha condições plenas de controlar pessoalmente, ou por terceiros confiáveis, toda informação protegida por criptografia desde sua origem até o seu destino.

Apesar do uso da criptografia ser conhecido há muitos anos [20] ele ainda é de difícil compreensão e uso para usuários finais. Visto que a facilidade de uso precede uma adoção em massa de qualquer tecnologia [27], existe necessidade de desenvolver formas mais simples de uso para potencializar ações de segurança da informação.

Este trabalho faz uma análise do uso atual de criptografia de chaves assimétricas utilizando o software GnuPG, que segue o padrão PGP, para troca de mensagens em ferramentas com recursos de criptografia. Propõe-se em seguida a implementação de um chaveiro criptográfico em dispositivo móvel que ofereça facilidade de gestão dos recursos de segurança. Tal chaveiro será dotado de web services para que aplicações que desejem fazer uso de seus recursos de segurança de criptografia consigam com complexidade agora reduzida.

Neste trabalho, procuramos oferecer uma solução de acesso simples via web-

services à chaveiro criptográfica em dispositivo móvel, a fim de que aplicações diversas possam consumi-lo e assim facilitar o uso da criptografia por usuários finais.

O capítulo 2 levanta alguns pontos desafiadores para a criptografia fim a fim e sua relação com a necessidade por software aberto. O capítulo 3 apresenta a tecnologia de criptografia PGP, e, em seguida, o capítulo 4 traz a análise de algumas ferramentas que implementam essa tecnologia de criptografia com ênfase na sua forma de acessar os recursos privados. No capítulo 5 propomos uma forma de acesso aos recursos de chaveiro mantidos em um celular e é feito um experimento desta proposta, apresentado no capítulo 6. O capítulo 7 explora algumas possibilidades de trabalhos futuros e, por fim, o capítulo 8 apresenta algumas conclusões obtidas no desenvolvimento desta obra.

## Desafios da criptografia fim a fim

---

A criptografia fim a fim é definida pela implementação de técnicas que garantam que somente o remetente e o destinatário tem acesso às mensagens trocadas, sendo computacionalmente inviável que alguém as leia, seja por meio de interceptação ou acesso indevido aos dispositivos físicos envolvidos. Esse tipo de criptografia não tem sido historicamente desenvolvida com foco no usuário final. Como explorado por Sheng et Al [25], várias dificuldades são encontradas por usuários finais quando é colocada à prova a usabilidade das ferramentas para criptografia avaliadas. Verificamos mais à frente como algumas ferramentas com essa proposta acessam o chaveiro criptográfico.

Neste trabalho exploramos a ideia de que o desenvolvimento de aplicações que usem os serviços de criptografia oferecidos por um chaveiro PGP pode ser simplificado. O experimento explorado em seções não trata a usabilidade final, que é delegada à aplicação OpenKeyChain [13]. Seu escopo limita-se à exploração de um facilitador de uso do chaveiro por desenvolvedores, simplificando o desenvolvimento de aplicações que precisem fazer uso de tais recursos de segurança.

Um dos grandes desafios dos sistemas operacionais modernos está na sua capacidade de manter a segurança dos dados de seus usuários. A segurança desses dados deve observar alguns conceitos fundamentais, como a confidencialidade - manter o acesso à informação somente à quem tem esse direito - e a integridade - a certeza de que o conteúdo não foi corrompido, seja por acidente ou de forma proposital.

Frequentemente os sistemas tem adotado alguma tecnologia para proteger as mensagens trocadas por meio de redes, como a internet. São exemplos os protocolos protegidos por camada de segurança SSL, como o HTTPS, SMTPS, para troca de hipertexto e e-mail, respectivamente. Essa estratégia busca impedir que mensagens interceptadas possam ser lidas por atacantes [21]. Também torna computacionalmente complexo introduzir conteúdo nas mensagens sem que isso seja notado, preservando sua integridade.

Esses meios, entretanto, somente protegem a mensagem no caminho. Quando aplicado à troca de mensagens, isso significa que o provedor do serviço terá acesso ao seu conteúdo se ele não for previamente encriptado e ele será armazenado sem proteção. Conforme exemplificado no anexo 1, o processamento das mensagens por provedores de

e-mail é uma prática comum nessas ferramentas [24].

A criptografia assimétrica propõe um modelo de solução para o problema de proteção dos dados armazenados. Cada usuário gera um par de chaves criptográficas, sendo uma de propósito privado e a outra de propósito público. A chave privada é usada para assinar e descriptografar as mensagens. A chave pública, por sua vez é usada para verificar assinaturas e criptografar as mensagens, que só poderão ser lidas por quem possuir a chave privada equivalente. Isso cria um mecanismo onde somente o destinatário pode ler as mensagens, visando a confidencialidade. Além disso, a capacidade de assinatura provê o recurso de não-repúdio e integridade da comunicação. Se esses recursos forem empregados na troca de mensagens temos um exemplo de criptografia fim a fim.

Outro componente essencial na criptografia fim a fim é a proteção das chaves privadas, geralmente realizado por meio de um software denominado chaveiro criptográfico. O chaveiro tem como funções a proteção das chaves privadas, a importação de chaves públicas alheias, revogação de chaves comprometidas e configuração do nível de confiança. Sendo a camada responsável por estas tarefas, está fortemente ligado à facilidade de uso das chaves pelo usuário final. O chaveiro desempenha a função crucial de proteção das chaves privadas por meio de senha. A solução adotada em implementações como o GnuPG cria um chaveiro na estação de trabalho do usuário durante a instalação, que pode então ser usado diretamente por meio de linha de comando ou acessado por através de bibliotecas específicas por softwares de terceiros.

## Uso do padrão OpenPGP para criptografia

---

PGP é uma família de softwares da área de segurança desenvolvidos inicialmente por Philip R. Zimmermann [32] e liberada como um freeware em 1991 e atualmente é mantida pela PGP Corp, adquirida em 2010 pela Symantec. Tendo como base esta experiência foi desenvolvido o padrão OpenPGP, que contém a mesma proposta de criptografia por meio de chaves assimétricas, uma pública e outra privada, mas agora com uma especificação publicada na RFC 4880 - OpenPGP Message Format [5]. A publicação desta especificação permitiu o nascimento de implementações abertas. A mais conhecida para desktop é a GnuPG, ou simplesmente GPG, tanto que, por vezes, os termos PGP e GPG são usados de forma intercambiável.

Esse formato de comunicação estabelece o sigilo da mensagem e o não-repúdio [18] - incapacidade de uma das partes de negar que assinou a mensagem se, de fato, o fez - da mensagem, tudo isso mantendo as chaves privadas - o recurso que guarda o poder de assinar e, portanto, de identificação - em sigilo.

Essa tecnologia encontrou um forte caso de uso nas trocas de e-mail, impedindo que a interceptação das mensagens comprometesse seu sigilo e, que um terceiro pudesse se passar por um dos interlocutores de forma despercebida ou, ainda, que um dos interlocutores mais tarde negasse que ele assinou a mensagem.

Outro caso de uso bastante explorado é a assinatura de arquivos. Dado que uma assinatura precisa da senha do chaveiro do usuário somada à posse da chave privada ela pode ser usada com propósitos legais na assinatura de documentos digitais.

GPG está disponível para todos os grandes sistemas operacionais, de estações desktop até celulares e várias bibliotecas permitem desenvolvimento sobre esta tecnologia.

## **Análise de formas de acesso à chaveiros criptográficos**

---

Várias aplicações fazem uso do chaveiro PGP para melhorar a segurança dos seus recursos e proteger a comunicação entre os usuários. O chaveiro em si precisa ser capaz de fornecer os serviços discutidos anteriormente e, para isso, existem algumas formas utilizadas por aquelas aplicações. Segue uma análise de algumas dessas formas de acesso ao chaveiro PGP e algumas aplicações que empregam tal forma.

### **4.1 Acesso através de linha de comando e bibliotecas**

A obra de W. Lucas traz várias das funções de acesso à recursos PGP explicadas com rico detalhe, além de exemplos de uso [19]. A aplicação GnuPG [8] é uma das formas mais conhecidas de fazer uso de chaves PGP.

O GnuPG é uma implementação da do formato OpenPGP, que permite a geração e uso das chaves privadas em interface de linha de comando. Essa é a principal implementação do formato em uso atualmente, e é instalada por padrão em várias distribuições Linux. Também está disponível para Microsoft Windows por meio da suite Gpg4win. Essa implementação é uma ferramenta completa, contando com todas as operações esperadas de um chaveiro criptográfico.

Como exemplo de acesso ao chaveiro por linha de comando, segue como criptografar um arquivo textual com a chave pública de um usuário fictício, cujo e-mail é bob@email.com e a assinatura pelo usuário de e-mail alice@email.com. Bob representa quem receberá a mensagem e Alice será o emitente neste exemplo. O anexo XX traz também um pequeno passo-a-passo de como criar novos pares de chave e como descriptografar a mensagem deste exemplo.

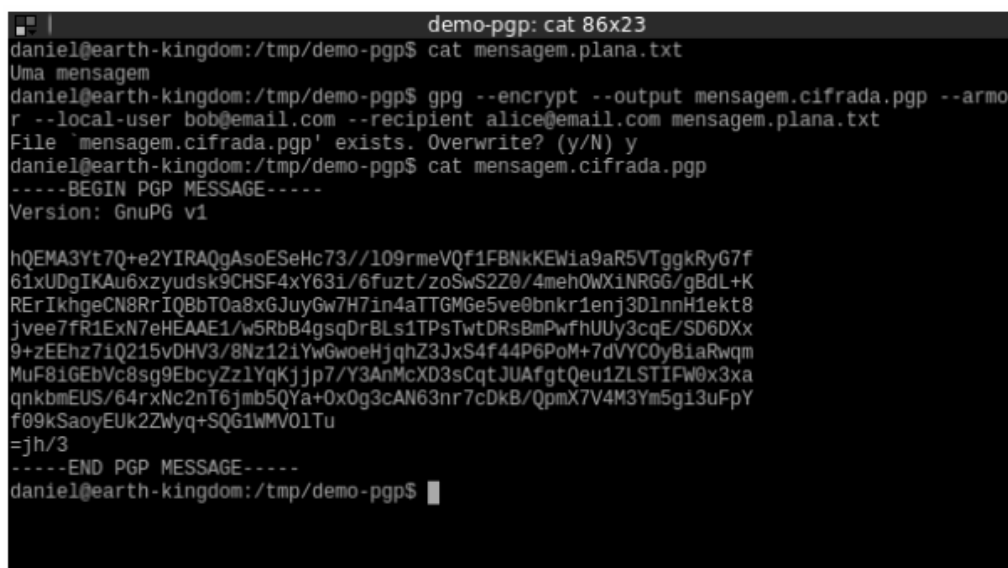
Para que esta chave seja usada para criptografar um conteúdo utilizamos o mesmo comando gpg, agora com a instrução `–encrypt`, seguido de `–local-user` indicando a chave do usuário emitente da mensagem e `–recipient`, que indica o usuário que receberá

a mensagem. No exemplo abaixo o usuário Bob encriptará uma mensagem para Alice. O comando completo será:

```
gpg --encrypt --output mensagem.cifrada.pgp \  
    --armor --local-user bob@email.com --recipient \  
    alice@email.com mensagem.plana.txt
```

