BTC Wallet

Dennis | Jan | Alkan | Samer | Mariusz

1

1

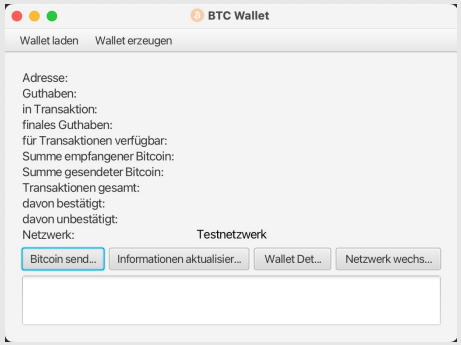
Inhaltsverzeichnis

1. Live-Demonstration der Wallet
2. Beschreibung einzelner Code-Abschnitte
3. Potenzielle Verbesserungsvorschläge für Wallet 2.0
4. Sicherheit die durch Blockchain gewährleistet wird
5. Sicherheit die durch die Wallet gewährleistet wird
6. Schutzbedarfsfeststellung
7. Risikoanalyse

2

2

Live-Demonstration der Wallet



3

3

Netzwerkwechsel (1)

```
@FXML
void handleSwitchNetwork() throws NoSuchAlgorithmException, NoSuchProviderException, IOException {
    // int option = JOptionPane.showOptionDialog(null, "Achtung! Wenn Sie auf das
    // Bitcoin-Hauptnetzwerk wechseln, handeln Sie mit echtem Geld. Möchten Sie den
    // Wechsel auf das Hauptnetzwerk bestätigen?", "Netzwerk wechseln",
    // JOptionPane.YES_NO_OPTION, JOptionPane.WARNING_MESSAGE, null, null, address);

    if (walletLoaded) {
        if (network) {
            int option = JOptionPane.showOptionDialog(null,
                    "Bitte bestätigen Sie den Wechsel auf das Testnetzwerk.", "Netzwerk wechseln",
                    JOptionPane.YES_NO_OPTION, JOptionPane.WARNING_MESSAGE, null, null, null);
            if (option == 0) {
                network = false;
                networkText.setText("Testnetzwerk");
                changeAddressLabel();
                extendLog("Auf Testnetzwerk gewechselt.");
            } else
                return;

        } else {
            int option = JOptionPane.showOptionDialog(null,
                    "Achtung! Wenn Sie auf das Bitcoin-Hauptnetzwerk wechseln, handeln Sie\nmit echtem Geld. Möchten Sie den Wechsel auf das Hauptnetzwe
                    "Netzwerk wechseln", JOptionPane.YES_NO_OPTION, JOptionPane.WARNING_MESSAGE, null, null, null);
            if (option == 0) {
                network = true;
                networkText.setText("Hauptnetzwerk");
                extendLog("Auf Hauptnetzwerk gewechselt.");
                changeAddressLabel();
            } else
                return;
        }
    } else {
```

4

4

Netzwerkwechsel (2)

```
} else {
    if (network) {
        int option = JOptionPane.showOptionDialog(null,
                "Möchten Sie wirklich auf das Bitcoin-Testnetzwerk wechseln?", "Netzwerk wechseln",
                JOptionPane.YES_NO_OPTION, JOptionPane.WARNING_MESSAGE, null, null, null);
        if (option == 0) {
            network = false;
            networkText.setText("Testnetzwerk");
            extendLog("Auf Testnetzwerk gewechselt.");
        } else
            return;
    } else {
        int option = JOptionPane.showOptionDialog(null,
                "Achtung! Wenn Sie auf das Bitcoin-Hauptnetzwerk wechseln, handeln Sie\nmit echtem Geld. Möchten Sie den Wechsel auf das Hauptnetzwerk"
                        + " bestätigen?",
                "Netzwerk wechseln", JOptionPane.YES_NO_OPTION, JOptionPane.WARNING_MESSAGE, null, null, null);
        if (option == 0) {
            network = true;
            networkText.setText("Hauptnetzwerk");
            extendLog("Auf Hauptnetzwerk gewechselt.");
        } else
            return;
    }
}
```

5

5

Automatische Auswahl von UTXO's

```
int check_utxo_sats = 0;

for (int i = 0; i < temp.size(); i++) {
    check_utxo_sats += temp.get(i).getValue();
}

if (check_utxo_sats >= debit + 1000) {
    int vint = 0;
    int utxo_sats = 0;
    while (utxo_sats < (debit + 1000)) {
        utxo_sats += temp.get(vint).getValue();
        vint++;
    }

    LinkedList<UTXO> utxo = new LinkedList<UTXO>();

    for (int i = 0; i < vint; i++) {
        utxo.add(temp.get(i));
    }

    if ((debit + 1000) < utxo_sats) {
        outputs.add(new Output((utxo_sats - 1000 - debit), address));
    }
```

6

7

Signieren von mehreren UTXO's

```
public String sign(BigInteger privateKey) throws NoSuchAlgorithmException, SignatureDecodeException {
    LinkedList<UTXO> signed_inputs = new LinkedList<UTXO>();

    // Jeden Input signieren
    for (int i = 0; i < utxos.size(); i++) {
        String rawTxToSign = "";
        String outputs = rawTX.getOutput();
        String inputs = "";
        int input_n = utxos.size();
        for (int j = 0; j < utxos.size(); j++) {
            inputs += utxos.get(j).getTxID_LE();
            inputs += utxos.get(j).getVout_LE();
            inputs += utxos.get(j).getScriptSize(i == j);
            inputs += utxos.get(j).getScriptPubKey(i == j);
            inputs += "ffffffff";
        }
        rawTxToSign += RawTX.getRawTX2String(outputs, inputs, input_n) + "01000000"; // SigHashAll
        byte[] messageHashedTwice = SHA256.hashTwice(Hex.decode(rawTxToSign));
        ECKey key = ECKey.fromPrivate(privateKey, false);
        ECDSASignature sig = key.sign(Sha256Hash.wrap(messageHashedTwice));
        sig.toCanonicalised();
        byte[] derSigned = sig.encodeToDER();
        String pubKeyHex = key.getPublicKeyAsHex();
        byte[] derSigned01 = ByteBuffer.allocate(derSigned.length + 1).put(derSigned).put(Hex.decode("01")).array();
        int sigLength = derSigned01.length;
        int pubKeyLength = (pubKeyHex.length() / 2);
        String scriptSig = Integer.toHexString(sigLength) + Hex.toHexString(derSigned01)
                + Integer.toHexString(pubKeyLength) + pubKeyHex;
        UTXO temp = new UTXO(utxos.get(i).getTxID(), scriptSig, utxos.get(i).getVout(), utxos.get(i).getValue());
        signed_inputs.add(temp);
    }
    RawTX signed = new RawTX(signed_inputs, rawTX.getOutput());

    return signed.getRawTX(true);
}
```

7

9

Abfrage der UTXO's von Blockchain

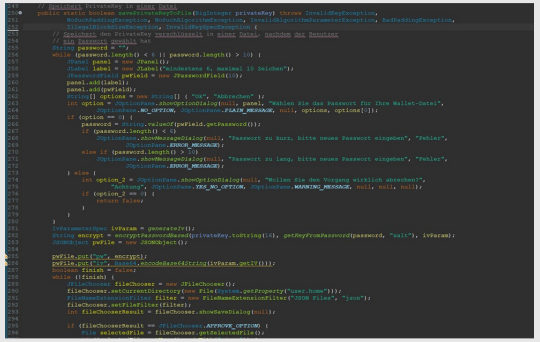
```
public static LinkedList<UTXO> getUTXOs(String address, boolean network) throws IOException {
    LinkedList<UTXO> utxos = new LinkedList<UTXO>();
    String utxosFromAdress = "";
    if (network)
        utxosFromAdress = mainnetURL + address + apiFilter;
    else
        utxosFromAdress = testnetURL + address + apiFilter;
    try (CloseableHttpClient httpClient = HttpClients.createDefault()) {
        HttpGet request = new HttpGet(utxosFromAdress);

        try (CloseableHttpResponse response = httpClient.execute(request)) {
            HttpEntity entity = response.getEntity();
            String result = EntityUtils.toString(entity);

            JSONArray unspentOutputs = new JSONArray();
            try {
                unspentOutputs = new JSONObject(result).getJSONArray("txrefs");
            } catch (JSONException e) {
                return null;
            }
            for (int i = 0; i < unspentOutputs.length(); i++) {
                JSONObject unspentOutput = unspentOutputs.getJSONObject(i);

                String txid = unspentOutput.getString("tx_hash");
                String scriptPubKey = unspentOutput.getString("script");
                int vout = unspentOutput.getInt("tx_output_n");
                int value = unspentOutput.getInt("value");

                utxos.add(new UTXO(txid, scriptPubKey, vout, value));
            }
        }
    }
    return utxos;
}
```

8

11

Potenzielle Verbesserungen für die hypothetische Version 2 der Wallet

- Speicherung der des privaten Schlüssels in einer JSON Datei:



9

13

Erhöhung der Sicherheit bei Speicherung des privaten Schlüssels in einer Datei

Bei Speicherung der des privaten Schlüssels in einer JSON Datei wird immer der gleiche Salt-Wert genutzt:

```
        )
        IvParameterSpec ivParam = generateIv();
        String encrypt = encryptPasswordBased(privateKey.toString(16), getKeyFromPassword(password, "salt"), ivParam);
        JSONObject pwFile = new JSONObject();

```

Folgende Verbesserung würden die Entropie und Sicherheit an dieser Stelle steigen:

1. Verwendung eines sicheren Zufallsgenerators für die Erstellung des Salt wertes. Dies würde die Zufälligkeit und Einmaligkeit des Salt-Wertes gewährleisten
2. Verlängerung des Salt-Wertes zum 128 bit Format

10

14

2FA Zwei-Faktor-Authentifizierung

Die Integration der zwei-Faktor-Authentifizierung wäre an den folgenden Stellen möglich:

1. Eingabe des OTP-Codes während des Anmeldevorgangs, um seine Identität zu bestätigen
2. Eingabe des OTP-Codes zur Bestätigung und Signierung der Transaktion
3. Eingabe des OTP-Codes bei Änderung des Passworts

11

15

2FA Sicherheit

1. Schutz vor gestohlenen Zugangsdaten
2. Verhinderung von Phishing-Angriffen
3. Schutz vor Brute-Force-Angriffen
4. Erhöhte Sicherheit bei Verlust oder Diebstahl des Geräts

12

16

Sicherheit die durch die Wallet gewährleistet wird

- Verwendung des secp256k1 Algorithmus zur Generierung des privaten und öffentlichen Schlüssels

```
55
56      // Schlüsselpaar generieren
57      public void generateKeyPair()
58              throws NoSuchAlgorithmException, NoSuchProviderException, InvalidAlgorithmParameterException {
59
60          // BouncyCastleProvider asd;
61
62          // BouncyCastleProvider hinzufügen
63          Security.addProvider(new BouncyCastleProvider());
64          KeyPairGenerator ecKeyGen = KeyPairGenerator.getInstance("EC", "BC");
65          ECGenParameterSpec ecSpec = new ECGenParameterSpec("secp256k1");
66          ecKeyGen.initialize(ecSpec);
67          ecKeyPair = ecKeyGen.generateKeyPair();
68          ecPubKey = (ECPublicKey) ecKeyPair.getPublic();
69          ecPrivKey = (ECPrivateKey) ecKeyPair.getPrivate();
70      }
71
```

13

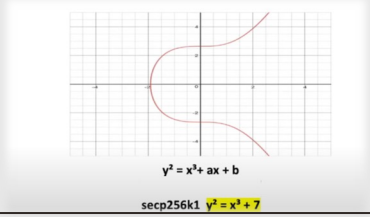
17

secp256k1 Algorithmus Sicherheit

Einige der wichtigsten Sicherheitsaspekte, die durch die Verwendung des secp256k1-Algorithmus erreicht werden:

1. Schlüssellänge von 256 Bit
2. Immunität gegen Angriffe auf Hash Funktionen
3. Wird als sicherer Standard für kryptographische Methoden anerkannt

(Secure Hash Algorithm 256-bit)



14

18

Speicherung des Privaten Schlüssels in einer verschlüsselten JSON-Datei

Die Verschlüsselung erfolgt anhand des Passworts des Salt-Werts und Initialisierungsvektor

Sicherheitsaspekte:

-Der private Schlüssel kann nicht einfach durch den Angreifer ausgelesen werden

-Durch die zusätzliche Verwendung des Initialisierungsvektors (IV) wird jede Datei mit dem gleichen Passwort und Salt-Wert anders verschlüsselt

15

19

Sicherheit die durch die Blockchain gewährleistet wird

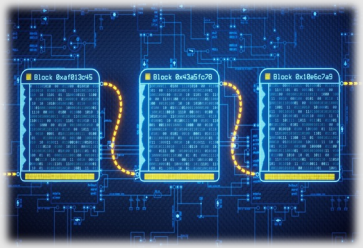
- **Dezentraliesierung** - Die Daten werden auf vielen Knoten im Netzwerk verteilt und gemeinsam verwaltet, was die Sicherheit und Robustheit erhöht.
- **Konsequenzmechanismus** - alle Knoten im Netzwerk die gleiche Version der Blockchain haben und nur gültige Transaktionen werden akzeptiert
- **Unveränderlichkeit** - Wenn ein Block einmal in die Blockchain aufgenommen wurde, kann er nicht mehr verändert werden, da sich sonst der Hash ändern würde.
- **Kryptografie** - Jeder Block in der Blockchain enthält einen einzigartigen Hash, der Manipulationen der Daten erschwert

16

20

Vorteile der Blockchain Technologie

1. Dezentralisierung
2. Erhöhte Sicherheit
3. Erhöhte Transparenz
4. Zuverlässigkeit
5. Effizienz



17

21

Nachteile der Blockchain Technologie

1. Probleme mit der Skalierbarkeit
2. Kostenaufwendiger Energieverbrauch
3. Regulatorische Unsicherheit
4. Verlust des Privaten Schlüssels

18

22

Weitere Anwendungsfehler der Blockchain Technologie

1. Mangelnde Interoperabilität
2. Begrenzte Transaktionsgeschwindigkeit
3. Unumkehrbare Transaktionen

19

23

Was betrachten wir heute?

Folgende Funktionen haben wir aus Sicht der Informationssicherheit, BSI-Grundschutz, Schutzbedarfsstellung, Risikoanalyse analysiert.

SHA25 Hashfunktionen

AES-Algorithmus

DER Kodierung von Datenstrukturen

ECDSA asymmetrisches Kryptosystem

20

24

SHA256 Hashfunktion

Informationssicherheit

- Hashfunktion wird verwendet, um eine Nachricht oder eine Datenmenge zu verarbeiten und eine feste, eindeutige Ausgabe zu erzeugen.
- Zwecken: Überprüfung der Integrität von Daten, zur Authentifizierung von Nachrichten oder zur Generierung von Schlüsseln

BSI-Grundschutz

- empfohlenen Hash-Funktionen für die sichere Speicherung von Passwörtern und zur Integritätssicherung von Daten empfohlen.
- SHA-256 eine stark kollisionsresistente Hash-Funktion, Achtung vor der sicheren Nutzung & Schutz vor bekannten Angriffen wie Brute-Force-Angriffen.

Schutzbedarfsstellung
Zielsetzung
Technische Sicherheit
Kommunikationssicherheit
Notfallmanagement
Rechtskonformität

Risikoanalyse

- Identifizierte Risiken
- Bewertung der Risiken:
- Eintrittswahrscheinlichkeit: Moderat bis hoch
- Auswirkungen: Hoch

Risikobewertung
Risikobehandlung
Risikomonitoring
Empfohlene Maßnahmen zur Risikobehandlung

21

25

AES-Algorithmus

Informationssicherheit

- kryptographischer Algorithmus, der für die Verschlüsselung von sensiblen Daten verwendet wird.
- Schlüssellängen 128, 192 und 256 Bit
- symmetrischen Verschlüsselungsverfahrens (Verschlüsselung & Entschlüsselung)

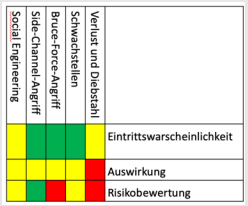
BSI-Grundschutz

- bevorzugten Verschlüsselungsmethoden
- geeigneten Schlüssellängen und Verschlüsselungsmodi
- Schlüsselverwaltung, kryptographische Sicherheitsmechanismen und die Integration

Schutzbedarfsstellung

- **Vertraulichkeit** (hohen Niveau, beschränkter Zugriff, Geeignete Verschlüsselungsmaßnahmen und Kontrollen)
- **Integrität** (Gewährleistung der Datenintegrität, Hash-Verfahren)
- **Verfügbarkeit: (**Gewährleistung einer kontinuierlichen Verfügbarkeit)
- **Authentizität** (Gewährleitung Digitale Signaturen)
- **Verbindlichkeit** (Protokollierung und Überwachung)

Risikoanalyse



22

26

DER Kodierung

Informationssicherheit
- Kodierung von Datenstrukturen, basiert auf dem Basic Encoding Rules (BER)-Format
- Public Key Infrastructure (PKI), X.509-Zertifikate ⟶ SSL/TLS, S/MIME
- sichere Übertragung und Speicherung von Zertifikaten, Public Keys, Signaturen und anderen kryptografischen Daten.

BSI-Grundschutz
- Standardkodierung für Zertifikate und Schlüssel
- X.509-Zertifikaten -> Public Key Infrastructure (PKI), Feststellung Zertifikate und Schlüssel korrekt kodiert und übertragen

Schutzbedarfsstellung

Schutzbedarfskategorie: Hoher Schutzbedarf

Schutzbedarfsziel: Gewährleistung der Integrität, Vertraulichkeit & Verfügbarkeit von Daten
- Bedrohungen und Risiken
- Schutzmaßnahmen
- Organisatorische Maßnahmen
- Technische Maßnahmen

Risikoanalyse
- Unzureichende Validierung
- Sicherheitsrisiken

23

27

ECDSA

Informationssicherheit
- basiert auf elliptischen Kurven und wird zur Erzeugung und Überprüfung digitaler Signaturen verwendet
- Sicherheit durch elliptische Kurven

BSI-Grundschutz
- **Schlüssellängen** (variiert und sollte möglichst lang gewählt werden)
- **Schlüsselverwaltung** (sichere Generierung der Schlüssel, der Schutz vor unbefugtem Zugriff)
- **Schlüsselspeicherung** (Hardware-Sicherheitsmodulen (HSM) oder sicheren Schlüsselverwaltungssystemen)

Schutzbedarfsstellung
- **Authentifizierung und Zugriffskontrolle** (Verwendung starker Authentifizierungsmethoden)
- **Verschlüsselung** (Einsatz von sicheren Verschlüsselungsalgorithmen)
- **Sicherheitsüberwachung und Ereignisprotokollierung** (Überwachungs- und Protokollierungsmechanismen)

Risikoanalyse
- Unbefugter Zugriff auf private ECDSA-Schlüssel
- Schwache Schlüssellängen
- Fehlende oder unsichere Schlüsselverwaltung
- Mangelnde Überwachung und Protokollierung
- Fehlende regelmäßige Sicherheitsbewertungen und Updates

24

28

Identifizierung und Bewertung aller Angriffsvektoren

1. Phishing-Angriffe
   gefälschte Websites oder E-Mails hereinfallen
2. Malware und Keylogger
   Schadsoftware und Keylogger
3. Social Engineering
   gefälschte Kundensupport-Anfragen oder betrügerische Investitionsangebot
4. Wallet-Hacks
   Sicherheitslücken in der Software oder menschliches Versagen zurückzuführen
5. 51%-Angriffe
   Bitcoin-Technologie und ihre Sicherheitspraktiken ständig weiterentwickelt

25

29

Vielen Dank

26

30