Aluno: Anderson Simioni Assunção
Professor: Prof. Lucas Debatin, MSc.
Curso: Ciências da computação
Disciplina: Programação para Dispositivos Móveis

Seminário M3

Sumário

1.Introdução

O projeto consiste no desenvolvimento de uma aplicação que auxiliará em transações exclusivamente de Bitcoin de forma segura, fazendo então o papel de uma carteira de criptomoedas, porém ela possibilitará realiza operações em outro moedas além do Bitcoin, é importante ressaltar que as transações ocorrem na rede principal denominada "MAIN_NET", por tanto não possível realizar testes em redes secundárias ou na rede “TEST_NET”.

2. Resolução do problema

A segurança das carteiras de criptomoedas é um assunto polêmico nos dias atuais, é comum vermos pessoas se questionando se o uso de apps são tão seguros quanto o próprio bitcoin core, software oficial da rede, a maior preocupação da maioria dos usuários é a possibilidade do vazamento de suas chaves privadas.

Usuários mais antigos optam por comprar um computador e dedicá-lo para uso exclusivo de carteira, geralmente usando o software oficial da própria rede como meio mais seguro.

3.Proposta de solução

Portanto, visando a segurança das informações privadas das carteiras dos usuários, decidimos implementar o mesmo sistema de segurança do software oficial da rede Bitcoin, a criptografia utilizada para proteger as chaves de carteira é nomeada de AES-256-CBC, é uma criptografia síncrona com chave de 256 bits, alguns desenvolvedores consideram ela pós-quântica devido ao tamanho da chave. A criptografia AES do tipo CBC permite a utilização de duas informações como base, a chave propriamente dita e um “Initial vector” que é um dado com 128 bits, dessa forma a soma equivale a uma criptografia de 384 bits, a senha do usuário é utilizada para calcular essas duas informações com base nas hashes SHA256 para a chave principal e a hash MD5 para o “Initial Vector”, ambas hashes utilizam sistema de salto que o mesmo é calculado com base na senha. No nosso caso, não temos uma criptografia igual ao Bitcoin core pois temos uma superior.

4.Justificativa

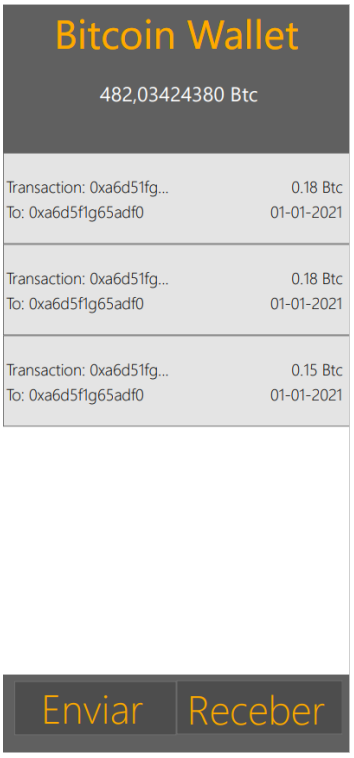
Como visto antes, trata-se de uma solução de segurança, para que o usuário tenha maior mobilidade e não fique dependente de um software disponível unicamente para computadores notebook ou desktop para se sentir seguro, a nossa intenção é que o usuário se sinta seguro no seu próprio celular, basta o mesmo utilizar uma senha forte para a criptografia da carteira instalada e armazenar com cuidado suas palavras secretas para gerar a carteira.

5.Objetivo geral

Portanto, nosso objetivo é ajudar na segurança e mobilidade do usuário ao realizar transações na rede Bitcoin.