No comando acima, os detalhes da operação são os seguintes:

- `--encrypt` : instruiu o gpg que se trata de uma operação de encriptação
- `--output`: informa o arquivo destino, onde será gravada a mensagem cifrada
- `--armor`: opcionalmente usada para que a mensagem cifrada esteja em formato ASCII
- `--local-user`: usuário local que está gerando a mensagem
- `--recipient`: usuário que receberá a mensagem. A sua chave pública deve ser conhecida no chaveiro.



```
demo-gpg: cat 86x23  
daniel@earth-kingdom:/tmp/demo-gpg$ cat mensagem.plana.txt  
Uma mensagem  
daniel@earth-kingdom:/tmp/demo-gpg$ gpg --encrypt --output mensagem.cifrada.pgp --armor  
--local-user bob@email.com --recipient alice@email.com mensagem.plana.txt  
File 'mensagem.cifrada.pgp' exists. Overwrite? (y/N) y  
daniel@earth-kingdom:/tmp/demo-gpg$ cat mensagem.cifrada.pgp  
-----BEGIN PGP MESSAGE-----  
Version: GnuPG v1  
  
hQEMA3Yt7Q+e2YIRAQgAsoESeHc73//109rmeVQf1FBNkKEWia9aR5VTggkRyG7f  
61xUDgIKAu6xzyudsk9CHSF4xY63i/6fuzt/zoSwS2Z0/4meh0WxiNRGG/gBdL+K  
RErIkhgeCN8RrIQBbT0a8xGJuyGw7H7in4aTTGMGe5ve0bnkr1enj3DlnnH1ekt8  
jvee7fR1ExN7eHEAAE1/w5RbB4gsqDrBLs1TPsTwtdRsBmPwFhUuy3cqE/SD6DXx  
9+zEEhz7iQ215vDhV3/8Nz12iYwGwoeHjqhZ3JxS4f44P6PoM+7dVYC0yBiaRwqm  
MuF81GEbVc8sg9EbcyZz1YqKj7p7/Y3AnMcXD3sCqtJUAfgtQeu1ZLSTIFW0x3xa  
qnkbmEUS/64rxNc2nT6jmb5QYa+Ox0g3cAN63nr7cDkB/QpmX7V4M3Ym5g13uFpY  
f09kSaoyEUK2ZWYq+SQG1wMV01Tu  
=jh/3  
-----END PGP MESSAGE-----  
daniel@earth-kingdom:/tmp/demo-gpg$
```

**Figura 4.1:** Exemplo de operação de criptografia sendo realizada por linha de comando

Essa forma de acesso pode ser explorada por qualquer aplicação que resida na máquina onde o GnuPG está instalado. Além da chamada pelo binário gpg, demonstrada anteriormente, o gpg instalado em estações de trabalho contém uma biblioteca padrão chamada GPGME, que oferece exposição de recursos da aplicação de forma padronizado para linguagens diversas. Em aplicações que fazem uso do chaveiro instalado localmente essas chamadas são abstraídas por diversas bibliotecas, de acordo com a linguagem desejada. A página de ferramentas do GnuPG trás alguns exemplos dessas bibliotecas [11]. Alguns exemplos são:

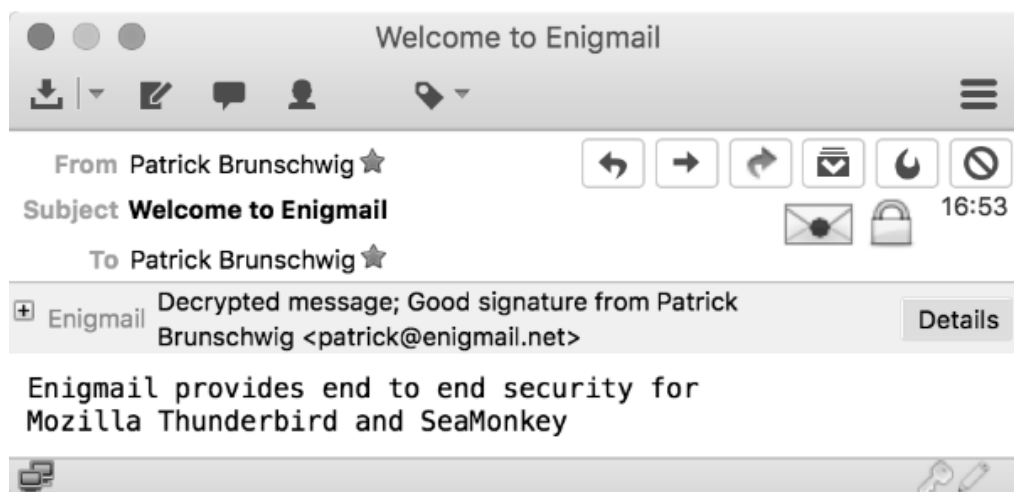
- gpgme - A biblioteca padrão fornecida pelo GnuPG, já citada [10]



- `gpg_encrypt()` - Função nativa da linguagem PHP [12]
- `py-gnupg` - Módulo para linguagem Python que faz interface com o GnuPG [23]
- `gnupg-for-java` - Biblioteca para linguagem Java que expõe os recursos da biblioteca `gpgme` [9]
- `ruby-gpgme` - Biblioteca para linguagem Ruby que expõe os recursos da `gpgme` [14]

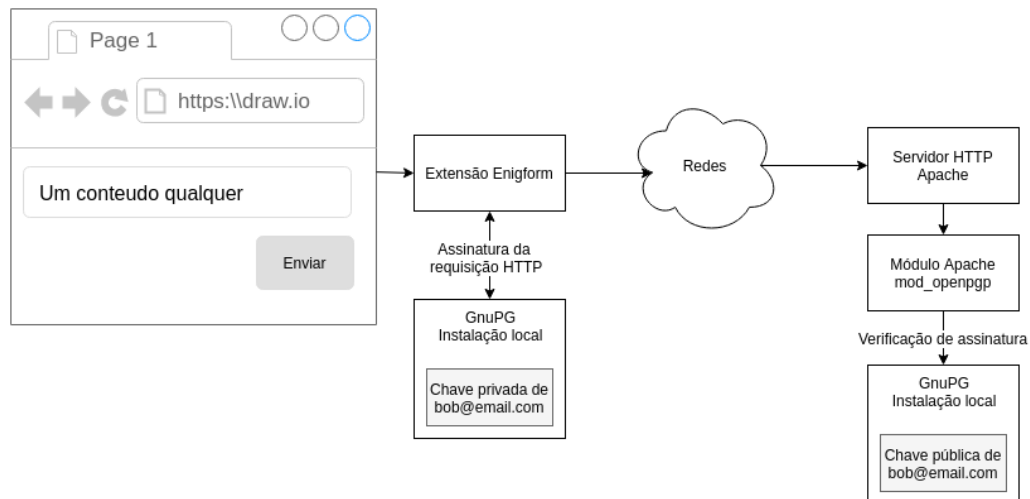
Algumas ferramentas conhecidas que fazem uso de bibliotecas como essas são o Enigmail, o Apache OpenPGP, o Claws Mail e o WebPG para Firefox. Em todos os casos a instalação oferecida na estação de trabalho é usada para realizar as operações necessárias junto ao chaveiro criptográfico.

O Enigmail é um plugin desenvolvido para o cliente de e-mail Mozilla Thunderbird. Este plugin estende as capacidades do Thunderbird dando-lhe a capacidade de encriptar, descriptar, assinar e verificar assinatura de e-mails. O Claws Mail é outro cliente de e-mail que também implementa os recursos de segurança usando o chaveiro local para proteger as mensagens.



**Figura 4.2:** Captura de tela a aplicação Enigmail, exibindo detalhes do recurso de descryptografia

O Apache OpenPGP [3] é uma extensão para o servidor HTTP Apache que permite que requisições HTTP sejam assinadas e verificadas pelo servidor. Ela faz par com a extensão Enigform [4] para o Firefox, que modifica as requisições usando a instalação local do chaveiro PGP.



**Figura 4.3:** Diagrama com a proposta de funcionamento do enigmail com o módulo openpgp para Apache HTTP

## 4.2 Token de Segurança Móvel

Tokens móveis são dispositivos capazes de oferecer recursos de segurança e ainda manter a capacidade de transporte desses recursos. Eles costumam carregar consigo senhas, dados biométricos ou chaves criptográficas.

Esses dispositivos são um conjunto de um token, com formato similar ao de um cartão de memória ou de um pendrive, e o software que é instalado na estação de trabalho que permite acessá-lo. Os dispositivos levam consigo as chaves privadas e públicas do usuário ou empresa, além de uma cadeia de certificados ligados à Autoridade Certificadora Raiz da ICP-Brasil [2]. Existe a possibilidade de usar esses recursos para criptografar conteúdo, realizar assinaturas digitais e mesmo encriptar conexões HTTPS, estabelecendo identificação entre aplicações remotas de forma confiável e juridicamente aceita.

Alguns dos tipos de tokens móveis são:

- Cartão e leitor de cartão criptográfico, como no e-CPF desenvolvido pela Serasa
- Token em formato de pendrive, conectado ao USB da estação de trabalho
- Token móvel, instalado em dispositivo móvel, como o MobileID, da Certisign
- Token não conectado, que gera um número secreto de tempo em tempo
- Token não conectado, que gera um número secreto mediante apresentação de senha pessoal

Uma das soluções mais conhecidas no Brasil é o e-CPF, para pessoa física, e o e-CNPJ, para pessoas jurídicas.



**Figura 4.4:** Exemplos de diversos tipos de token, como aqueles em formato de pendrive, cartão criptográfico e token desconectado, gerador de código secreto com e sem entrada de senha pessoal

Avaliando os dispositivos oferecidos pela Serasa, uma das maiores revendedoras do país, nota-se que não existe suporte para outros sistemas operacionais que não o Microsoft Windows a partir da versão XP [16]. O acesso às operações do token é feito a partir do software que o acompanha. Aplicações que desejam usar essa aplicação devem acessar as bibliotecas fornecidas por cada fabricante instaladas no sistema operacional.

A Certisign oferece, além desses modelos, uma versão para celular [7]. A aplicação está disponível para Windows, MacOS e Android. A criação e validação dos certificados acontece presencialmente e, com alguns códigos de segurança, o certificado pode ser emitido no dispositivo. O acesso à essa instalação no dispositivo móvel também é feito por meio de software da fabricante, de forma similar aos desenvolvidos pela Serasa.

Em ambos os casos apresentados, o acesso pode ser simplificado pelo seguinte diagrama:



**Figura 4.5:** Representação da forma de acesso aos tokens conectados

Alguns tokens podem ser usados de forma desconectada. Um exemplo no Brasil é utilizado por bancos para autenticar operações por internet banking. Um pequeno dispositivo gera um número aleatório que muda de tempo em tempo. Para realizar operações junto ao banco, o usuário deve fornecer o número que o token mostra naquele

momento. Somente se o sistema bancário autenticar a entrada fornecida pelo usuário a operação poderá continuar.

Nesse caso não há acesso direto ao token e também não é possível que diferentes aplicações usem seu sistema de segurança, já que a forma de geração do código secreto somente é conhecida pela instituição bancária, no exemplo citado.

## **4.3 Considerações**

Em várias das aplicações apresentadas é necessário que o usuário configure repetidamente as diversas estações de trabalho nas quais precisa usar os recursos do chaveiro criptográfico. A gestão das chaves já é, por si só, uma tarefa que lhe exigirá atenção. O uso de tokens móveis torna esta gestão mais simples, trazendo mobilidade ao chaveiro.

O desenvolvimento de outras aplicações que usem tais chaveiros também dependerá da configuração do ambiente, que se repete em cada estação de trabalho.

## **Proposta de chaveiro pessoal móvel acessível por Webservices**

---

### **5.1 Descrição da proposta**

Neste trabalho propõe-se uma solução para as dificuldades com configuração de ambiente e mobilidade do chaveiro, dando ao usuário a capacidade de manter consigo o chaveiro PGP com seus recursos privados de criptografia e, ainda assim, ser capaz de utilizar tais recursos nas aplicações com as quais interage cotidianamente em suas estações de trabalho.

O capítulo 4 apresentamos formas de acesso ao chaveiro estão concentradas em bibliotecas e outros softwares na estação de trabalho.. Buscamos neste trabalho oferecer uma interface única de acesso ao chaveiro por meio de webservices seguindo a filosofia REST. É esperado que isso simplifique a manutenção do chaveiro pelo usuário final. Também espera-se contribuir com o desenvolvimento de aplicações que usam os recursos do chaveiro, de forma que possam usar os webservices de forma centralizada e ubíqua e seja desnecessário ter a instalação em várias estações de trabalho espalhadas. cada estação de trabalho.

Para que aplicações de terceiros consigam realizar as operações necessárias, como criptografar, assinar, descriptografar e verificar assinatura, é necessário um canal de comunicação para que as aplicações localizadas nas estações de trabalho possam acessar o chaveiro e fazer uso dos recursos criptográficos agora protegidos enclausurados no dispositivo móvel. Para este fim, é proposta neste trabalho uma camada de serviço implementando estas operações como web services.

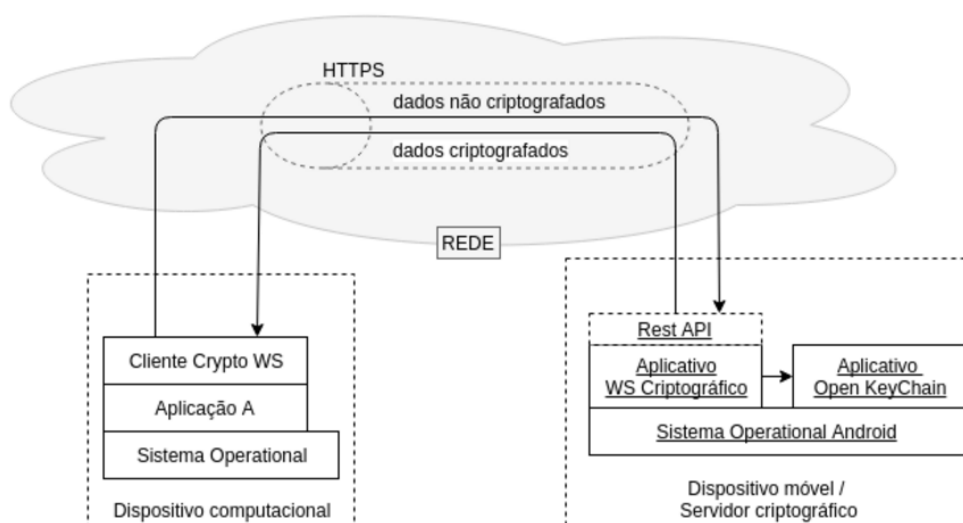
A principal vantagem do uso de web services é a simplicidade de uso por parte das aplicações clientes. Como este é um protocolo bem conhecido o desenvolvedor do cliente poderá usar bibliotecas já testadas para a sua linguagem de escolha. Somado a isto, o uso de REST com JSON torna esta API simples e transparente algo mais familiar para o desenvolvedor, visto que estas tecnologias são conhecidas e suportadas em todas as grandes linguagens de programação e são plenamente provadas no mercado.

Usando chamadas remotas de webservice será possível fazer uso dos recursos independente de uma instalação na máquina cliente dedicada à manipulação das chaves conhecidas, mantendo todo o arcabouço de segurança de interesse do usuário centralizado em seu dispositivo.

Para isso será usada uma implementação de chaveiro criptográfico disponível para Android chamada Open KeyChain [13]. Ela implementa um chaveiro PGP com uma interface simples para o usuário final. Essa ferramenta de código aberto dá suporte completo à geração de chaves privadas, importação de chaves públicas de diversos servidores comunitários, criptografia, descriptografia, assinatura e verificação de assinaturas no conteúdo desejado. Note-se que esta aplicação já se encontra pronta para uso na plataforma Android.

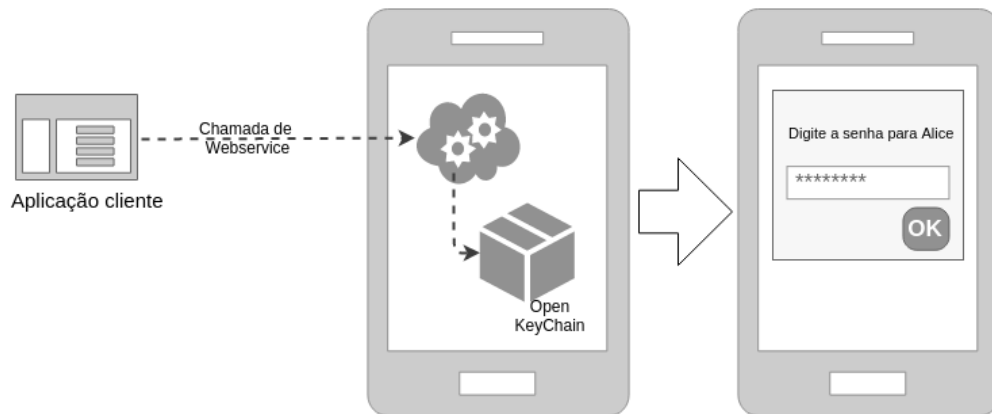
O escopo deste trabalho está em demonstrar a viabilidade do uso de um chaveiro criptográfico em dispositivo móvel por meio de consultas HTTP aos web services desenvolvidos, na forma de um software com serviços desenvolvidos para demonstrar essa viabilidade e testar a proposta.

Abaixo uma representação simplificada da proposta, exibindo os principais elementos envolvidos numa operação de criptografia:

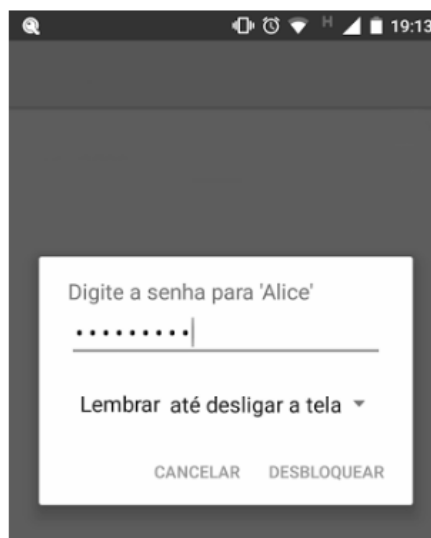


**Figura 5.1:** representação de uma operação de criptografia com chaveiro em dispositivo móvel

Quando necessário, será pedida na tela do usuário a senha de sua chave privada para realizar operações de assinatura e descriptografia.



**Figura 5.2:** ilustração de como a chamada de um serviço criptográfico aciona o Open KeyChain que, por sua vez, pede a senha do chaveiro do usuário para proceder



**Figura 5.3:** Captura de tela da aplicação Open KeyChain pedindo a senha da chave privada para o usuário *alice@email.com*

## 5.2 Operações dos Webservices

Os webservices respondem através de um servidor web em execução no dispositivo móvel. Aplicações clientes capazes de enviar mensagens no formato especificado à frente poderão usufruir dos recursos do chaveiro no celular.

A forma de implementação desta demonstração adota um modelo assíncrono para as operações de criptografia e descriptografia. Essa forma foi escolhida pois os aplicativos móveis são geralmente desenvolvidos explorando-se recursos assíncronos e o webservice precisa tratar os casos em que o usuário deve interagir com a tela sem bloquear

as chamadas ao serviço de forma indefinida. Dessa forma, contamos na demonstração deste trabalho com quatro operações básicas, sendo duas delas de pedido e duas de consulta de resultados destes. As operações são as seguintes:

- Pedido para criptografar um novo conteúdo endereçado à um destinatário
- Consulta a um pedido de criptografia anterior
- Pedido para descriptografar um novo conteúdo recebido
- Consulta a um pedido de descriptografia realizado anteriormente

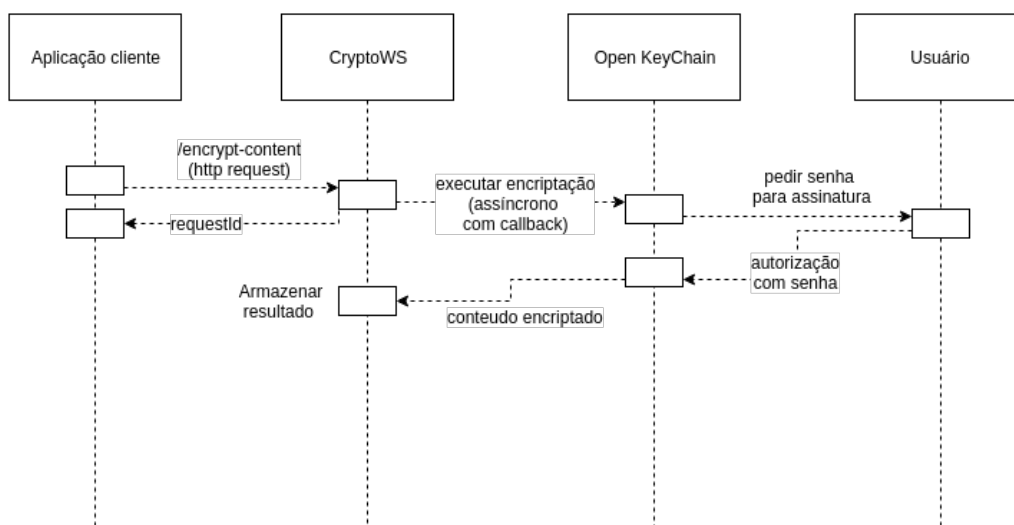
Seguem nas próximas subseções o detalhamento destes serviços.

### 5.2.1 Autorização para utilizar os serviços

Para que uma aplicação cliente utilize os serviços ela deve usar uma chave de segurança gerada no aplicativo móvel. Todas as requisições devem contar com o cabeçalho Authorization preenchido com uma chave aleatória, exposta no dispositivo do usuário. A aplicação cliente deve ser atualizada com a chave atual, do contrário as suas requisições serão negadas pelo serviço, com o código HTTP 403 - Unauthorized e uma mensagem indicando que a chave está vazia ou incorreta.

### 5.2.2 Criptografar um novo conteúdo

Esta operação demanda um conteúdo textual para ser encriptado e o e-mail do destinatário. A chave pública do destinatário deve estar importada na instalação do Open KeyChain no dispositivo móvel.



**Figura 5.4:** Funcionamento de uma requisição de encriptação com assinatura

- **Endpoint:** /cryptows/encrypt-content



- **Método HTTP:** POST
- **Content-type:** application/json

Para criptografar um novo conteúdo deve ser enviado um objeto JSON com os seguintes atributos:

- **content** (String) - A mensagem que será encriptada.
- **receiver** (String) - E-mail do destinatário da mensagem. Sua chave pública deve estar no chaveiro do Open KeyChain

---

**Código 5.1** Mensagem do pedido de criptografia

---

```
1  {  
2      "content": "Mensagem para ser encriptada",  
3      "receiver": "bob@email.com"  
4  }
```

---

A resposta deste serviço em caso de sucesso fará uso do código HTTP 200 - OK indicando que o processamento foi iniciado. A aplicação cliente receberá uma resposta com o identificador desta ação. Ele deve ser guardado e usado posteriormente para consultar o resultado do pedido de criptografia. O formato dessa resposta conterá os seguintes dados, em formato JSON:

- **requestId** (String) - Identificador gerado para consulta posterior do processamento da encriptação
- **code** (Número inteiro) - Código HTTP referente ao sucesso ou falha da operação
- **time** (Número / Timestamp) - Timestamp do momento do recebimento da requisição pelo aplicativo no dispositivo

---

**Código 5.2** Resposta do pedido de criptografia

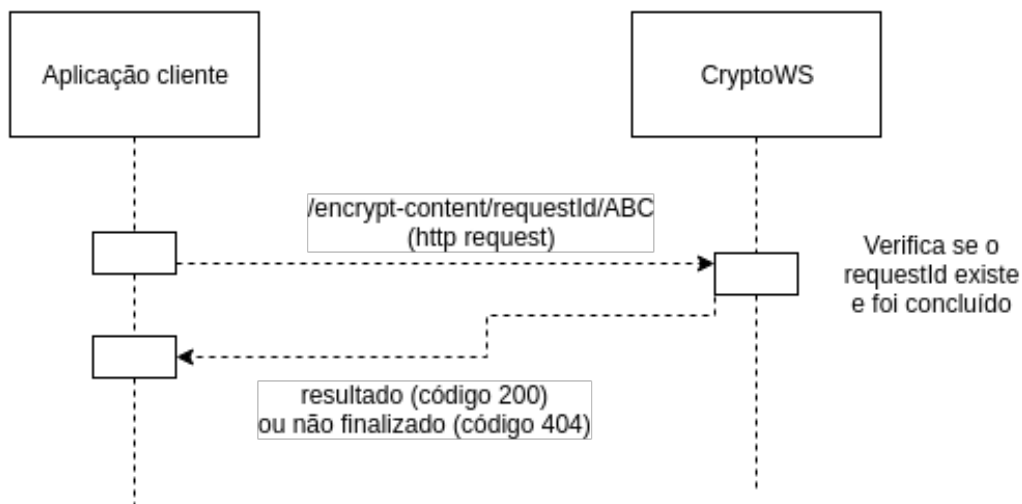
---

```
1  {  
2      "requestId": "123e4567-e89b-12d3-a456-4266",  
3      "code": 200,  
4      "time": 1480902494377  
5  }
```

---

### 5.2.3 Consultar resultado da criptografia de conteúdo

Esta operação somente exige o identificador do do pedido anterior de criptografia, chamado na resposta daquele pedido de requestId. Esse identificador deve ser informado na URL de consulta.



**Figura 5.5:** Diagrama com o fluxo de consulta do resultado de uma operação de criptografia

- **Endpoint exemplo:** /cryptows/encrypt-content/requestId/123e4567-e89b-12d3
- **Método HTTP:** GET
- **Content-type:** application/json

A resposta deste serviço em caso de sucesso fará uso do código HTTP 200 - OK indicando que o processamento foi concluído. Caso a aplicação no celular ainda esteja processando a requisição, o código HTTP 404 - Not Found é retornado. A aplicação cliente deve tentar novamente em alguns segundos. Em caso de retorno com sucesso a mensagem criptografada será descartada pelo serviço para poupar recursos do dispositivo móvel. O formato dessa resposta conterà os seguintes dados, em formato JSON:

- **cipherContent** (String) - Mensagem criptografada com a chave pública do usuário informado em receiver, no ato da requisição de criptografia
- **code** (Número inteiro) - Código HTTP referente ao sucesso ou falha da operação
- **statusMessage** (String) - Mensagem auxiliar do resultado da operação. Pode conter detalhes em caso de erros durante o processamento.

---

**Código 5.3** Resposta do pedido de criptografia em caso de processamento concluído

---

```

1  {
2      "cipherContent": "-----BEGIN PGP MESSAGE-----\nVersion: GnuPG v1.4.5\n
3  hQEOA1e+1x6YuUMCEAP9F2 ...",
4      "code": 200,
5      "statusMessage": "Requisição processada com sucesso"
6  }
  
```

---

**Código 5.4** Resposta da consulta à pedido de criptografia ainda em andamento

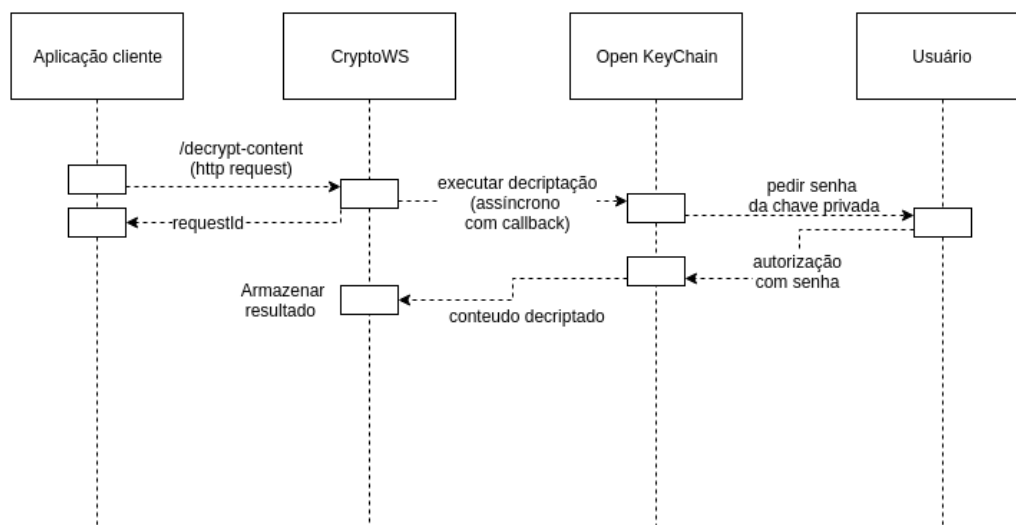
```

1  {
2      "cipherContent": "",
3      "code": 404,
4      "statusMessage": "O processamento ainda não foi concluído."
5  }

```

### 5.2.4 Descriptografar uma mensagem PGP recebida

Esta operação demanda uma mensagem em formato OpenPGP em ASCII Armour para uma tentativa de descriptação. O formato ASCII Armour permite que a mensagem possa ser enviada com formato JSON sem mais processamento, visto que JSON não suporta dados binário. Essa operação somente é possível se o Open KeyChain contém a chave privada equivalente à chave pública utilizada para criptografia. O conteúdo não necessariamente precisa ter sido encriptado usando o webservice apresentado no item 6.1.1, uma vez que o padrão OpenPGP não leva isso em conta.



**Figura 5.6:** Funcionamento de uma requisição de encriptação com assinatura

- **Endpoint:** /cryptows/decrypt-content
- **Método HTTP:** POST
- **Content-type:** application/json

Para descriptografar um novo conteúdo deve ser enviado um objeto JSON com os seguintes atributos:

- **content** (String / Mensagem OpenPGP) - A mensagem que deve ser descriptada com uma das chaves privadas conhecidas

---

**Código 5.5** Mensagem enviada para descriptografar uma mensagem OpenPGP

---

```
1  {
2    "content": "-----BEGIN PGP MESSAGE-----\nVersion:\n
3    GnuPG v1.4.5 hQEOA1e+1x6YuUMCEAP9F2 ..."
4  }
```

---

A resposta deste serviço em caso de sucesso fará uso do código HTTP 200 - OK indicando que o processamento foi iniciado. A aplicação cliente receberá uma resposta com o protocolo desta ação. Ele deve ser guardado e usado posteriormente para consultar o resultado do pedido de criptografia. O formato dessa resposta conterá os seguintes dados, em formato JSON:

- **requestId** (String) - Identificador gerado para consulta posterior do processamento da decriptação
- **code** (Número inteiro) - Código HTTP referente ao sucesso ou falha da operação
- **time** (Número / Timestamp) - Timestamp do momento do recebimento da requisição pelo aplicativo no dispositivo

---

**Código 5.6** Exemplo de resposta do pedido de descriptografia

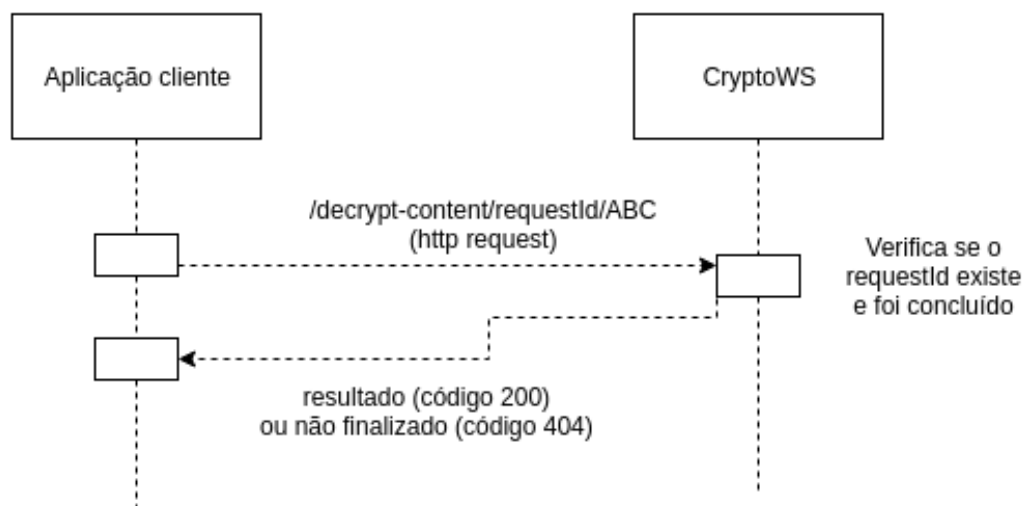
---

```
1  {
2    "requestId": "123e4567-e89b-12d3-a456-426655440000",
3    "code": 200,
4    "time": 1480902494377
5  }
```

---

### 5.2.5 Consultar resultado da descriptografia de conteúdo

Esta operação somente exige o identificador do do pedido anterior de criptografia, chamado na resposta daquele pedido de requestId. Esse identificador deve ser informado na URL de consulta.



**Figura 5.7:** Diagrama com o fluxo de consulta do resultado de uma operação de criptografia

- **Endpoint exemplo:** /cryptows/decrypt-content/requestId/123e4567-e89b-12d3
- **Método HTTP:** GET
- **Content-type:** application/json

A resposta deste serviço em caso de sucesso fará uso do código HTTP 200 - OK indicando que o processamento foi concluído. Caso a aplicação no celular ainda esteja em processamento a requisição, o código HTTP 404 - Not Found é retornado. A aplicação cliente deve tentar novamente em alguns segundos. O formato dessa resposta conterà os seguintes dados, em formato JSON:

- **plainContent** (String) - Mensagem descriptografada com a chave privada do usuário.
- **code** (Número inteiro) - Código HTTP referente ao sucesso ou falha da operação
- **statusMessage** (String) - Mensagem auxiliar do resultado da operação. Pode conter detalhes em caso de erros durante o processamento.

---

**Código 5.7** Exemplo de resposta em caso de processamento concluído

---

```

1  {
2      "plainContent": "Uma mensagem que foi enviada",
3      "code": 200,
4      "statusMessage": "Requisição processada com sucesso"
5  }
  
```

---

---

**Código 5.8** Exemplo de resposta em caso de processamento ainda em andamento

---

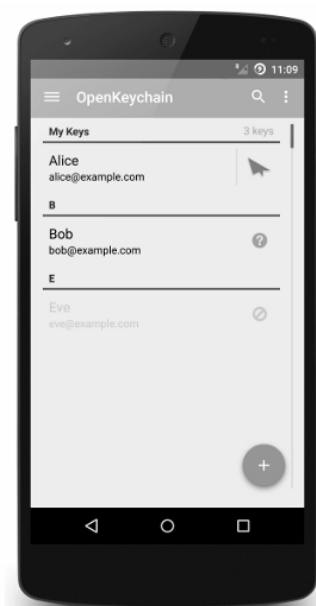
```
1  {  
2      "plainContent": "",  
3      "code": 404,  
4      "statusMessage": "O processamento ainda não foi concluído."  
5  }  
6
```

---

## Experimento

Para verificar a viabilidade da proposta de serviços criptográficos funcionando a partir de um dispositivo móvel, foi desenvolvido neste trabalho de conclusão um aplicativo que contém algumas funcionalidades demonstrativas de tais serviços e uma aplicação cliente para verificação do seu funcionamento de forma simplificada.

A aplicação desenvolvida faz uso dos serviços de uma outra aplicação Android chamada Open KeyChain. Ela é em uma carteira de chaves PGP com uma interface simples para o usuário final.



**Figura 6.1:** *Captura de tela da aplicação OpenKey Chain, com duas chaves privadas de exemplo*

O protocolo de comunicação e troca de mensagens segue a especificação apresentada anteriormente. As operações foram implementadas em um aplicativo Android de modo a demonstrar de forma minimizada a viabilidade da proposta.

## 6.1 Implementação dos serviços para o chaveiro PGP

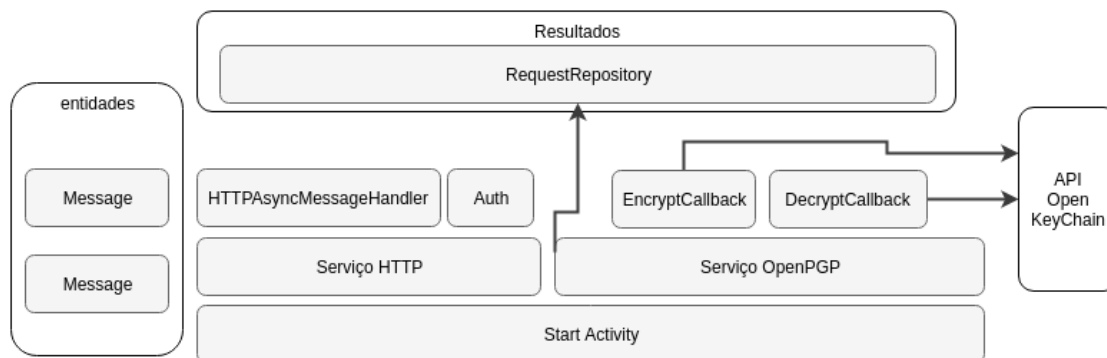
Os serviços PGP foram implementados a partir de um aplicativo Android seguindo as especificações da proposta no capítulo anterior. Ao iniciar o aplicativo exibe seu IP e porta para os serviços, que devem ser configurados em aplicações clientes para a correta invocação dos serviços.



**Figura 6.2:** Captura de tela da aplicação desenvolvida neste experimento.

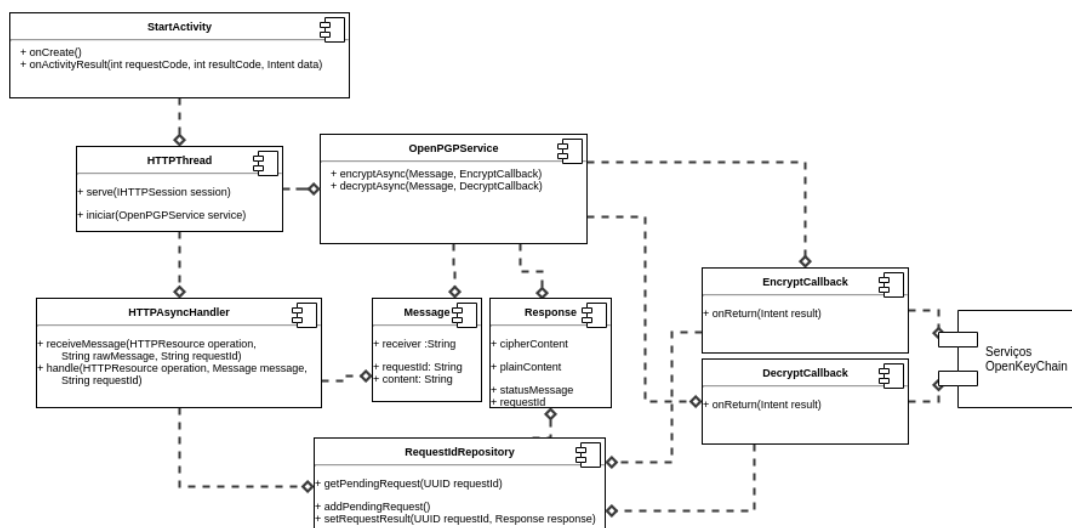
Para sua implementação foram usadas os seguintes itens de configuração detalhados no anexo 6. O dispositivo móvel usado no desenvolvimento e testes está detalhado no anexo 5. Os detalhes do ambiente de desenvolvimento utilizado em todos os casos está no anexo 7. O anexo 8 contém um exemplo de aplicação cliente capaz de consumir os serviços fornecidos a partir do dispositivo móvel.





**Figura 6.3:** Ilustração em alto nível das camadas envolvidas na implementação dos webservices em dispositivo móvel

A arquitetura adotada pode ser representada pelo seguinte diagrama, as principais entidades da aplicação:



**Figura 6.4:** Principais entidades que compõem a aplicação desenvolvida no experimento

Legenda:

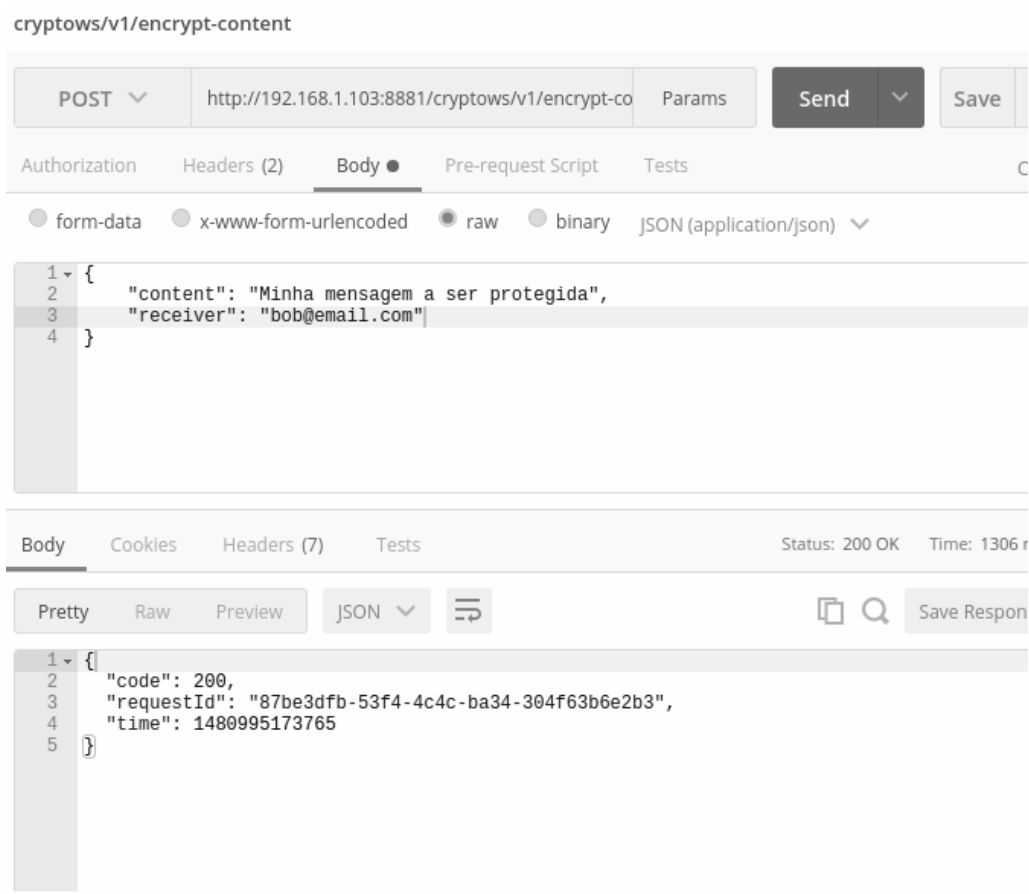
São destacados nesta implementação:

- **Serviço HTTP:** responsável por lidar com novas requisições e despachar respostas de requisições de consulta. Caso o resultado já esteja disponível em RequestRepository, este serviço o retorna para a aplicação cliente. Também é responsável por validar a chave de acesso que deve ser informada pelas aplicações clientes para fazer uso do serviço.
- **Serviço OpenPGP:** contém as rotinas de operações criptográficas. Gerencia o que está disponível para o serviço HTTP quando este precisa realizar chamadas de ações criptográficas.
- **EncryptCallback** - Classe que lida com os eventos de criptografia

- **DecryptCallback** - Classe que lida com os eventos de descriptografia
- **Message**: representa toda mensagem de entrada, seja no pedido de uma operação criptográfica, seja na consulta do resultado de seu processamento
- **Response**: representa todas as saídas da aplicação. Contém o resultado da operação requisitada, além de poder transportar o requestID quando gerado e os conteúdos, tanto criptografados quanto em formato plano/original.

## 6.2 Demonstração do consumo dos serviços

Abaixo seguem algumas capturas de tela da ferramenta Postman em sua versão 4.9.0 [29], demonstrando como o consumo dos webservices pode ser feito. Note-se que esta ferramenta somente faz consultas à APIs e mostra o resultado das suas chamadas. Esta demonstração busca expor a versatilidade dos webservices desenvolvidos, uma vez que qualquer aplicação capaz de chamadas REST poderia fazer uso desses recursos.



**Figura 6.5:** *Requisição de criptografia de conteúdo usando o web-service criptográfico*

cryptows/v1/encrypt-content/requestId/000

GET 192.168.1.103:8881/cryptows/v1/encrypt-content/requestId/87be3df1 Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Code

Authorization c3df13d6 Bulk Edit Presets

key value

Body Cookies Headers (7) Tests Status: 200 OK Time: 188 ms

Pretty Raw Preview JSON Save Response

```

1 {
2   "cipherContent": "-----BEGIN PGP MESSAGE-----\n\nnhQGMaZ77AnhVroT+AQwAh85KqKvJ/W6LVmuAxw
   /zhDtuPpVw5bAevf61NuWc0ay\NuaM9kbvF1Dr94qFJt5kbjR0fo8y0BKdyUaYdt0G1U1vmAw00KgV0QCjfbwu1zx2b\NEPr0
   xFJttAms8ctXMuFhIgzUQbbz1mUhmxcq4j50UjcwLNPdMLBY1DyfnZIAMa0L\NPRJrxknZFEXQH7sz1iBx8A207TsPr
   +evPCARFYU1SL47+VoqHXoIb52y4Sw70Cj/\N0sLH0kq1vQ1bzUqF7/CUH
   /AztMeiYch0YwBQAdc6DgRHTS16WQPX1osFF0pIcNc\NrhATiv08tdm6kLAT6wDQ5aNPq3BjIOF7ZEJdLjcy5suFMNCrZgiZ
   dqys03PgTF1p\NsRTcZY8pMM25d/k36V68Ada84eD11nvWK7L1B199c0XcZ10P310ZI5RBf678d6ju\NnuStoFhJ6nfTnJj
   /gHimj0wFu4MzK6rtizG0SqbJMGyZbq0kP7Qh0Y
   /waxccKX9gg\NmqHnV37e3iNI8HVXXwc9hQGMa7W9kGgCrhVjAQwAiDEeRam2XcOpEwe0JLGz0ey4\ngFTqodUy0DKAKy3VEC
   z004piPHSfST7dsW2Q0gQ1MikR27kjv58XzeyLXELfaLEX\NubhNQzYZHc1KXaYRQGF80trX132zYFigt8srFeXuhBmFCMeTb
   xUwpD2dQMXOLbJD\Ny1mVYqa3XdTI3S8Su3YhydXgFYk8eAh3xi1E
   /UWi0kq95aXS1qYhhbvrF1Aabqa\NjWU1DIvfUGxLJ73qvkWkj3Yx0JrWsv7sM875wpHYizqCt+J8gcb7
   +i7aRWGIwe05\NAeB9zHZpTQITxLXpMCsCtQVUkuZAEYwNThyF6THPJKVdpaBGFC1YFoYqyj1Z\N6N6mkMJPUPJPopy3B
   9EvxCYHafr81+cM53Wr+GofIzb3PUWFyw4JcoXmR9octL\N81w0J3PrCjTV1YU9Wfg3iER68BmxDnr4DJHY+I6n0FYa
   +081krxVhP/prmTAxgrBvN1QKJ/p1
   +ueMkFmxDKr4m4AVZyR2TUIYKGKAF8oYE0sFdAcvwsxb1cVdNgGm7u1G\NvFDcHGKgDif2B7rHqv71EVnPdQLrrnLAUYjr7Z
   Z8vPTU02Nd/JAGAR6HxyxwTK5\N41TIGxenH9IeIkV1aZSTsDns8Wmo62qE9XTtoJGcGq9LqErao
   /y1jxdbcIkfr5nZ\Nn4cp06Ymv3zF22uSaP6pMoqyqbWniXiDjE6CDmQ1EK0Q4VrkDmVpdDNVdnk2815\NsiawAPrhFWPS3m
   2YrEdg4L/grCp66bnn1bL6MGzcB2UADt8FRGHwMQTP1jU0PaxV\NzaP7p0h

```

**Figura 6.6:** Consultando resultado de um pedido de criptografia com o requestId

cryptows/v1/decrypt-content

POST 192.168.1.103:8881/cryptows/v1/decrypt-content Params Send Save

Authorization Headers (2) Body Pre-request Script Tests Code

form-data x-www-form-urlencoded raw binary JSON (application/json)

1 "content": "-----BEGIN PGP MESSAGE-----\n\nnhQGMaZ77AnhVroT+AQwAh85KqKvJ/W6LVmuAxw
 /zhDtuPpVw5bAevf61NuWc0ay\NuaM9kbvF1Dr94qFJt5kbjR0fo8y0BKdyUaYdt0G1U1vmAw00KgV0QCjfbwu1zx2b\NEPr0
 xFJttAms8ctXMuFhIgzUQbbz1mUhmxcq4j50UjcwLNPdMLBY1DyfnZIAMa0L\NPRJrxknZFEXQH7sz1iBx8A207TsPr
 +evPCARFYU1SL47+VoqHXoIb52y4Sw70Cj/\N0sLH0kq1vQ1bzUqF7/CUH
 /AztMeiYch0YwBQAdc6DgRHTS16WQPX1osFF0pIcNc\NrhATiv08tdm6kLAT6wDQ5aNPq3BjIOF7ZEJdLjcy5suFMNCrZgiZ
 dqys03PgTF1p\NsRTcZY8pMM25d/k36V68Ada84eD11nvWK7L1B199c0XcZ10P310ZI5RBf678d6ju\NnuStoFhJ6nfTnJj
 /gHimj0wFu4MzK6rtizG0SqbJMGyZbq0kP7Qh0Y
 /waxccKX9gg\NmqHnV37e3iNI8HVXXwc9hQGMa7W9kGgCrhVjAQwAiDEeRam2XcOpEwe0JLGz0ey4\ngFTqodUy0DKAKy3VEC
 z004piPHSfST7dsW2Q0gQ1MikR27kjv58XzeyLXELfaLEX\NubhNQzYZHc1KXaYRQGF80trX132zYFigt8srFeXuhBmFCMeTb
 xUwpD2dQMXOLbJD\Ny1mVYqa3XdTI3S8Su3YhydXgFYk8eAh3xi1E
 /UWi0kq95aXS1qYhhbvrF1Aabqa\NjWU1DIvfUGxLJ73qvkWkj3Yx0JrWsv7sM875wpHYizqCt+J8gcb7

Body Cookies Headers (7) Tests Status: 200 OK Time: 8316 ms

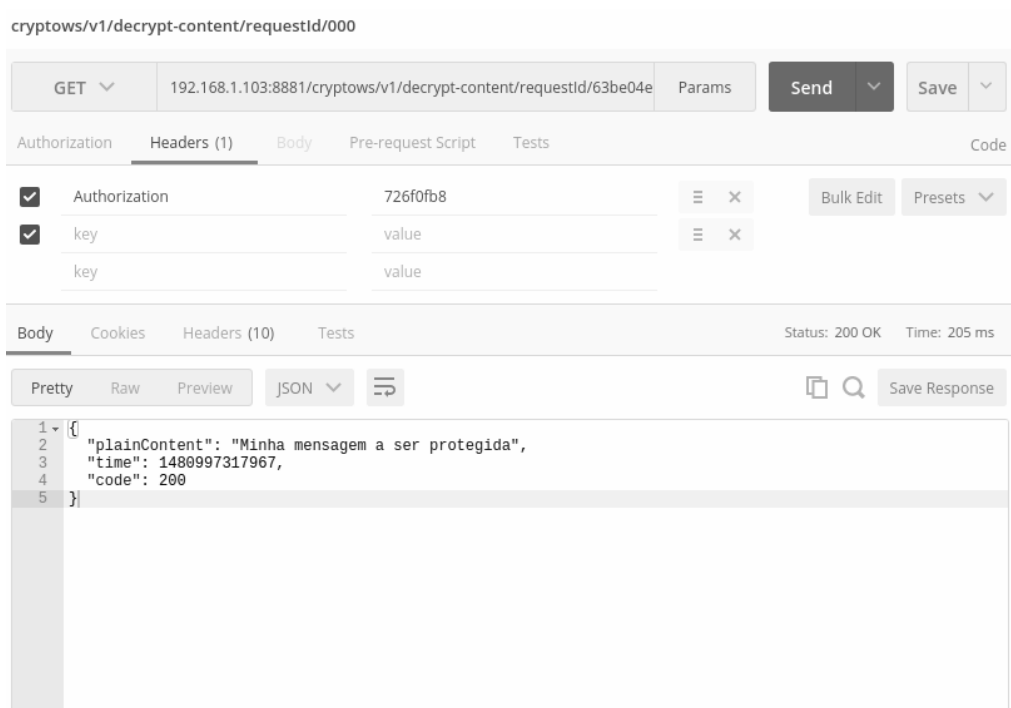
Pretty Raw Preview JSON Save Response

```

1 {
2   "code": 200,
3   "requestId": "be84301f-a625-46bf-9aa5-adc2da905cc8",
4   "time": 1480995765911
5 }

```

**Figura 6.7:** Executando pedido de descryptografia de conteúdo



**Figura 6.8:** Executando pedido de descriptografia de conteúdo

## 6.3 Análise de desempenho da solução

## 6.4 Análise de vulnerabilidades da solução

A solução se propõe a funcionar em redes potencialmente expostas, como redes domésticas ou mesmo corporativas. Nesses cenários se mostra imperativo o uso de HTTPS na comunicação, para evitar ataques de interceptação das mensagens quando ainda trafegam em forma não protegida. Na falta de implementações que usem HTTPS um roteador comprometido, por exemplo, poderia capturar o conteúdo plano das mensagens, tornando todo o esforço em vão. Poderia ser capturada, por exemplo, a senha de acesso daquela sessão, criando uma condição de acesso indevido, mas impossível de identificar pela aplicação. A versão da implementação explorada no experimento não conta com HTTPS, mas a biblioteca utilizada, o NanoHttpd, possui suporte para introdução de certificados nas requisições, tornando-a segura.

É possível, por exemplo, que aplicações que implementam webservices como especificado em seções anteriores gerem certificados próprios, diferentes em cada dispositivo de instalação. A conexão SSL então poderia ser estabelecida, desde que a aplicação cliente confiasse no certificado utilizado pelo serviço no dispositivo. A forma de manter este certificado seguro em caso de comprometimento do dispositivo é, em si, um desafio importante.

Outro ponto importante é a própria capacidade do usuário em comprometer o chaveiro memorizando as senhas. O Open KeyChain memoriza senhas informadas recentemente. Isso traz praticidade, pois múltiplos pedidos por criptografia seriam atendidos sem incomodar novamente o usuário, mas poderia gerar riscos do serviço. A implementação também não tem como forçar atualmente que o usuário use senhas fortes nos seus chaveiros ou que as compartilhe, o que pode comprometer a segurança desta solução se alguém tiver acesso indevido ao dispositivo.

É preciso também observar que a implementação de proteção contra ataques de negação de serviço (DoS) seria necessária em uma solução final. Como este trabalho preocupa-se em demonstrar a viabilidade da implementação a solução para esse problema não foi buscada.

Um último ponto é a forma mais simples de comprometimento vislumbrada: que o usuário perca o dispositivo e alguém o use para se passar por ele. Nesse ponto é importante notar que o Open KeyChain já oferece a capacidade de exportar as chaves privadas e, com elas, é possível gerar certificados de revogação, que informam à rede de confiança de chaves que aquela chave não é mais confiável. Essa ação, entretanto, não é tão simples e, por não ser obrigatória, pode ser esquecida por um usuário.

## Trabalhos futuros

---

Um possível ponto de trabalhos futuros está no estudo do uso prático desta técnica em diferentes tipos, topologias de rede e meios de transmissão, como bluetooth, NFC ou outras tecnologias de transmissão de dados. Também é ponto de experimento futuro verificar, dentro da diversidade de cenários em que uma mensagem precisa ser protegida, quais encontram na proposta deste trabalho o melhor caso de aplicação.

A implementação atual não provê proteção contra ataques de negação de serviço. A solução deste problema para serviços contidos em dispositivos móveis também é de interesse para trabalhos futuros.

Também são pontos interessantes de evolução a implementação destes serviços com proteção HTTPS, além de tratamentos para casos em que o usuário perde o dispositivo sem ter criado antes um certificado de revogação para suas chaves privadas.

## Conclusão

---

Neste trabalho foi proposto um modelo de web services para disponibilizar os serviços de um chaveiro criptográfico seguindo padrão PGP em dispositivo móvel. Como apresentado no experimento, tal implementação é viável, e espera-se que isso possa servir como uma nova opção de acesso, oferecendo maior comodidade para o usuário, permitindo maior mobilidade e segurança criptográfica fim-a-fim, com o máximo possível de controle do usuário de seu chaveiro criptográfico. Esse modelo pode contribuir com a facilidade da manutenção centralizadas das chaves PGP do usuário, fazendo uso de seu celular, e com o desenvolvimento de aplicações que usem esses recursos, fazendo acesso de uma forma simples e bem testada no mercado. Esses fatores conjuntos podem contribuir para a adoção da criptografia nas aplicações do dia-a-dia do usuário.

---

## Referências Bibliográficas

---

- [1] AKRAM, R. N.; KO, R. K. **End-to-end secure and privacy preserving mobile chat application**. In: *IFIP International Workshop on Information Security Theory and Practice*, p. 124–139. Springer, 2014.
- [2] BRASIL, I. **Instituto nacional de tecnologia da informação**. <http://www.iti.gov.br/icp-brasil/estrutura>. Acessado em: 2016-12-03.
- [3] BUSLEIMAN, A. B. **Apache OpenPGP Support**. <http://www.securiteam.com/tools/5HP0220LFU.html>, 2007.
- [4] BUSLEIMAN, A. B. **Enigform project main page**. <http://enigform.mozdev.org/>, 2016.
- [5] CALLAS, J.; DONNERHACHE, L.; FINNEY, H.; SHAW, D.; THAYER, R. **Openpgp message format**. RFC 4880, RFC Editor, November 2007. <http://www.rfc-editor.org/rfc/rfc4880.txt>.
- [6] CARMO, F. J.; LEMES, P. A.; FREITAS, T. H. **Criptografia e PGP**.
- [7] CERTISIGN. **Instalação de tokens fabricados pela certisign**. <https://www.certisign.com.br/sistemas>. Acessado em: 2016-12-03.
- [8] COMMUNITY, G. **Gnupg home page**. <https://www.gnupg.org/>. Acessado em: 2016-10-10.
- [9] COMMUNITY, G. **Gnupg home page**. <https://www.gnupg.org/>. Acessado em: 2016-10-10.
- [10] COMMUNITY, G. **Gnupg made easy**. [https://www.gnupg.org/\(es\)/related\\_software/gpgme/index.html](https://www.gnupg.org/(es)/related_software/gpgme/index.html). Acessado em: 2016-12-02.
- [11] COMMUNITY, G. **Gnupg software list**. [https://www.gnupg.org/related\\_software/swlist.html](https://www.gnupg.org/related_software/swlist.html). Acessado em: 2016-12-01.
- [12] COMMUNITY, G. **Manual do php gnupg gnupg\_encrypt**. [http://php.net/manual/pt\\_BR/function.gnupg-encrypt.php](http://php.net/manual/pt_BR/function.gnupg-encrypt.php). Acessado em: 2016-12-03.



- [13] COMUNITY, O. K. **Open keychain**. <https://www.openkeychain.org>.
- [14] COMUNITY, R. G. **Ruby gpgme**. <http://www.rubydoc.info/gems/gpgme>. Acessado em: 2016-12-03.
- [15] DIFFIE, W. **The first ten years of public-key cryptography**. *Proceedings of the IEEE*, 76(5):560–577, 1988.
- [16] EXPERIAN, S. **Instalação e requisitos técnicos em tokens da serasa experian**. <https://serasa.certificadodigital.com.br/instalacao/requisitos-tecnicos/>. Acessado em: 2016-12-03.
- [17] HACHIMORI, M. **Simplicial complex library**. [www.qci.jst.go.jp/~hachi](http://www.qci.jst.go.jp/~hachi).
- [18] LEHTONEN, S.; PARSSINEN, J. **Pattern language for cryptographic key management**. In: *EuroPLoP*, p. 245–258, 2002.
- [19] LUCAS, M. **PGP & GPG: Email for the practical paranoid**. No Starch Press, 2006.
- [20] McDONALD, N. G. **Past, present, and future methods of cryptography and data encryption**. *Department of Electrical and Computer Engineering, University of Utah*, 2009.
- [21] NAYLOR, D.; FINAMORE, A.; LEONTIADIS, I.; GRUNENBERGER, Y.; MELLIA, M.; MUNAFÒ, M.; PAPAGIANNAKI, K.; STEENKISTE, P. **The cost of the s in https**. In: *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, p. 133–140. ACM, 2014.
- [22] OPEN KEYCHAIN ABOUT PAGE. **About**. <https://www.openkeychain.org/about/>, 2016.
- [23] PYTHON-GNUPG COMUNITY. **python-gnupg - a python wrapper for gnupg**. <https://pythonhosted.org/python-gnupg/>. Acessado em: 2016-12-03.
- [24] RUSHE, D. **Google: don't expect privacy when sending to gmail**. *The Guardian*. Retrieved Dec, 19:2014, 2013.
- [25] SHENG, S.; BRODERICK, L.; KORANDA, C. A.; HYLAND, J. J. **Why johnny still can't encrypt: evaluating the usability of email encryption software**. In: *Symposium On Usable Privacy and Security*, p. 3–4, 2006.
- [26] SMITH, R. F. **The 3 pillars of information security**. <http://windowsitpro.com/security/3-pillars-information-security>, 2005. Acessado em: 2016-10-10.

- [27] SWEIKATA, M.; WATSON, G.; FRANK, C.; CHRISTENSEN, C.; HU, Y. **The usability of end user cryptographic products.** In: *2009 Information Security Curriculum Development Conference*, p. 55–59. ACM, 2009.
- [28] SYMANTEC. **Whitepaper - keeping your private data secure.** <https://www.symantec.com/content/dam/symantec/docs/white-papers/keeping-your-private-data-secure-en.pdf>, 2015. Acessado em: 2016-10-10.
- [29] TEAM, P. **Postman.** <https://www.getpostman.com>. Acessado em: 2016-12-03.
- [30] THE GUARDIAN. **Whatsapp adds end-to-end encryption using text-secure.** <https://www.theguardian.com/technology/2014/nov/19/whatsapp-messaging-encryption-android-ios>, 2014.
- [31] WHATSAPP. **Whatsapp encryption overview.** <https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf>.
- [32] ZIMMERMANN, P. R. **Creator of pgp and co-founder of silent circle.** <http://philzimmermann.com/EN/background/index.html>. Acessado em: 2016-10-10.
- [33] ZIMMERMANN, P. R. **Significant moments in pgp's history: Zimmermann case dropped.** [http://philzimmermann.com/EN/news/PRZ\\_case\\_dropped.html](http://philzimmermann.com/EN/news/PRZ_case_dropped.html), 1996. Acessado em: 2016-10-10.

## Informações históricas do PGP

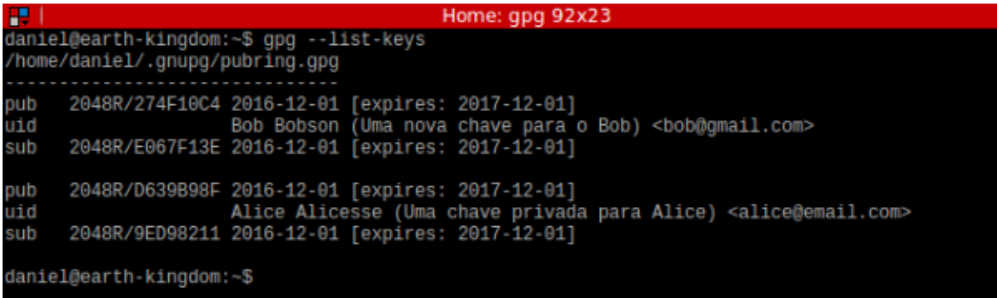
---

PGP é uma família de softwares da área de segurança desenvolvidos inicialmente por Philip R. Zimmermann [32] e liberada como um freeware em 1991 e atualmente é mantida pela PGP Corp, adquirida em 2010 pela Symantec. Essa liberação foi motivo de processos movidos pelo Governo dos Estados Unidos sob a acusação de ferir as leis de exportação de tecnologia criptográfica vigentes. O caso foi encerrado em 1996 sem prejuízo à Zimmermann [33]. Foi então fundada a PGP Inc - mais tarde PGP Corp - com o objetivo de manter essa tecnologia. Esta empresa foi adquirida em 2010 pela Symantec e sua versão gratuita deixou de ser oferecida.

## Demonstração de criação de chave e descriptografia com GnuPG

Como exemplo do acesso ao chaveiro por linha de comando, a seguir teremos o exemplo da geração de um novo par de chaves PGP através do GnuPG e de como usá-la para criptografar o conteúdo textual de um arquivo e descriptografá-lo logo depois.

Para a geração de novas chaves, deve-se usar o comando `gpg` seguido da instrução `-gen-key`. A aplicação então questionará o usuário a respeito dos detalhes da operação e, após a conclusão, uma nova chave privada estará disponível no chaveiro. Essas informações dizem respeito ao tipo de chave desejada, aos dados do usuário, como nome e endereço de e-mail, a validade da chave e a senha que protegerá a chave privada. O será informado ao final da criação do identificador da sua chave, que no exemplo abaixo é o código A4D9 244A 24DE 2555 3177 C88B ED54 9C38 274F 10C4. O comando `gpg --list-keys` pode ser usado para listar as chaves conhecidas, revelando a nova chave recém-criada.



```
daniel@earth-kingdom:~$ gpg --list-keys
/home/daniel/.gnupg/pubring.gpg
-----
pub  2048R/274F10C4 2016-12-01 [expires: 2017-12-01]
uid                               Bob Bobson (Uma nova chave para o Bob) <bob@gmail.com>
sub  2048R/E067F13E 2016-12-01 [expires: 2017-12-01]

pub  2048R/D639898F 2016-12-01 [expires: 2017-12-01]
uid                               Alice Alicesse (Uma chave privada para Alice) <alice@email.com>
sub  2048R/9ED98211 2016-12-01 [expires: 2017-12-01]

daniel@earth-kingdom:~$
```

**Figura B.1:** Listagem de chaves presentes no chaveiro GnuPG

Todas os demais parâmetros possíveis podem ser consultados com o comando `gpg --help` e `man gpg`.

```

x                                     gpg --gen-key
gpg --gen-key 115x45
daniel@earth-kingdom:~$
daniel@earth-kingdom:~$ gpg --gen-key
gpg (GnuPG) 1.4.20; Copyright (C) 2015 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) 2048
Requested keysize is 2048 bits
Please specify how long the key should be valid.
    0 = key does not expire
    <n> = key expires in n days
    <n>w = key expires in n weeks
    <n>m = key expires in n months
    <n>y = key expires in n years
Key is valid for? (0) 1y
Key expires at Qui 30 Nov 2017 22:54:02 BRST
Is this correct? (y/N) y

You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: Bob Bobson
Email address: bob@gmail.com
Comment: Uma nova chave para o Bob
You selected this USER-ID:
    "Bob Bobson (Uma nova chave para o Bob) <bob@gmail.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? 0
You need a Passphrase to protect your secret key.

```

**Figura B.2:** Comando e questionário para geração de nova chave com GnuPG

```

gpg: key 274F10C4 marked as ultimately trusted
public and secret key created and signed.

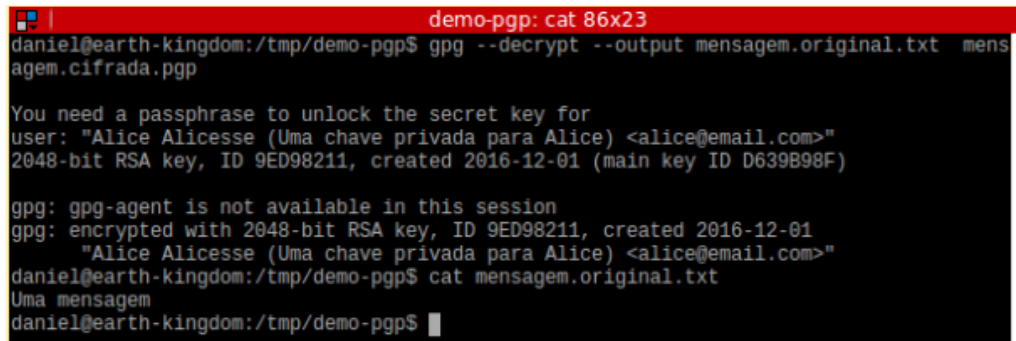
gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2017-12-01
pub   2048R/274F10C4 2016-12-01 [expires: 2017-12-01]
       Key fingerprint = A4D9 244A 24DE 2555 3177 C88B ED54 9C38 274F 10C4
uid     Bob Bobson (Uma nova chave para o Bob) <bob@gmail.com>
sub   2048R/E067F13E 2016-12-01 [expires: 2017-12-01]

daniel@earth-kingdom:~$
daniel@earth-kingdom:~$

```

**Figura B.3:** Resultado da geração da chave

Para descriptografar a mesma mensagem usando a chave do usuário destinatário podemos usar o comando `gpg --decrypt --output mensagem.original.txt mensagem.cifrada.pgp`

A terminal window titled 'demo-gpg: cat 86x23' showing a GPG decryption process. The user runs 'gpg --decrypt --output mensagem.original.txt mensagem.cifrada.pgp'. The terminal prompts for a passphrase to unlock a 2048-bit RSA key for 'Alice Alicesse'. The user enters the passphrase, and the message 'Uma mensagem' is displayed.

```
demo-gpg: cat 86x23
daniel@earth-kingdom:/tmp/demo-gpg$ gpg --decrypt --output mensagem.original.txt mensagem.cifrada.pgp

You need a passphrase to unlock the secret key for
user: "Alice Alicesse (Uma chave privada para Alice) <alice@email.com>"
2048-bit RSA key, ID 9ED98211, created 2016-12-01 (main key ID D639B98F)

gpg: gpg-agent is not available in this session
gpg: encrypted with 2048-bit RSA key, ID 9ED98211, created 2016-12-01
      "Alice Alicesse (Uma chave privada para Alice) <alice@email.com>"
daniel@earth-kingdom:/tmp/demo-gpg$ cat mensagem.original.txt
Uma mensagem
daniel@earth-kingdom:/tmp/demo-gpg$
```

**Figura B.4:** Resultado da geração da chave

## **Detalhes do dispositivo móvel usado para testes**

---

Para a implementação foi utilizado um dispositivo com o sistema operacional Android com as seguintes especificações:

- Modelo : Motorola Moto G4
- Número de versão : MPJ24.139-50 SKU XT 1626
- Memória RAM : 2Gb
- Processador :Qualcomm® Snapdragon? 410 quad-core
- Sistema operacional : Android 6.0.1
- Versão do Kernel : 3.10.84-g9d4ce29

## Detalhes de itens de configuração do aplicativo

---

Abaixo estão expostos detalhes de versão e descrição de linguagens, bibliotecas e aplicações externas utilizadas no desenvolvimento do aplicativo móvel que implementa a proposta deste trabalho.

- Java 8 (OpenJDK 8) 1.8.0\_111 : Linguagem de programação para plataforma Android
- Android SDK Level 22: Kit de desenvolvimento para a API Android 5.1 Lollipop
- Gradle 2.14.1: Gestão das dependências
- OpenPGP API 11 : Cliente de APIs PGP oferecidas por aplicativos de dispositivos móveis
- OpenKeyChain 4.1 : Aplicativo de código aberto que implementa operações de chaveiro PGP
- Apache Commons IO 2.4 : Biblioteca Java para abstração de operações com Streams, dentre outras funções
- Java NanoHTTTPD 2.3.1 : Biblioteca que permite a implementação de servidor HTTP, abstraindo o recebimento e a resposta das requisições dentro do aplicativo.
- Jackson JR 2.8.5 : Implementação menor da biblioteca Jackson Databind, para serialização e deserialização de JSON



## **Ambiente de desenvolvimento utilizado no experimento**

---

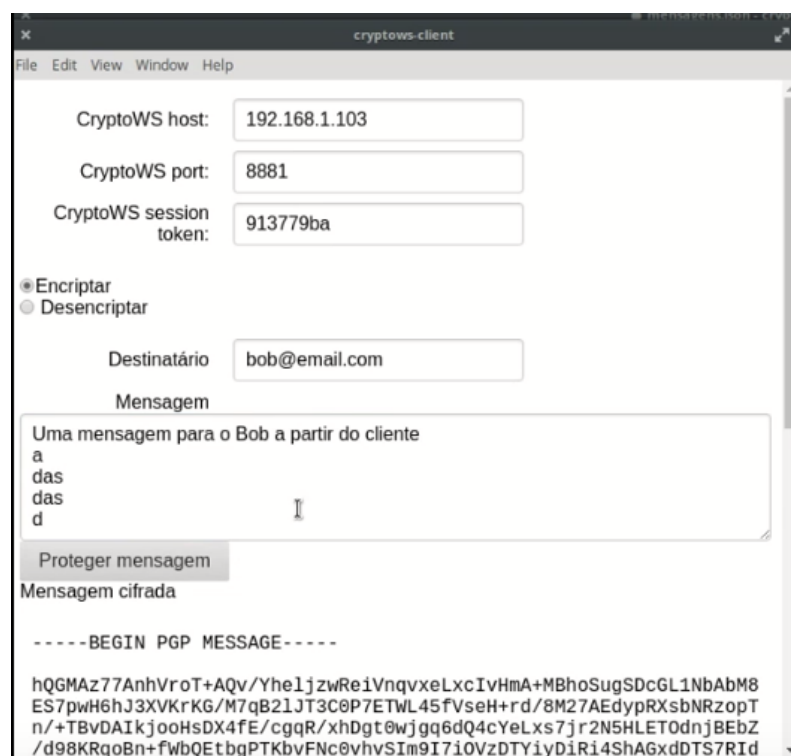
Abaixo são dispostos itens que descrevem o ambiente de desenvolvimento utilizado e as ferramentas que fizeram parte do processo de desenvolvimento deste trabalho e do experimento associado.

- Java 8 (OpenJDK 8) 1.8.0\_111-b14 : Linguagem de desenvolvimento principal
- Git para Linux 2.7.4 : Ferramenta de controle de versão do código-fonte
- Github : Plataforma em nuvem para compartilhamento de projetos Git
- Linux Ubuntu Kernel 4.4.0-47-generic 64 bits : Sistema Operacional utilizado para desenvolvimento
- Android Studio 2.2.2 : Ambiente de desenvolvimento para projetos Android
- Visual Studio Code 1.7.2 - Editor simples de texto para atividades auxiliares

## Exemplo de cliente que consome os serviços criptográficos

Para demonstrar um caso de uso por uma aplicação cliente mínima, foi construído um software com uma única tela. Essa aplicação é capaz de fazer uso dos recursos de criptografia oferecidos a partir do dispositivo móvel.

A tela dessa aplicação permite realizar uma operação de encriptação e desencriptação, exibindo os campos em acordo. A seguir uma imagem da tela implementada. Note que ela contém a configuração de IP, porta e senha de acesso. Neste exemplo uma operação de criptografia e a sua respectiva consulta foram realizadas e o resultado exposto na tela.



**Figura F.1:** Captura de tela de aplicação experimental desenvolvida para fazer uso dos serviços criptográficos contidos no dispositivo móvel.

A aplicação executa um pedido de operação, seja de criptografia, seja de descryptografia e consulta repetidamente o celular. Quando o identificador do pedido (requestId) usado tem uma resposta associada, a mensagem PGP é retornada para a tela, como visto na imagem anterior. Para a demonstração apenas é exibido o texto cifrado em tela.

O desenvolvimento da aplicação cliente foi feito com as bibliotecas listadas a seguir:

- HTML 5 : Linguagem de marcação para construção da tela
- CSS 3 : Estilização da tela
- Javascript ES2015 : Linguagem de programação da tela
- Electron 1.4.1 : Biblioteca para NodeJs que permite construir aplicações desktop usando HTML e Javascript com privilégios de aplicação desktop convencional.
- NodeJs 1.6.9 : Runtime Javascript
- jQuery 3.1.1 : Biblioteca usada para as chamadas HTTP, como uma camada facilitadora para chamadas XMLHttpRequest
- VueJS 2.1.4 : Biblioteca para gestão da camada de visão

## **Exemplo de um Apêndice**

---

## **Exemplo de um Apêndice**

---