Aplicativo de Troca de Mensagens Utilizando Criptografia Baseada em Imagens

Marcus Antonio Grécia Brandt[1], **Mikhail Yasha Ramalho** [1]

[1] Centro de Ensino Superior FUCAPI (CESF)
Av. Gov. Danilo de Mattos Areosa, 381 – Distrito Industrial – Manaus – AM – Brasil

{marcusbrandt, mikhailramalho}@gmail.com

Abstract. *This meta-paper describes the style to be used in articles and short papers for SBC conferences. For papers in English, you should add just an abstract while for the papers in Portuguese, we also ask for an abstract in Portuguese ("resumo"). In both cases, abstracts should not have more than 10 lines and must be in the first page of the paper.*

Resumo. *Aplicações de troca de mensagens possuem naturalmente um forte apelo popular e figuram como aplicações de destaque nas lojas de aplicativos. Porém levantam questões relativas a segurança e confiabilidade das informações que por eles trafegam. Este artigo descreve a implementação de um aplicativo de comunicação por meio do bluetooth que utiliza um algoritmo de criptografia baseado em imagens como chave de acesso.*

1. Introdução

2. Fundamentação Teórica

2.1. ASCII

O ASCII *American Standard Code for Information Interchange* (Código Padrão Americano Para o Intercâmbio de Infromação) é um esquema de codificação inicialmente baseado no alfabeto americano, posteriormente se tornou um esforço para padronização da representação de caracteres pelos fabricantes de computadores.

Cada caracter (pontuação, valores alfanuméricos e valores de controle) é representado por um valor numérico divididos em uma tabela. Originalmente como eram representados apenas caracters americanos, acentuações e letras com essa característica não podiam ser representados, posteriormente a tabela passou por uma revisão e foram criadas 12 novas partes pra suprir essa necessidade.

2.2. Imagem

Quando digitalizadas as imagens são representadas por bits e bytes que são convertidos para números pra representar cada cor de um pixel específico, podendo ser representada por uma matriz de pixels(Figura 1) .

2.3. Criptografia

Criptografia é o estudo de técnicas pela qual as informações podem ser trocadas de forma segura através de um canal de comunicação aberto, tornando a mensagem em algo conhecido apenas pelos seus agentes.

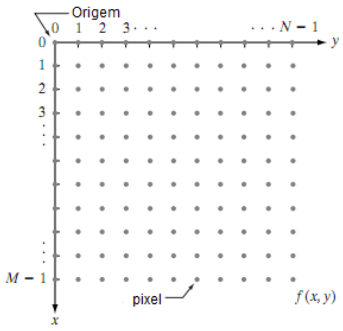

Figura 1. Representação de uma imagem digital

2.3.1. Chave Simétrica

No processo de criptografia simétrica existe uma chave que é compartilhada entre todos os que pretendem desenciptar a mensagem, essa é a principal desvantagem desse método pois assumimos que todos que se utilizam da chave poderam obter as informações relacionadas a criptografia da mensagem, outro problema é o gerenciamento das chaves, olhando por um cenário de uma empresa todos que utilizar chaves que permitissem acesso a informações específicas tornando o gerenciamento das chaves problemático (Figura 2).

O processo de criptografia simétrica pode se dar por duas formas substituição e cifra de transposição (ou blocos), ambos podem ser bit-a-bit ou bloco de dados.

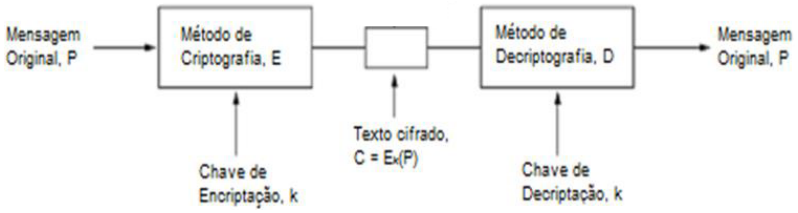

Figura 2. Sistema Criptográfico de Chave Simétrica
[Gadelha 2010]

2.3.2. Chave Assimétrica

No processo de criptografia assimétrica existem duas chaves diferentes mas matematicamente relacionadas para criptografar e descriptografar, são conhecidas como chave pública e chave privada, a chave pública pode ser distribuída para todas as outras partes para o processo de criptografia e a informação só poderá ser desencriptada com a chave privada (Figura 2) e essa é a principal diferença entre os dois métodos.

2.4. Bluetooth

Bluetooth é uma tecnologia sem fio para troca de informação em pequenas distâncias baseada em ondas curtas e é padronizada pelo IEEE 802.15.1, foi originalmente concebida para ser alternativa ao RS-232.

Ele opera em uma faixa de freqência de 2400–2483.5 MHz, utiliza a tecnoligia de rádio chamada de salto de frêquencia em espalhamento espectral (*frequency-hopping*

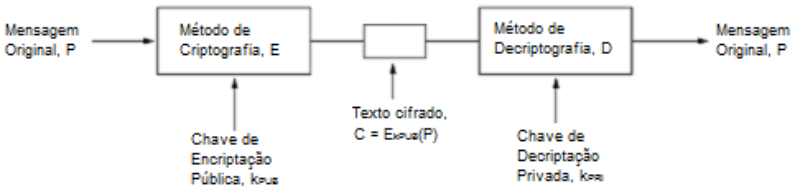

Figura 3. Sistema Criptográfico de Chave Assimétrica
[Gadelha 2010]

spread spectrum). Baseado em um protocolo de troca de pacotes com estrutura de *master-slave*.

Um servidor Bluetooth pode se comunicar com até 7 devices em uma rede piconet (*ad-hoc*) baseada na tecnologia Bluetooth.

3. Sistema

3.1. Android

O protótipo apresentado neste trabalho foi desenvolvido em Android, por ser uma plataforma popular para desenvolvimento de aplicativos móveis. O Android possui interfaces de programação de aplicativo (APIs), que vêm com o seu Software Development Kit (SDK) e possui recursos completos de interface[Android 2014].

Para este protótipo utilizamos a versão 4.4.2 do Android conhecida como Jelly Beans, por ser a versão mais recente disponível, quando do desenvolvimento deste trabalho.

3.2. OpenCV

O OpenCV é um conjunto de bibliotecas desenvolvidas com o objetivo de provêr visão computacional em tempo real. Desenvolvida originalmente em C mudou sua plataforma para C++ com o objetivo de se tornar mais facil o desenvolvimento de novas funcionalidades, melhor implementação e aplicação de padrõee, com suporte em Python e Java (Android), com suporte em Windows, Linux, iOS, Mac OS e Android.

3.3. Java & JNI

Java é uma linguagem de programação adotada como padrão no desenvolvimento de aplicativos em Android, é baseada em classes, orientada a objeto e exitem várias APIS (biblioteca) [qual o melhor?] prontas, oferecendo uma sintaxe muito parecida com C e C++ é atualmente a linguagem mais popular do mundo utilizada principalmente em sistemas web cliente-servidor, foi originalmente desenvolvida por James Gosling na Sun Microsystems porteriormente adquirida pela Oracle.

Para comunicações mais baixo nível o java utiliza um artificio de comunicação via callbacks através da sua api de comunicação nativa a JNI (*Java Native Interface*)

3.4. Processo de Criptografia

3.5. Sistema de Criptografia Utilizando Imagens

Nossa proposta é composta primeiramente por um algoritmo de chave simétrica, pois haverá uma troca de cada caracter por uma correspondente da imagem escolhida. O processo de criptografia que utilizamos nesse artigo proposto por [Gadelha 2010] consistem

em utilizar uma imagem como chave para encriptação (Figura 4) e para desencriptação (Figura 5). O algoritmo se baseia na seguinte ideia:

*Uma imagem é representada por pixels que tem valores de 0 até 255 assim como a tabela ASCII que tem valores de 0 à 255 incluindo digítos de controle.

*Um problema pode ocorrer na substituição de alguns caracters quando não houver correspondente na imagem. Podendo tornar o custo do algoritmo maior.

