Relatório trabalho 1: StegFile, uma aplicação para Esteganografia de imagens

Victor Guedes[1]*, Walter Gomes[2]

Abstract
Este trabalho tem como objetivo descrever o desenvolvimento de um aplicativo Android e de um servidor com foco em prover um serviço de esteganografia em mensagem, ou seja, esconder uma mensagem dentro de de uma imagem para que ela passe despercebida durante uma comunicação. Foi utilizado um método LSB (least significant bit) em cada cor de cada pixel para armazenar a mensagem. O aplicativo tem a função de esconder e retirar a mensagem de dentro de imagem, enquanto o servidor tem a função de guardar imagens com mensagens escondidas para servir como meio de enviar a imagem sem altera-la e comprometer a mensagem escondida. O aplicativo consegue esconder mensagens sem alterar muito a imagem original, permitindo boa invisibilidade da mensagem, tem um bom espaço de armazenamento e também tem 100% de precisão ao extrair mensagens.

Palavras Chave:
Esteganografia — Android

[1] *Estig, IPB, Bragança, Portugal*
[2] *Estig, IPB, Bragança, Portugal*

Contents

Introdução

A esteganografia é a prática de esconder mensagem de forma a elas passarem despercebidas, atualmente existem diversos trabalhos na área de esteganografia digital, onde a esteganografia se mostrou muito importante para segurança de informação e comunicação.

A esteganografia digital ainda é uma atividade difícil de ser aplicada manualmente por um usuário de baixo nível, o objetivo desse trabalho é desenvolver um aplicativo que torne o ato de esconder mensagens dentro de imagens e enviá-las sem comprometer a mensagem um serviço viável e fácil.

O aplicativo utiliza de uma esteganografia de substituição LSB (Least Significant Bit) que utiliza os dois bits menos significantes ao invés de um. O modelo de esteganografia utilizado foi proposto por [1] e aplicado por [2] em uma biblioteca de esteganografia de imagens para desenvolvimento Android, essa biblioteca foi utilizada para a criação do aplicativo. O modelo proposto por [1] também exige o uso de criptografia, que na aplicação foi utilizado o AES (Advanced Encryption Startdard).

O relatório a seguir contém as ferramentas e bibliotecas utilizadas na seção de Ferramentas, a descrição do desenvolvimento na seção de desenvolvimento, um relatório sobre nossos resultados e testes na seção de testes e uma conclusão.

1. Ferramentas

1.1 Image-Steganography-Library-Android

É uma biblioteca recente de esteganografia criada por Ayush Agarwal [2], seu código foi baseado no artigo de 2011 de Ibrahim e Kuan [1], o algoritmo em geral é simples de entender.

Primeiramente ele compacta a mensagem que irá ser inserida na imagem, em seguida, a mensagem compactada será encriptada e então inserida dentro da imagem.[2]

O processo de compactação da mensagem auxilia a estenografia de diversas formas, primeiramente permitindo mais espaço de armazenamento e também auxiliando a dificultar que caso uma mensagem seja descoberta e extraida, ainda seja muito dificil de descobrir seu conteúdo.

O algoritmo de criptografia utilizado o é o AES (Advanced Encryption Standard). A mensagem encriptada recebe dois marcadores, um em seu início e outro em seu final, para auxiliar o processo de extração da mensagem.

Para inserir a mensagem dentro da imagem é utilizado um algoritmo de LSB (Least Significant Bit)[3], que em cada cor de cada pixel, os dois bits menos relevantes de cada cor serão trocados por dois bits da mensagem, dessa forma, a mensagem pode ocupar cerca de 25% da imagem, oferecendo um bom espaço de armazenamento e baixa alteração na imagem original, o que garante uma boa invisibilidade para a mensagem.

O uso de um simples LSB pode vir a ser problemátivo em alguns casos. Apenas trocar os bits menos relevantes não garante uma boa resistência a ataques na imagem feitos para detectar a existência de uma mensagem escondida e devido a forma simples como os dados foram colocados dentro da imagem, pode ser muito fácil remover a mensagem de dentro da imagem e descobrir seu conteúdo, por isso a encriptação e a compactação tem um papel muito importante dentro desse processo esteganográfico, auxiliando a dificultar a extração da mensagem por ataques externos.