6.Protótipo inicial

Nosso protótipo inicial não se comparada ao resultado final do aplicativo, foram realizadas diversas melhorias no projeto, abaixo seguem as imagens do protótipo:

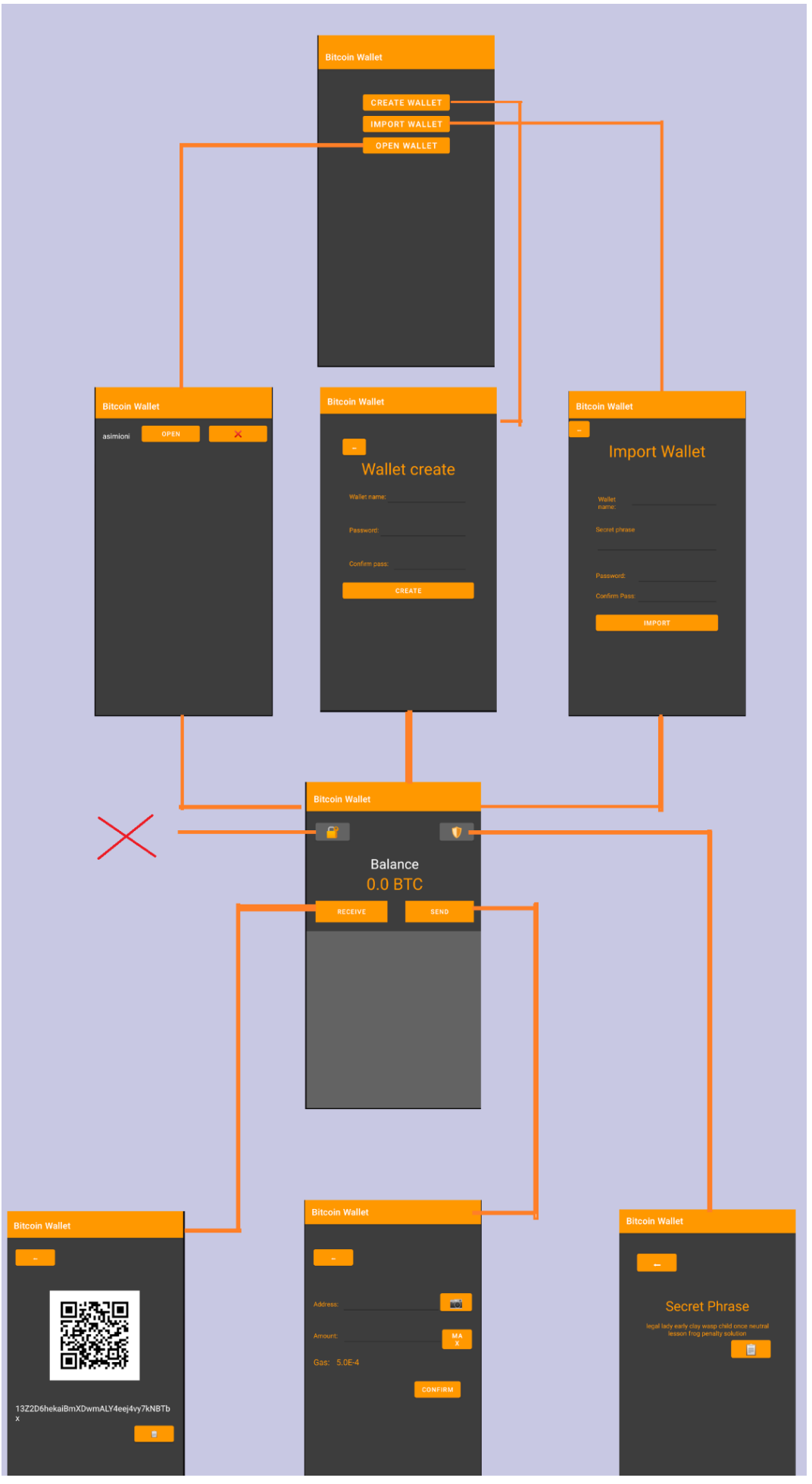


7.Navegação

A navegação pelas telas consistem em 100% por fragmentos, começando pelo fragmento com as opções de criar, importar ou abrir uma carteira, clicando em criar uma carteira o usuário será direcionado para a fragment de criação, ele deverá ditar qual será o nome da carteira nova e a sua senha de segurança, após confirmar, aparecerá a sua frase secreta que deve ser anotada e ele será redirecionado para a página inicial da carteira. Caso o usuário clique em importar uma carteira, ele irá para o fragmento de importação que pedirá sua frase secreta, um novo nome para a carteira e uma nova senha de segurança, após confirmar ele será redirecionado para a página inicial da carteira. Por fim, caso ele clique em abrir uma carteira, aparecerá uma lista com todas as suas carteiras disponíveis, assim que ele clicar em alguma pedirá sua senha de segurança e o redireciona para a página inicial da carteira.