Encriptação com procura de pixels próximos

1. Ler mensagem original *msg* de um arquivo
2. Ler imagem-chave *img*
3. Calcule o número de bits *n* máximos necessários para representar as coordenadas de *img*
4. Calcule *flag* de parada *escape* como a maior dimensão da imagem + 1
5. Inicia mensagem cifrada *msgCript* nula com inteiro sem sinal de precisão *n*
6. **Para** cada caractere *c* na *msg* **faça**
    a. Retorne uma lista *list* de par de coordenadas *(x,y)* onde *img(x,y) = c*
    b. **Se** existir o par coordenadas
        i. Escolha aleatoriamente uma posição *k* na *list*
        ii. *encryptedChar* = par de coordenadas de *list(k)*
    c. **Senão**
        i. Retorne o vetor contendo *temp* = [1 (43/45) *dist escape* 0]
        ii. Retorne uma lista *list* de par de coordenadas *(x,y)* onde *img(x,y) = c (+/-) dist*
        iii. Escolha aleatoriamente uma posição *k* na *list*
        iv. *encryptedChar* = *temp* concatenado com *list(k)*
    d. FimSe
    e. **Se** o tamanho de *encryptedChar* == 2 e *msgCript* está vazia
        i. *msgCript* = 0 concatenado com ele mesmo.
    f. FimSe
    g. **Se** o tamanho de *encryptedChar* > 2 e *msgCript* não está vazia
        i. *msgCript* = *escape* concatenado com *encryptedChar*
    h. FimSe
    i. Adicione ao fim de *msgCript* a variável *encryptedChar*
7. **Fim para**
8. Salve *msgCript* em um arquivo utilizando *n* bits de precisão

Figura 4. Algoritmo de encriptação [Gadelha 2010]

*Será utilizada a mesma imagem para decriptografia do mesmo.

*Havendo caracteres sendo representados pelos mais próximos calcularemos o valor real para desencriptar.

4. Resultados

5. Conclusão

Referências

Android, D. (2014). Develop. android developers.

Gadelha, M. Y. R. (2010). Estudo de métodos de criptografia de dados utilizando imagens.

M. Y. R. Gadelha, C. F. F. Costa Filho, M. G. F. C. (2012). Proposal of a cryptography method using gray scale digital images. *The 7th International Conference for Internet Technology and Secured Transactions*.

Decriptação com procura de pixels próximos

1. Ler imagem-chave *img*
2. Calcule o número de bits *n* máximos necessários para representar as coordenadas de *img*
3. Ler mensagem cifrada *msgCript* utilizando *n*-bits para cada coordenada
4. Calcule *flag* de parada *escape* como a maior dimensão da imagem + 1
5. Inicia mensagem decriptada *msgDecript* nula
6. Inicia *hasOp* como *false*
7. Inicia i = 1
8. **Enquanto** i < tamanho da mensagem encriptada **faça**
    a. **Se** msgCript(i) == 0
        i. i = i + 1
        ii. **Enquanto** *msgCript*(i) ≠ *escape* **faça**
            1. *x* = *msgCript*(i)
            2. *y* = *msgCript*(i+1)
            3. *decryptedChar* = valor do pixel *p* na coordenada *img(x,y)*
            4. **Se** *hasOp* == *true*
                a. *decryptedChar* = *decryptedChar* + *corr*
                b. *hasOp* = *false*
            5. **FimSe**
            6. *msgDecript* = *decryptedChar* concatenado com *msgDecript*
            7. i = i + 2
            8. **Se** i > tamanho de *msgCript*
                a. *Break*
            9. **FimSe**
    b. **SenãoSe** msgCript(i) == 1
        i. i = i + 1
        ii. *hasOp* = *true*
        iii. **Se** *msgCript*(i) == 43
            1. *corr* = msgCript(i+1)*(-1)
        iv. **SenãoSe** *msgCript*(i) == 45
            1. *corr* = msgCript(i+1)
        v. **FimSe**
        vi. i = i + 2
    c. **FimSe**
    d. i = i + 1
9. **FimEnquanto**
10. Salve *msgDecript* em um arquivo

Figura 5. Algoritmo de desencriptação [Gadelha 2010]