A figura 1 apresenta uma comparação feita por Agarwal[2] entre um imagem normal e ela após ser alterada pelo seu programa, demonstrando o cumprimento da condição de invisibilidade do processo esteganográfico.

1.2 Other tools

- **Retrofit:** é uma API desenvolvida pela Square seguindo padrão REST, fornecendo um padrão simples de imple-mentação para transmissão de dados entre aplicação e servidor, que faz uso do JSON.

- **ASP.net Web Api:** é uma estrutura que facilita a criação de serviços HTTP que atingem uma ampla gama de clientes, incluindo navegadores e dispositivos móveis. O ASP.NET Web API é uma plataforma ideal para criar aplicativos RESTful no .NET Framework. Desenvolvida e mantida pela Microsoft.

2. Implementação

O software desenvolvido é divido em duas partes, uma imple-mentação de Api para armazenamento das imagens de todos os usuário e uma aplicação cliente Android que consome essa api.

O primeiro passo foi a criação do banco de dados 2. O banco da Api é somente uma tabela com o Nome do arquivo, um link e o caminho do arquivo no servidor. O que é importante aqui é o campo de Link, que é uma String do tipo UUID, ou seja, um número gerado randomicamente (no cliente) que serve como identificador único para a imagem. O UUID é um número de 128 bits representado por 32 dígitos hexadecimais, exibidos em cinco grupos separados por hifens, na forma de String sendo um total de 36 caracteres (32 caracteres alfanuméricos e 4 hifens). Este é um exemplo de código em java: *link = UUID.randomUUID().toString();*

No lado do cliente já temos os campos de Nome do arquivo, o mesmo link enviado ao servidor, o caminho local do arquivo, e a chave publica que é usada para encriptar e decriptar a mensagem dentro da imagem.

Um exemplo de como os dados estão dispostos no banco do servidor pode ser visto na imagem 3

2.1 Web Api

Com os bancos criados foi implementado uma Api com Asp.net Web Api da Microsoft com a linguagem C#. Basicamente temos métodos com requisição Http de Get e Post para serem usadas pelos clientes. As respectivas rotas são:

- api/Anexos - Get
- api/Anexos/download/link - Get
- api/Anexos/id - Put
- api/upload/link - Post
- api/Anexos/5 - Delete

Os métodos de Put e Delete foram inicialmente implementados no servidor, entretanto não utilizados no lado do cliente. Todos os dados da Api são exportados em formato Json. Todos os código abaixo são apenas exemplo, a codificação completa pode ser encontrada neste GitHub [4].

A rota api/Anexos serve basicamente para buscar todos os dados inseridos no banco, o código em questão pode ser visto abaixo.

```
// GET: api/Anexos
public IQueryable<Anexo> GetAnexo()
{
      return db.Anexo;
}
```

Já a rota api/Anexos/download/link - Get tem como objetivo buscar uma respectiva imagem dada o link, primeiramente ele faz uma consulta no banco para saber se aquele link existe, se sim busca o caminho do arquivo para o usuário poder realizar o download. Se o arquivo existe este é atribuído para um FileStream e adicionado na resposta do http para efetivar o download do Arquivo. O exemplo de código abaixo mostra como fazer isso.

```
[Route("api/Anexos/download/{link}")]
[HttpGet]
public HttpResponseMessage DownloadFile(
    string link)
{
      FileStream(fullPath, FileMode.Open);
   Content = new StreamContent(fileStream);
}
```

Para postar uma nova foto no servidor é usado o código abaixo, que basicamente ajusta a o arquivo enviado para o caminho do servidor e cria um novo objeto para inserir ao banco. alem da Foto, também é enviado o UUID criado pelo cliente.

Figure 1. Comparação: antes e depois de inserir mensagem na imagem [2]

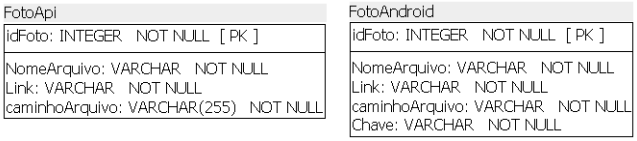

Figure 2. Tabelas do banco

idAnexo	caminhoA...	NomeArquivo	Link
2029	C:\Users\vi...	gato	8c918c79-91e9-48e5-a317-fdc01931fa6b
2030	C:\Users\vi...	cabelo	1b525aaa-3979-423e-aa8a-6e1a351937be
2031	C:\Users\vi...	pc	1511b674-6fef-4204-9596-130c0cb9aa6f
2032	C:\Users\vi...	bigode	145f0c8b-49c2-4c71-98ca-5e1a5b0f8cc1
2033	C:\Users\vi...	Gato no pc	da1d121c-160f-4123-815a-1c1049c56f53

Figure 3. Banco servidor com dados

```
[Route("api/upload/{link}")]
[HttpPost]
 public async Task<HttpResponseMessage>
     PostAnexo(string link)
{
       httpPostedFile = httpContext.Request.
           Files[i];

   httpPostedFile.SaveAs(fileSavePath);
   Anexo anexo = new Anexo();
   anexo.NomeArquivo = httpPostedFile.
       FileName;
   anexo.caminhoArquivo = fileSavePath;
   anexo.Link = link;
   db.Anexo.Add(anexo);
   await db.SaveChangesAsync();
}
```

2.2 Cliente Android

No lado do cliente temos uma aplicação android para fazer a as requisições ao servidor. Para isso foram utilizadas as bibliotecas Retrofit, Sugar ORM e Image-Steganography que é a mais importante para efetivar o processo de esteganografia.

Iniciado a aplicação é apresentado todas as mensagem armazenadas no servidor. Aqui não é interessante saber qual mensagem é qual, de que usuário pertence, apenas é mostrado uma lista com os respectivos nomes. Para facilitar a busca de uma imagem é utilizado o UUID criado, a imagem 4 mostra o layout da aplicação.

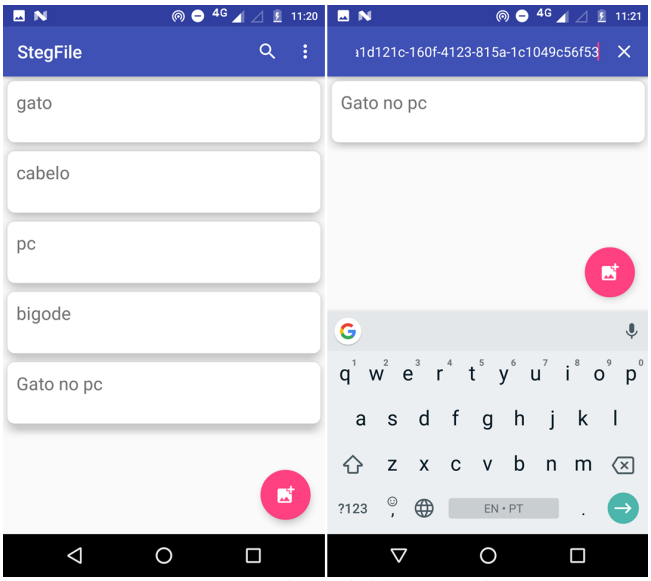

Figure 4. Pagina Inicial e Pesquisa

Na figura 5 é apresentado as telas de Encode e Decode de imagens selecionas. Para realizar o Encode que é esconder um texto em uma imagem é necessário passar o nome da imagem, a chave de desejo (encriptada com AES, como já dito anteriormente), e o texto a ser encriptado. No código também é criado o UUID da imagem. Ao apertar o botão de enviar, primeiramente é realizado o processo de esconder a imagem, em seguida é enviado ao servidor e postado no banco local do usuário.

Para realizar o encode é utilizado a chamado do método abaixo que é da biblioteca Image-Steganography. Basicamente é instanciado um objeto do tipo ImageSteganography e passado no seu construtor os atributos texto a ser escondido, a chave, e a imagem que precisa estar em formato de Bitmap. Bitmap é uma matriz de bits que especifica a cor de cada pixel numa matriz retangular de pixeis. Posterior a isso tambem é criado um objeto do tipo TextEncoding que é um

Thread (AsyncTask) para executar o encode em background, passando a ImageSteganography criada.

Assim é usada a biblioteca. No seu interior o processo começa primeiramente com a mensagem secreta que é extraída será compactada, para que o conteúdo da string compactada seja difícil de detectar e ler, em seguida esta cadeia é encriptada com com o algoritmo de AES e finalmente codifica a mensagem criptografaca na imagem utilizando o LSB.

```
new ImageSteganography(message, secret_key,
    image);

textDecoding = new TextEncoding(this, this);

textEncoding.execute(imageSteganography);
```

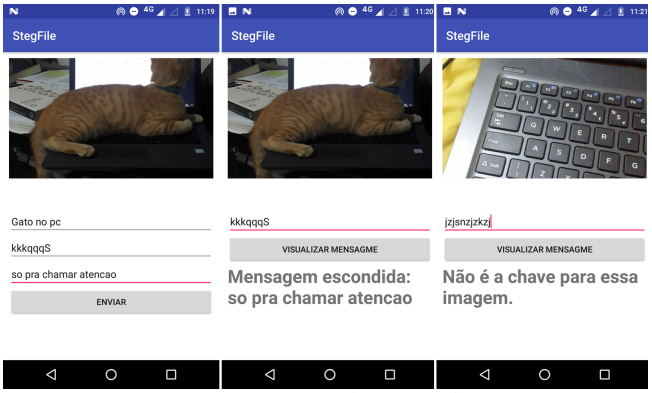

Figure 5. Encode / Decode

Para extrair o texto escondido da imagem é interessante que o usuário saiba o link e a chave respectiva. Assim com o link em mãos é possível pesquisar a imagem e usar a chave. Tais informações são passadas por fora do aplicativo, deixando o usuário escolher quem pode ou não ler sua mensagem.

Com a chave em mãos o usuário vai na tela de decode é visualiza a mensagem, Caso seja a chave correta. A imagem 5 mostra um exemplo em que a chave é correta e errada.

O processo na biblioteca funciona da seguinte forma: primeiramente é decodificado a mensagem da imagem criptografada usando a decodificação LSB, em seguida é decifrado a mensagem comprimida da mensagem decodificada usando a chave, e por fim e descompactada a mensagem para obter a mensagem original. Em questão de código basta utilizar o mesmo processo do encode, mas ao inves de chamar o Texecoding chama-se o TextDecoding como pode ser visto no codigo abaixo.

```
new ImageSteganography(message, secret_key,
    image);

textDecoding = new TextDecoding(this,this);

textEncoding.execute(imageSteganography);
```

Como dito anteriormente, todas as mensagem que o usuário envia também são armazenas em um banco local, assim o usuário pode visualizar todas de desejo e o mais importante as

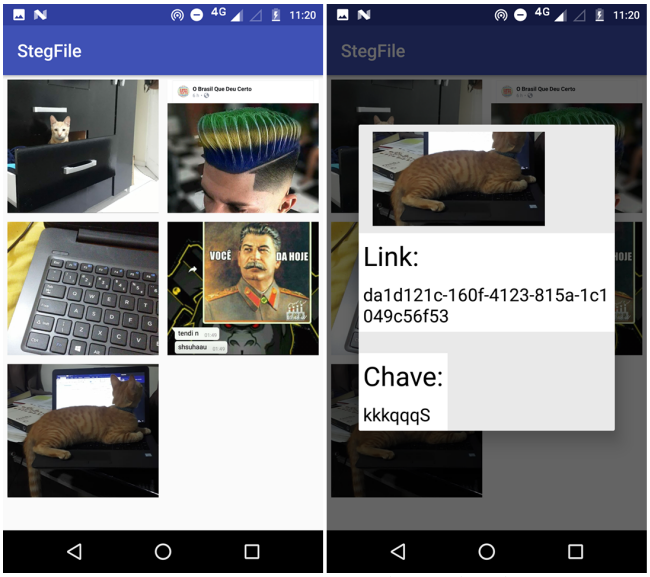

Figure 6. Imagens em banco local

informações de link e chave para serem compartilhadas com outras pessoas. A imagem 6 mostra as telas deste caso.

3. Análise

O interessante agora é fazer uma análise de trafego para identificar como as requisições http estão se comportando, bem como saber se é possível visualizar o texto dentro da imagem Para isso foi utilizado o programa WireShark.

Neste exemplo foi utilizado uma imagem no formato JPG com 56,60 Kb que depois com o texto fica com aproximadamente 57,29 kb como pode ser visto na imagem 7. O texto escondido neste teste é ”Gato na gaveta”, o que é considerado pouco para um download de imagem, entretanto com textos maiores a imagem também fica maior o que pode causar suspeitas por parte do usuário.

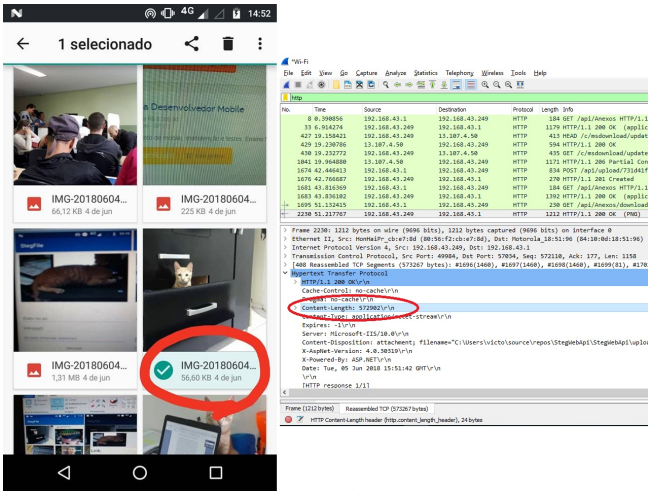

Figure 7. Tamanho Imagem

Já em relação a se é possível a visualização do texto es-

condido é necessário olhar cada Frame da imagem e decifrar o texto, entretanto este esta encriptado com o algoritmo do AES o que torna esta tarefa árdua apenas com análise comum. Obriga o usuário descobrir a chave publica para aquela mensagem, ou descobri-la por métodos de força bruta. A imagem a seguir 8 mostra este caso. Entretanto como dito anteriormente, um arquivo png com tamanho exagerado poderia causar suspeitas.

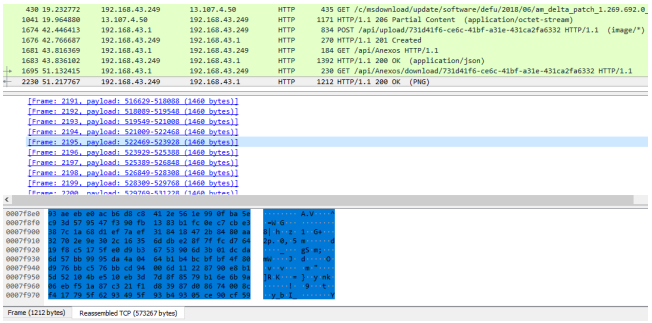

Figure 8. Frames Imagem

A invisibilidade da mensagem foi garantida com sucesso, embora os arquivos gerados possam ser suspeitos devido a seus tamanhos exagerados. As imagens enviadas pelo servidor conseguem manter a integridade da mensagem intacta, porém ao colocar uma mensagem grande demais aplicativo não consegue removê-la com sucesso.

Houveram dificuldades em determinar qual o tamanho máximo que uma mensagem pode ter relativamente ao tamanho da imagem, devido ao tamanho da imagem ser alterado.

4. Conclusão

Esteganografia é um conceito ainda pouco conhecido por usuários comum, entender tais conceitos é fundamental para que não seja compartilhado virus, malwares, qualquer tipo de arquivo malicioso porque assim como em nosso exemplo é preciso a utilização de chaves para descobrir o texto, este pode ser executa automaticamente o que poderia causar problemas para usuários.

Portanto fica a sugestão já conhecida de tomar cuidado com os downloads realizados, onde ”clickar” na web e para os mais fanáticos, desconfiar de qualquer download disposto na Internet.

Em trocas de informações sigilosas a esteganografia apresenta características fora do esperado por um observador externo, garantindo ao usuário do aplicativo novos meios de enviar dados aumentando sua capacidade de sigilo. Tais funções de comunicação podem ser muito úteis para comunicação de dados empresariais sigilosos entre outras coisas do genero.

References

[1] Teoh Suk Kuan Rosziati Ibrahim. Steganography algorithm to hide secret message inside an image. *Computer Technology and Application 2*, pages 102–108, 2011.

[2] Ayush Agarwal. Image-steganography-library-android. `https://github.com/aagarwal1012/Image-Steganography-Library-Android`, 2018.

[3] Jeffrey A. Bloom Jessica Fridrich Ton Kalker Ingemar J. Cox, Matthew L. Miller. *Digital Watermarking and Steganography: Second Edition*. Elsevier, 2008.

[4] G. Victor and G. Walter. Aplicação stegfile. `https://github.com/VictorGuedes/StegFile`, 2018.