Na página inicial da carteira, o usuário tem as opções de receber, enviar, trancar ou mostrar a frase se segurança, clicando em receber ele será direcionado para um fragmento que conterá um Qr Code que possibilita outra carteira escanear seu endereço e também o seu endereço escrito e com botão para copiar, ao clicar em enviar o usuário irá para um fragmento cujo ele deverá delegar para qual endereço deseja enviar e a sua quantia. Por fim, temos a opção de trancar e mostrar a frase secreta, clicando trancar, a carteira fechará sendo necessário colocar novamente a senha para abrir, e clicando em mostrar a frase secreta o usuário poderá digitar sua senha e ver sua frase de recuperação da carteira.

Ilustração navegação:



8.Imagens do código fonte

Geração da sequência “Initial Vector”:

```java
/**
 * Derive initial vector to encryption
 * @return
 */
@RequiresApi(api = Build.VERSION_CODES.KITKAT)
private byte[] getInitialVector() throws NoSuchAlgorithmException {
    int len = PASS_KEY.length;
    int sum = 0;
    int mult = 0;

    for (int i = 0; i < len; i++){
        sum += PASS_KEY[i];
        mult *= PASS_KEY[i];
    }

    int year = 2021;
    if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.O) {
        year = LocalDateTime.now().getYear();
    }
    int salt = (int) len * sum * mult * year;
    byte[] concatKey = (BytesOperator.bytesToString(ENCRYPTION_BASE_KEY) + BytesOperator.bytesToString(PASS_KEY)).getBytes();

    MD5 md5Alg = new MD5();

    byte[] newKey = md5Alg.compute(concatKey, salt);

    return newKey;
}
```

Geração da “Private key”:

```java
/**
 * Generate 256bis random information with pass key
 * @return
 */
@RequiresApi(api = Build.VERSION_CODES.KITKAT)
private byte[] get256BitsEncryptionKey() throws NoSuchAlgorithmException {
    int len = PASS_KEY.length;
    int sum = 0;
    int mult = 0;

    for (int i = 0; i < len; i++){
        sum += PASS_KEY[i];
        mult *= PASS_KEY[i];
    }

    int year = 2021;
    if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.O) {
        year = LocalDateTime.now().getYear();
    }
    int salt = (int) len * sum * mult * year;
    byte[] concatKey = (BytesOperator.bytesToString(ENCRYPTION_BASE_KEY) + BytesOperator.bytesToString(PASS_KEY)).getBytes();

    SHA256 sha256Alg = new SHA256();

    byte[] newKey = sha256Alg.compute(concatKey, salt);

    return newKey;
}
```

Função de criptografia e bases para chaves:

```
private static final byte[] ENCRYPTION_BASE_KEY = "G`$pijtg230hg2bg9ugfbsogrg".getBytes();
private static final byte[] ENCRYPTION_BASE_IV = "G($(*%(@)#*&$WR40e5yu04ehng9noger".getBytes();
private final byte[] PASS_KEY;

/**
 * Encode data text into cipher hex data
 * @param src
 * @return
 */
@RequiresApi(api = Build.VERSION_CODES.KITKAT)
public String encrypt(String src) {
    try {
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        IvParameterSpec initialVector = new IvParameterSpec(getInitialVector());
        SecretKeySpec key = new SecretKeySpec(get256BitsEncryptionKey(), algorithm: "AES");

        cipher.init(Cipher.ENCRYPT_MODE, key, initialVector);
        return BytesOperator.bytesToHex(cipher.doFinal(src.getBytes()));
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
```

Resumo do código fonte: