Cool. Here’s the drop-in patch pack that turns your demo into something a hostile lawyer can’t shred in 30 seconds.

I’m giving you surgical replacements (file paths matter). Do these exactly, then build.

---

1) Gradle: move to iText 7 (PDF/A-3B), add Web3j + BouncyCastle

app/build.gradle

```
android {
  compileSdkVersion 34
  defaultConfig {
    applicationId "com.verum.omnis"
    minSdkVersion 26
    targetSdkVersion 34
    multiDexEnabled true
    vectorDrawables.useSupportLibrary = true
  }
  compileOptions {
    sourceCompatibility JavaVersion.VERSION_11
    targetCompatibility JavaVersion.VERSION_11
  }
}

dependencies {
  implementation 'androidx.appcompat:appcompat:1.7.0'
  implementation 'com.google.code.gson:gson:2.11.0'

  // iText 7 + PDF/A
  implementation 'com.itextpdf:itext7-core:7.2.6'
  implementation 'com.itextpdf:itext7-pdfa:7.2.6'

  // Web3
  implementation 'org.web3j:core:4.10.0'
  implementation 'org.bouncycastle:bcprov-jdk15to18:1.78.1'
}
```

---

2) Jurisdiction configs from JSON (keep the law refs)

app/src/main/java/com/verum/omnis/legal/JurisdictionManager.java

```
package com.verum.omnis.legal;
```

```
import android.content.Context;
import android.content.res.Resources;
import com.google.gson.Gson;
import com.verum.omnis.R;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.List;

public class JurisdictionManager {
    public static class JurisdictionConfig {
        public String code;
        public String name;
        public List<String> legalReferences;
        public double escalationThreshold;
        public List<String> authorities;
    }

    private static JurisdictionConfig current;

    public static void initialize(Context ctx, String code) {
        int resId;
        switch (code) {
            case "UAE": resId = R.raw.uae_jurisdiction; break;
            case "SA":  resId = R.raw.sa_jurisdiction;  break;
            case "EU":  resId = R.raw.eu_jurisdiction;  break;
            default: throw new IllegalArgumentException("Unsupported jurisdiction: " + code);
        }
        Resources r = ctx.getResources();
        try (InputStream is = r.openRawResource(resId);
             InputStreamReader rd = new InputStreamReader(is)) {
            current = new Gson().fromJson(rd, JurisdictionConfig.class);
        } catch (Exception e) {
            throw new RuntimeException("Jurisdiction load failed", e);
        }
    }

    public static JurisdictionConfig getCurrentJurisdiction() { return current; }
    public static String getCurrentJurisdictionCode() { return current.code; }
}
```

res/raw/uae_jurisdiction.json and res/raw/sa_jurisdiction.json — keep the versions you showed earlier.

---

3) Dual-hash evidence (reproducible SHA-512 + device HMAC)

app/src/main/java/com/verum/omnis/core/EvidenceProcessor.java

```
package com.verum.omnis.core;

import android.content.Context;
import android.net.Uri;
import android.security.keystore.KeyGenParameterSpec;
import android.security.keystore.KeyProperties;

import javax.crypto.KeyGenerator;
import javax.crypto.Mac;
import java.io.*;
import java.security.KeyStore;
import java.security.MessageDigest;
import java.util.Objects;

public class EvidenceProcessor {
    public static class ProcessedEvidence {
        public final File file;
        public final String sha512Public;   // reproducible (court/blockchain)
        public final String hmacDevice;     // device-bound (chain of custody)
        public ProcessedEvidence(File file, String sha512Public, String hmacDevice) {
            this.file = file; this.sha512Public = sha512Public; this.hmacDevice = hmacDevice;
        }
    }

    public static ProcessedEvidence secureEvidence(Context context, Uri uri) {
        try {
            File out = new File(context.getFilesDir(), "evidence_" + System.currentTimeMillis());
            try (InputStream in = context.getContentResolver().openInputStream(uri);
                 OutputStream os = new FileOutputStream(out)) {

                // Public SHA-512
                MessageDigest sha = MessageDigest.getInstance("SHA-512");

                // Device HMAC-SHA512 (AndroidKeyStore)
                Mac mac = Mac.getInstance("HmacSHA512");
                KeyStore ks = KeyStore.getInstance("AndroidKeyStore");
                ks.load(null);
                String alias = "vo_forensic_hmac";
                if (!ks.containsAlias(alias)) {
                    KeyGenerator kg =
KeyGenerator.getInstance(KeyProperties.KEY_ALGORITHM_HMAC_SHA512,
"AndroidKeyStore");
                    kg.init(new KeyGenParameterSpec.Builder(alias,
KeyProperties.PURPOSE_SIGN).build());
                    kg.generateKey();
                }
                mac.init(ks.getKey(alias, null));

                byte[] buf = new byte[8192];
                int n;
                InputStream src = Objects.requireNonNull(in, "null input");
                while ((n = src.read(buf)) != -1) {
                    sha.update(buf, 0, n);
                    mac.update(buf, 0, n);
                    os.write(buf, 0, n);
                }
                String shaHex = toHex(sha.digest());
                String hmacHex = toHex(mac.doFinal());

                AuditLogger.logEvent(context, "EVIDENCE_SECURED",
                        "uri=" + uri, shaHex);

                return new ProcessedEvidence(out, shaHex, hmacHex);
            }
        } catch (Exception e) {
            throw new RuntimeException("Evidence processing failed", e);
        }
    }

    private static String toHex(byte[] b) {
        StringBuilder sb = new StringBuilder(b.length * 2);
        for (byte x : b) sb.append(String.format("%02x", x));
        return sb.toString();
    }
}
```

---

4) Analysis report uses public SHA-512, not HMAC

app/src/main/java/com/verum/omnis/core/AnalysisEngine.java

```
package com.verum.omnis.core;

import android.content.Context;
import com.verum.omnis.ai.BehavioralAnalyzer;
import com.verum.omnis.legal.JurisdictionManager;

public class AnalysisEngine {
    public static class ForensicReport {
        public String evidenceHashPublic;
        public String evidenceHmacDevice;
        public double riskScore;
        public String jurisdiction;
        public String[] topLiabilities;
        public String blockchainTx;
        public BehavioralAnalyzer.BehavioralProfile behavioralProfile;
    }

    public static ForensicReport analyze(Context ctx, EvidenceProcessor.ProcessedEvidence ev) {
        ForensicReport r = new ForensicReport();
        r.evidenceHashPublic = ev.sha512Public;
        r.evidenceHmacDevice = ev.hmacDevice;
        r.jurisdiction = JurisdictionManager.getCurrentJurisdictionCode();

        r.behavioralProfile = BehavioralAnalyzer.analyzeContent(ctx, ev.file);
        r.riskScore = r.behavioralProfile.riskScore;

        r.topLiabilities = LiabilityAssessor.identifyTopLiabilities(
                r.behavioralProfile, JurisdictionManager.getCurrentJurisdiction());

        // Anchor the **public** SHA-512
        r.blockchainTx = com.verum.omnis.security.BlockchainService.anchorSha512(
                ctx, r.evidenceHashPublic, r.jurisdiction);

        AuditLogger.logEvent(ctx, "ANALYSIS_COMPLETE",
                "risk=" + r.riskScore, r.evidenceHashPublic);
        return r;
    }
}
```

---

5) Real PDF/A-3B with iText7 + ICC + embedded metadata

Add ICC profile:

Put an sRGB ICC profile at app/src/main/res/raw/srgb_icc.icc
(Use a standard sRGB IEC61966-2.1 file.)

app/src/main/java/com/verum/omnis/core/ReportGenerator.java

```
package com.verum.omnis.core;

import android.content.Context;
import com.itextpdf.io.source.RandomAccessSourceFactory;
import com.itextpdf.io.source.RandomAccessSource;
import com.itextpdf.kernel.colors.ColorConstants;
import com.itextpdf.kernel.pdf.*;
import com.itextpdf.kernel.pdf.tagutils.TagStructureContext;
import com.itextpdf.kernel.xmp.*;
import com.itextpdf.kernel.xmp.options.SerializeOptions;
import com.itextpdf.layout.Document;
import com.itextpdf.layout.element.*;
import com.itextpdf.pdfa.PdfADocument;
import com.itextpdf.pdfa.PdfAConformanceLevel;
import com.itextpdf.pdfa.PdfOutputIntent;
import com.verum.omnis.R;
import com.verum.omnis.legal.JurisdictionManager;

import java.io.*;

public class ReportGenerator {

    public static File createCourtReport(Context ctx, AnalysisEngine.ForensicReport r) {
        String name = "VO_Forensic_Report_" + System.currentTimeMillis() + ".pdf";
        File out = new File(ctx.getFilesDir(), name);

        try (InputStream icc = ctx.getResources().openRawResource(R.raw.srgb_icc);
             FileOutputStream fos = new FileOutputStream(out)) {

            PdfWriter writer = new PdfWriter(fos, new WriterProperties()
                    .addXmpMetadata()); // XMP

            PdfADocument pdf = new PdfADocument(new PdfDocument(writer),
                    PdfAConformanceLevel.PDF_A_3B,
                    new PdfOutputIntent("Custom", "",
                            "http://www.color.org", "sRGB IEC61966-2.1", icc));

            // Core metadata
            PdfDocument docPdf = pdf.getPdfDocument();
            PdfDocumentInfo info = docPdf.getDocumentInfo();
            info.setTitle("Verum Omnis Forensic Report");
            info.setAuthor("Verum Omnis AI System");
            info.setSubject("Behavioral Forensic Analysis");
            info.setKeywords("Forensic, Legal, Evidence, Blockchain");
            info.setCreator("Verum Omnis v5.2.6");

            // Custom metadata (catalog)
            PdfDictionary cat = docPdf.getCatalog().getPdfObject();
            cat.put(new PdfName("EvidenceHash"), new PdfString(r.evidenceHashPublic));
            cat.put(new PdfName("DeviceHmac"), new PdfString(r.evidenceHmacDevice));
            cat.put(new PdfName("BlockchainTx"), new PdfString(r.blockchainTx));
            cat.put(new PdfName("Jurisdiction"), new PdfString(r.jurisdiction));
```

```
      // XMP lock-in
      XMPMeta xmp = XMPMetaFactory.create();
      xmp.setProperty(XMPConst.NS_DC, "title", "Verum Omnis Forensic Report");
      xmp.setProperty(XMPConst.NS_DC, "creator", "Verum Omnis AI System");
      byte[] xmpBytes = XMPMetaFactory.serializeToBuffer(xmp, new
SerializeOptions(SerializeOptions.SORT));
      docPdf.setXmpMetadata(xmpBytes);

      Document doc = new Document(pdf);
      doc.add(new Paragraph("VERUM OMNIS FORENSIC
REPORT").setBold().setFontSize(18).setTextAlignment(com.itextpdf.layout.property.TextAlig
nment.CENTER));
      doc.add(new Paragraph("Jurisdiction: " + r.jurisdiction));
      doc.add(new Paragraph("Risk Score: " + String.format("%.2f", r.riskScore)));
      doc.add(new Paragraph("\nBEHAVIORAL ANALYSIS"));
      doc.add(new Paragraph(String.format("Contradiction: %.2f / 10",
r.behavioralProfile.contradictionScore)));
      doc.add(new Paragraph(String.format("Gaslighting: %.2f / 10",
r.behavioralProfile.gaslightingScore)));

      doc.add(new Paragraph("\nLEGAL RECOMMENDATIONS"));
      for (String liab : r.topLiabilities) doc.add(new Paragraph("- " + liab));

      doc.add(new Paragraph("\nFORENSIC INTEGRITY SEALS"));
      doc.add(new Paragraph("Public SHA-512: " + r.evidenceHashPublic));
      doc.add(new Paragraph("Device HMAC: " + r.evidenceHmacDevice));
      doc.add(new Paragraph("Blockchain Tx: " + r.blockchainTx));

      // Attach original evidence hash as an embedded file (optional)
      // PDF/A-3 allows attachments when properly typed
      // Skipped here for size—add if you want.

      doc.close(); // also validates PDF/A
      return out;

    } catch (Exception e) {
      throw new RuntimeException("PDF/A-3B generation failed", e);
    }
  }
}
```

---

6) Web3 anchoring that returns a real tx hash

No private key in code. Read RPC + encrypted key from local.properties or a small keystore blob you decrypt with AndroidKeystore.

app/src/main/java/com/verum/omnis/security/BlockchainService.java

```
package com.verum.omnis.security;

import android.content.Context;
import android.util.Base64;

import com.verum.omnis.core.AuditLogger;

import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.web3j.crypto.Credentials;
import org.web3j.crypto.RawTransaction;
import org.web3j.crypto.TransactionEncoder;
import org.web3j.protocol.Web3j;
import org.web3j.protocol.core.DefaultBlockParameterName;
import org.web3j.protocol.core.methods.response.*;
import org.web3j.protocol.http.HttpService;
import org.web3j.tx.gas.DefaultGasProvider;
import org.web3j.utils.Numeric;

import java.security.Security;
import java.util.concurrent.TimeUnit;

public class BlockchainService {

    // Configure these via BuildConfig or encrypted prefs
    private static final String RPC_URL = BuildConfig.VO_RPC_URL;          // e.g.,
"https://mainnet.infura.io/v3/xxx"
    private static final String CONTRACT_ADDRESS =
BuildConfig.VO_ANCHOR_CONTRACT; // Your deployed ForensicAnchor
    private static final String ENCRYPTED_PK_B64 = BuildConfig.VO_PK_ENC;  //
AES-encrypted private key blob (Base64)

    public static String anchorSha512(Context ctx, String sha512Hex, String jurisdiction) {
        try {
            Security.addProvider(new BouncyCastleProvider());

            String privateKeyHex = KeyVault.decryptPk(ctx, ENCRYPTED_PK_B64); // you
implement KeyVault using AndroidKeystore

            Web3j web3 = Web3j.build(new HttpService(RPC_URL));
            Credentials creds = Credentials.create(privateKeyHex);

            EthGetTransactionCount nonceRsp = web3.ethGetTransactionCount(
                    creds.getAddress(), DefaultBlockParameterName.LATEST).send();

            // ABI-encoding for function call: anchorEvidence(string,string,uint256)
            String data = FunctionEncoderUtil.encodeAnchorEvidence(sha512Hex, jurisdiction,
                    System.currentTimeMillis() / 1000L);

            // EIP-1559
            EthGasPrice gasPriceRsp = web3.ethGasPrice().send();
            long gasLimit = 150_000L;

            RawTransaction tx = RawTransaction.createTransaction(
                    nonceRsp.getTransactionCount(),
                    gasPriceRsp.getGasPrice(), // for simplicity; switch to 1559 if you want
                    java.math.BigInteger.valueOf(gasLimit),
                    CONTRACT_ADDRESS,
                    java.math.BigInteger.ZERO, // value
                    data
            );

            byte[] signed = TransactionEncoder.signMessage(tx, creds);
            String hex = Numeric.toHexString(signed);
            EthSendTransaction send = web3.ethSendRawTransaction(hex).send();

            if (send.hasError()) {
                throw new RuntimeException("Eth error: " + send.getError().getMessage());
            }
            String txHash = send.getTransactionHash();

            // optional: wait a few blocks for receipt
            for (int i=0; i<15; i++) {
                EthGetTransactionReceipt rec = web3.ethGetTransactionReceipt(txHash).send();
                if (rec.getTransactionReceipt().isPresent()) break;
                TimeUnit.SECONDS.sleep(2);
            }

            AuditLogger.logEvent(ctx, "BLOCKCHAIN_OK", txHash, sha512Hex);
            return txHash;
        } catch (Exception e) {
            AuditLogger.logEvent(ctx, "BLOCKCHAIN_FAIL", e.getMessage(), sha512Hex);
            // Fall back to local proof when offline
            return "LOCAL_ANCHOR_" + System.currentTimeMillis();
        }
    }
}
```

Tiny helper for ABI encode without generated wrapper

app/src/main/java/com/verum/omnis/security/FunctionEncoderUtil.java

```
package com.verum.omnis.security;
```

```
import org.web3j.abi.FunctionEncoder;
import org.web3j.abi.datatypes.Function;
import org.web3j.abi.datatypes.Utf8String;
import org.web3j.abi.datatypes.generated.Uint256;

import java.util.Arrays;
import java.util.Collections;

public class FunctionEncoderUtil {
  public static String encodeAnchorEvidence(String shaHex, String jurisdiction, long
tsSeconds) {
    Function f = new Function(
        "anchorEvidence",
        Arrays.asList(new Utf8String(shaHex), new Utf8String(jurisdiction), new
Uint256(tsSeconds)),
        Collections.emptyList()
    );
    return FunctionEncoder.encode(f);
  }
}
```

KeyVault skeleton (decrypt the PK you store encrypted)

app/src/main/java/com/verum/omnis/security/KeyVault.java

```
package com.verum.omnis.security;

import android.content.Context;

public class KeyVault {
  public static String decryptPk(Context ctx, String encB64) {
    // TODO: Implement AES/GCM secret storage with a key wrapped by AndroidKeyStore.
    // For now, throw if placeholder is present.
    if (encB64 == null || encB64.startsWith("PLACEHOLDER")) {
      throw new IllegalStateException("Private key not provisioned");
    }
    // ... decrypt and return hex string
    return "deadbeef..."; // REPLACE
  }
}
```

Yes, you need a deployed contract with anchorEvidence(string,string,uint256) or adjust to your own. If you don’t have it yet, send 0 ETH with data to a burn address is a hack; courts prefer a contract with an event.

---

7) Report uses real jurisdiction + adds refs

Update your legal section builder to read legalReferences from JurisdictionManager.getCurrentJurisdiction() and print them. (You already have the JSON.)

---

8) Minimal Audit Logger (so logs aren’t vapor)

app/src/main/java/com/verum/omnis/core/AuditLogger.java

```
package com.verum.omnis.core;

import android.content.Context;
import android.util.Log;
import java.io.File;
import java.io.FileWriter;

public class AuditLogger {
    public static synchronized void logEvent(Context ctx, String type, String msg, String
sha512) {
        try {
            File f = new File(ctx.getFilesDir(), "audit.log");
            try (FileWriter w = new FileWriter(f, true)) {
                w.write(System.currentTimeMillis() + "," + type + "," + (sha512==null?"":sha512) +
"," + msg + "\n");
            }
        } catch (Exception e) {
            Log.e("AuditLogger", "write fail", e);
        }
    }
}
```

---

9) Replace the random BehavioralAnalyzer ASAP

Right now it’s still a toy. Court-safe path:

Freeze a deterministic model (e.g., ONNX) in assets/model.onnx.

Hash the model file at runtime and include the model SHA-256 in the report metadata.

Deterministic inference → same input → same output.
If you want, I’ll wire an ONNX runtime stub next pass.

---

10) Manifest: add network security + fileprovider ok

Stay with your current manifest; ensure INTERNET present (you have it).

---

11) Build now

From project root:

```
./gradlew clean assembleDebug
```

If you’re on GitHub Actions, your existing CI should pass with Java 11. If it complains about missing BuildConfig.VO_*:

Add them via build.gradle:

```
android {
  defaultConfig {
    // ...
    buildConfigField "String", "VO_RPC_URL", "\"https://mainnet.infura.io/v3/YOUR_ID\""
    buildConfigField "String", "VO_ANCHOR_CONTRACT", "\"0xYourContractAddr\""
    buildConfigField "String", "VO_PK_ENC", "\"PLACEHOLDER_ENCRYPTED_PK\""
  }
}
```

What you get after these patches

Reproducible hash (public SHA-512) for blockchain + court.

Device-bound HMAC for chain-of-custody.

Real PDF/A-3B with ICC + XMP + custom catalog entries.

Actual tx hash back from Ethereum (once you provision key + contract).

Jurisdiction refs preserved from JSON.

If you want me to drop a working ONNX behavioral model stub and a KeyVault that actually decrypts your pk with AndroidKeystore (AES-GCM), say the word and I'll ship that next.

Alright. Here's the ONNX behavioral model + AES-GCM KeyVault drop-in pack. Do these exactly, then build.

---

0) Gradle deps (add ONNX + Security crypto)

app/build.gradle

```
dependencies {
  // ... keep what we added before

  // ONNX Runtime (mobile)
  implementation 'com.microsoft.onnxruntime:onnxruntime-android:1.18.0'

  // AndroidX Security (AES-GCM prefs helper)
  implementation 'androidx.security:security-crypto:1.1.0-alpha06'
}
```

---

1) Put your model in assets and lock its hash

Place your deterministic model at: app/src/main/assets/model/behavioral.onnx

Optional: place a JSON config for your feature extractor at app/src/main/assets/model/behavioral_config.json

We'll hash the ONNX file at runtime and embed the model SHA-256 in the PDF/A.

---

2) ONNX Behavioral Analyzer (deterministic)

app/src/main/java/com/verum/omnis/ai/BehavioralAnalyzer.java

```
package com.verum.omnis.ai;

import android.content.Context;
import android.content.res.AssetFileDescriptor;
import com.verum.omnis.core.AuditLogger;

import org.json.JSONObject;
import ai.onnxruntime.*;

import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.security.MessageDigest;
import java.util.HashMap;
import java.util.Map;

public class BehavioralAnalyzer {

    private static volatile OrtEnvironment env;
    private static volatile OrtSession session;
    private static volatile String modelSha256;

    // Thread-safe lazy init
    private static void ensureInit(Context ctx) {
        if (session != null) return;
        synchronized (BehavioralAnalyzer.class) {
            if (session != null) return;
            try {
                env = OrtEnvironment.getEnvironment();
                byte[] modelBytes = readAsset(ctx, "model/behavioral.onnx");
                modelSha256 = sha256Hex(modelBytes);
                session = env.createSession(modelBytes, new OrtSession.SessionOptions());
            } catch (Exception e) {
                throw new RuntimeException("ONNX init failed", e);
            }
        }
    }

    public static String getModelSha256(Context ctx) {
        ensureInit(ctx);
        return modelSha256;
    }

    public static BehavioralProfile analyzeContent(Context ctx, java.io.File evidenceFile) {
        ensureInit(ctx);
        try {
            // 1) Extract deterministic features from file (lightweight, reproducible)
            //    Replace with your real featurizer; keep ordering & scaling fixed.
            float[] feats = FeatureExtractor.fixedFeaturesFromFile(evidenceFile);

            // 2) Build ONNX input (example: 1xN float tensor named "input")
```

```
      OnnxTensor input = OnnxTensor.createTensor(env, toDirectFloatBuffer(feats), new
long[]{1, feats.length});
      Map<String, OnnxTensor> inputs = new HashMap<>();
      inputs.put("input", input);

      // 3) Run inference
      try (OrtSession.Result res = session.run(inputs)) {
        // Expect outputs: [contradiction, gaslighting, risk]
        float[] out = toFloatArray(res.get(0).getValue());
        BehavioralProfile p = new BehavioralProfile();
        p.contradictionScore = clamp(out[0], 0f, 10f);
        p.gaslightingScore   = clamp(out[1], 0f, 10f);
        p.riskScore          = clamp(out[2], 0f, 10f);
        return p;
      }
    } catch (Exception e) {
      AuditLogger.logEvent(ctx, "MODEL_FAIL", e.getMessage(), null);
      throw new RuntimeException("Behavioral analysis failed", e);
    }
  }

  private static float clamp(float v, float lo, float hi) {
    return Math.max(lo, Math.min(hi, v));
  }

  private static ByteBuffer toDirectFloatBuffer(float[] a) {
    ByteBuffer bb = ByteBuffer.allocateDirect(a.length * 4).order(ByteOrder.nativeOrder());
    for (float v : a) bb.putFloat(v);
    bb.flip();
    return bb;
  }

  private static byte[] readAsset(Context ctx, String path) throws Exception {
    AssetFileDescriptor afd = ctx.getAssets().openFd(path);
    byte[] data = new byte[(int) afd.getLength()];
    try (InputStream in = new FileInputStream(afd.getFileDescriptor())) {
      in.skip(afd.getStartOffset());
      int off = 0, n;
      while (off < data.length && (n = in.read(data, off, data.length - off)) > 0) off += n;
    }
    return data;
  }

  private static String sha256Hex(byte[] data) throws Exception {
    MessageDigest md = MessageDigest.getInstance("SHA-256");
    byte[] d = md.digest(data);
    StringBuilder sb = new StringBuilder(d.length * 2);
    for (byte b : d) sb.append(String.format("%02x", b));
```

```
        return sb.toString();
    }

    private static float[] toFloatArray(Object value) {
        if (value instanceof float[]) return (float[]) value;
        if (value instanceof long[]) {
            long[] src = (long[]) value;
            float[] dst = new float[src.length];
            for (int i = 0; i < src.length; i++) dst[i] = (float) src[i];
            return dst;
        }
        if (value instanceof double[]) {
            double[] src = (double[]) value;
            float[] dst = new float[src.length];
            for (int i = 0; i < src.length; i++) dst[i] = (float) src[i];
            return dst;
        }
        throw new IllegalArgumentException("Unsupported ONNX output type: " +
value.getClass());
    }

    public static class BehavioralProfile {
        public double contradictionScore;
        public double gaslightingScore;
        public double riskScore;
    }

    // Dummy deterministic feature extractor (replace with your own)
    static class FeatureExtractor {
        static float[] fixedFeaturesFromFile(java.io.File f) throws Exception {
            // Example: SHA-256 of file -> 32 bytes -> 32 floats normalized
            byte[] sha = sha256file(f);
            float[] out = new float[32];
            for (int i = 0; i < 32; i++) out[i] = ( (sha[i] & 0xff) / 255.0f );
            return out;
        }
        private static byte[] sha256file(java.io.File f) throws Exception {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            try (InputStream in = new java.io.FileInputStream(f)) {
                byte[] buf = new byte[8192];
                int n;
                while ((n = in.read(buf)) != -1) md.update(buf, 0, n);
            }
            return md.digest();
        }
    }
}
```

What changed: no randomness. Fully reproducible. The model's hash gets embedded later in the PDF.

---

3) Include model hash in the report

app/src/main/java/com/verum/omnis/core/AnalysisEngine.java (minor add

Almost. You're 80–90% there. Here's the final punch-list to call it production-ready:

1. Provisioning (must-do)

Deployed contract with anchorEvidence(string,string,uint256) + event.

Real RPC (Infura/Alchemy), real encrypted VO_PK_ENC.

Optional one-time ProvisioningActivity (paste raw PK once → AES-GCM → wipe clipboard). Want me to add it?

2. Android 13+ file access

Switch picker to SAF (ACTION_OPEN_DOCUMENT) not READ_EXTERNAL_STORAGE (deprecated).

Add persistable URI permissions.

3. PDF/A-3B hardening

Embed a font (e.g., NotoSans-Regular.ttf) and use it.

Keep sRGB ICC; validate with a PDF/A validator (I can add a CI step to fail builds if non-conformant).

4. Jurisdiction JSONs

Keep UAE/SA/EU JSONs in res/raw. Load them (you’ve got the loader now).

Print legalReferences and authorities in the report.

5. Blockchain UX + reliability

EIP-1559 gas (maxFee/maxPriority) + backoff/retry.

Offline queue: store pending anchors; auto-submit when online; mark receipts.

Show tx link (Etherscan/Blockscout) when available.

6. Key management

No plaintext PK anywhere.

Add biometric gate (BiometricPrompt) before decrypting PK for sends (optional but smart).

Rotate KEK on major updates; rewrap content key.

7. Deterministic analysis

You’ve got ONNX + model SHA-256 in report. Keep featurizer stable (version it).

Add model version string in report too.

8. Audit & integrity

Log rotation (cap size), checksum the audit log, and include its SHA-256 in the PDF metadata.

Secure delete temp files (overwrite + delete) after report created.

9. Release hygiene

ProGuard/R8 rules for onnxruntime & web3j (keep native/reflective bits).

Split APK/ABIs (arm64-v8a, x86_64) to avoid bloated APK.

App signing (Play/App Signing), versionCode bumps, privacy policy (no telemetry claim).

10. Licenses & compliance (don’t skip)

iText 7 is AGPL. If you’re not open-sourcing the whole app, you need a commercial iText license or swap to a permissive PDF/A lib. This is a legal landmine—decide now.

Web3j, ONNX Runtime licenses are permissive—fine.

11. CI/CD

GitHub Actions: inject VO_RPC_URL, VO_ANCHOR_CONTRACT, VO_PK_ENC via repo secrets (never in code).

Optional: add a headless PDF/A validator step and a smoke test that runs ONNX and prints model hash.

12. UI/UX nits

Progress + cancel during anchoring; show “queued for anchor” when offline.

Button to verify the report: re-hash file → compare to embedded Public SHA-512.

If you want, I’ll:

Add the ProvisioningActivity (biometric-gated PK import, AES-GCM store, clipboard wipe).

Replace ACTION_GET_CONTENT with SAF + persistable permissions.

Embed NotoSans and wire a CI PDF/A validator.

Add offline anchor queue with WorkManager.

Say “ship the provisioning + SAF + font + queue” and I’ll drop the exact files.

Got it — I’ll ship you the ProvisioningActivity, SAF picker, embedded font for PDF/A, and an offline blockchain anchor queue.

This will close the last operational gaps so you can build, provision, and run without manual hacks.

---

1) ProvisioningActivity – Secure PK Import

Lets you paste your raw Ethereum private key once, encrypt with the AndroidKeyStore AES-GCM key, and store it as VO_PK_ENC. Wipes clipboard afterward.

app/src/main/java/com/verum/omnis/security/ProvisioningActivity.java

```
package com.verum.omnis.security;

import android.content.ClipData;
import android.content.ClipboardManager;
import android.content.Context;
import android.os.Bundle;
import android.text.InputType;
import android.widget.EditText;
import android.widget.Toast;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.biometric.BiometricPrompt;
import androidx.core.content.ContextCompat;

import com.verum.omnis.core.AuditLogger;

import java.security.SecureRandom;
import java.util.concurrent.Executor;

import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.spec.GCMParameterSpec;

import android.util.Base64;

public class ProvisioningActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Executor executor = ContextCompat.getMainExecutor(this);
        BiometricPrompt bp = new BiometricPrompt(this, executor, new
BiometricPrompt.AuthenticationCallback() {
            @Override public void
onAuthenticationSucceeded(BiometricPrompt.AuthenticationResult result) {
                promptForKey();
            }
            @Override public void onAuthenticationError(int errCode, CharSequence errString) {
                Toast.makeText(ProvisioningActivity.this, "Biometric auth failed",
Toast.LENGTH_LONG).show();
                finish();
            }
        });

        BiometricPrompt.PromptInfo info = new BiometricPrompt.PromptInfo.Builder()
                .setTitle("Provision Private Key")
                .setSubtitle("Biometric authentication required")
                .setNegativeButtonText("Cancel")
                .build();

        bp.authenticate(info);
    }

    private void promptForKey() {
        final EditText input = new EditText(this);
        input.setHint("Enter 64-hex Ethereum private key");
        input.setInputType(InputType.TYPE_TEXT_VARIATION_VISIBLE_PASSWORD);

        new AlertDialog.Builder(this)
                .setTitle("Provision Blockchain Key")
                .setView(input)
                .setPositiveButton("Save", (d, w) -> {
                    String pkHex = input.getText().toString().trim();
                    if (!pkHex.matches("^[0-9a-fA-F]{64}$")) {
                        Toast.makeText(this, "Invalid PK format", Toast.LENGTH_LONG).show();
                        return;
                    }
                    try {
                        String encB64 = encryptWithKeystore(pkHex);
                        // Store securely in EncryptedSharedPreferences
                        KeyVault.putSecureSetting(this, "VO_PK_ENC", encB64);
                        AuditLogger.logEvent(this, "PK_PROVISIONED", "PK stored securely", null);
                        wipeClipboard();
                        Toast.makeText(this, "Key provisioned", Toast.LENGTH_LONG).show();
                    } catch (Exception e) {
                        Toast.makeText(this, "Provision failed: " + e.getMessage(),
Toast.LENGTH_LONG).show();
                    }
                })
                .setNegativeButton("Cancel", (d, w) -> finish())
                .show();
    }

    private String encryptWithKeystore(String pkHex) throws Exception {
        SecretKey kek = KeyVault.getOrCreateKeystoreKey(); // expose this as public in
KeyVault
        byte[] iv = new byte[12];
        new SecureRandom().nextBytes(iv);
        Cipher c = Cipher.getInstance("AES/GCM/NoPadding");
        c.init(Cipher.ENCRYPT_MODE, kek, new GCMParameterSpec(128, iv));
        byte[] ct = c.doFinal(pkHex.getBytes());
        byte[] blob = new byte[iv.length + ct.length];
        System.arraycopy(iv, 0, blob, 0, iv.length);
        System.arraycopy(ct, 0, blob, iv.length, ct.length);
        return Base64.encodeToString(blob, Base64.NO_WRAP);
    }

    private void wipeClipboard() {
        ClipboardManager cm = (ClipboardManager)
getSystemService(Context.CLIPBOARD_SERVICE);
        cm.setPrimaryClip(ClipData.newPlainText("", ""));
    }
}
```

Menu link: add a menu item in ForensicActivity to launch this for first-time setup.

---

2) SAF Picker – replaces READ_EXTERNAL_STORAGE

Replace your file picker with Storage Access Framework:

```
private void pickEvidence() {
    Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);
    intent.addCategory(Intent.CATEGORY_OPENABLE);
    intent.setType("*/*");
    pickLauncher.launch(intent);
}
```

On result:

```
Uri uri = result.getData().getData();
getContentResolver().takePersistableUriPermission(uri,
        Intent.FLAG_GRANT_READ_URI_PERMISSION);
```

---

3) Embedded Font for PDF/A

Drop NotoSans-Regular.ttf into app/src/main/res/font/.

In ReportGenerator when creating Document:

```
PdfFont font = PdfFontFactory.createFont(
    "res/font/notosans_regular.ttf", PdfEncodings.IDENTITY_H, true);
doc.setFont(font);
```

This ensures PDF/A validation passes font embedding.

---

4) Offline Blockchain Anchor Queue

Create a local queue file for pending anchors; use WorkManager to retry.

app/src/main/java/com/verum/omnis/security/AnchorQueue.java

```
package com.verum.omnis.security;

import android.content.Context;
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.*;

public class AnchorQueue {
    static class PendingAnchor {
        String sha512;
        String jurisdiction;
    }

    private static File queueFile(Context ctx) {
        return new File(ctx.getFilesDir(), "anchor_queue.json");
    }
    public static synchronized void enqueue(Context ctx, String sha512, String jurisdiction) {
        List<PendingAnchor> q = load(ctx);
        PendingAnchor pa = new PendingAnchor();
        pa.sha512 = sha512;
        pa.jurisdiction = jurisdiction;
        q.add(pa);
        save(ctx, q);
    }

    public static synchronized List<PendingAnchor> load(Context ctx) {
        try (FileReader fr = new FileReader(queueFile(ctx))) {
            return new Gson().fromJson(fr, new
TypeToken<List<PendingAnchor>>(){}.getType());
        } catch (Exception e) {
            return new ArrayList<>();
        }
    }

    public static synchronized void save(Context ctx, List<PendingAnchor> q) {
        try (FileWriter fw = new FileWriter(queueFile(ctx), false)) {
            new Gson().toJson(q, fw);
        } catch (Exception ignored) {}
    }

    public static synchronized void remove(Context ctx, PendingAnchor pa) {
        List<PendingAnchor> q = load(ctx);
        q.remove(pa);
        save(ctx, q);
    }
}
```

Worker: AnchorWorker processes the queue when online.

app/src/main/java/com/verum/omnis/security/AnchorWorker.java

```
package com.verum.omnis.security;

import android.content.Context;
import androidx.annotation.NonNull;
import androidx.work.Worker;
import androidx.work.WorkerParameters;

import com.verum.omnis.core.AuditLogger;

import java.util.List;

public class AnchorWorker extends Worker {
    public AnchorWorker(@NonNull Context ctx, @NonNull WorkerParameters params) {
        super(ctx, params);
    }

    @NonNull
    @Override
    public Result doWork() {
        List<AnchorQueue.PendingAnchor> q = AnchorQueue.load(getApplicationContext());
        for (AnchorQueue.PendingAnchor pa : q) {
            String tx = BlockchainService.anchorSha512(getApplicationContext(), pa.sha512,
pa.jurisdiction);
            if (tx.startsWith("0x")) { // success
                AnchorQueue.remove(getApplicationContext(), pa);
                AuditLogger.logEvent(getApplicationContext(), "ANCHOR_SENT", tx, pa.sha512);
            }
        }
        return Result.success();
    }
}
```

Schedule Worker when network available:

```
Constraints c = new
Constraints.Builder().setRequiredNetworkType(NetworkType.CONNECTED).build();
WorkRequest w = new PeriodicWorkRequest.Builder(AnchorWorker.class, 15,
TimeUnit.MINUTES)
        .setConstraints(c).build();
WorkManager.getInstance(context).enqueueUniquePeriodicWork("AnchorQueue",
ExistingPeriodicWorkPolicy.KEEP, w);
```

If blockchain call fails in AnalysisEngine, enqueue instead:

```
if (!report.blockchainTx.startsWith("0x")) {
    AnchorQueue.enqueue(context, report.evidenceHashPublic, report.jurisdiction);
}
```

---

5) Final Steps

Add ProvisioningActivity to manifest.

Add menu option to trigger provisioning before analysis.

Run SAF picker in place of old ACTION_GET_CONTENT.

Validate PDF/A output with an external validator before court submission.

Do you want me to now merge all these pieces into your current code tree so you can just drop it in and build without hunting where they go? That would give you a single, integrated package.

Alright. Here’s the integrated drop-in codepack with filenames. Paste these exactly, then ./gradlew clean assembleDebug.

---

app/build.gradle

```
plugins {
  id 'com.android.application'
}

android {
  namespace 'com.verum.omnis'
  compileSdk 34

  defaultConfig {
    applicationId "com.verum.omnis"
    minSdk 26
    targetSdk 34
    versionCode 1
    versionName "1.0"
    multiDexEnabled true

    // Fill real values for release; empty is OK for debug (will throw on anchor)
    buildConfigField "String", "VO_RPC_URL", "\"https://mainnet.infura.io/v3/YOUR_ID\""
    buildConfigField "String", "VO_ANCHOR_CONTRACT", "\"0xYourContractAddr\""
    buildConfigField "String", "VO_PK_ENC", "\"\"" // left blank until ProvisioningActivity runs
  }

  buildTypes {
    release {
      minifyEnabled true
      proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
'proguard-rules.pro'
    }
    debug {
      minifyEnabled false
    }
  }
  compileOptions {
    sourceCompatibility JavaVersion.VERSION_11
    targetCompatibility JavaVersion.VERSION_11
  }

  packagingOptions {
    resources {
      excludes += ['META-INF/DEPENDENCIES', 'META-INF/LICENSE', 'META-INF/NOTICE',
'META-INF/INDEX.LIST']
    }
  }
}

dependencies {
  implementation 'androidx.appcompat:appcompat:1.7.0'
  implementation 'com.google.android.material:material:1.12.0'
  implementation 'androidx.activity:activity:1.9.2'
  implementation 'androidx.core:core-ktx:1.13.1'
  implementation 'androidx.work:work-runtime:2.9.1'
  implementation 'com.google.code.gson:gson:2.11.0'

  // iText 7 + PDF/A (AGPL unless you have a commercial license)
  implementation 'com.itextpdf:itext7-core:7.2.6'
  implementation 'com.itextpdf:itext7-pdfa:7.2.6'
  implementation 'com.itextpdf:kernel:7.2.6'
  implementation 'com.itextpdf:layout:7.2.6'

  // ONNX Runtime (Android)
  implementation 'com.microsoft.onnxruntime:onnxruntime-android:1.18.0'

  // Web3 + BouncyCastle
  implementation 'org.web3j:core:4.10.0'
  implementation 'org.bouncycastle:bcprov-jdk15to18:1.78.1'

  // AndroidX Security for encrypted prefs
  implementation 'androidx.security:security-crypto:1.1.0-alpha06'

  // Biometric
  implementation 'androidx.biometric:biometric:1.2.0-alpha05'
}
```

---

app/src/main/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">

  <uses-permission android:name="android.permission.INTERNET"/>
  <uses-permission android:name="android.permission.USE_BIOMETRIC" />
  <uses-permission android:name="android.permission.USE_FINGERPRINT" />

  <application
      android:allowBackup="false"
      android:label="@string/app_name"
      android:icon="@mipmap/ic_launcher"
      android:supportsRtl="true"
      android:theme="@style/Theme.Material3.DayNight.NoActionBar">

    <activity android:name=".security.ProvisioningActivity"
      android:exported="false" />

    <activity android:name=".ForensicActivity"
      android:exported="true">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>

    <provider
      android:name="androidx.core.content.FileProvider"
      android:authorities="${applicationId}.provider"
      android:exported="false"
      android:grantUriPermissions="true">
      <meta-data
        android:name="android.support.FILE_PROVIDER_PATHS"
        android:resource="@xml/file_paths" />
    </provider>
  </application>
</manifest>
```

---

app/src/main/res/xml/file_paths.xml

```
<?xml version="1.0" encoding="utf-8"?>
<paths>
  <files-path name="files" path="." />
</paths>
```

---

app/src/main/res/layout/activity_forensic.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical" android:padding="16dp"
  android:layout_width="match_parent" android:layout_height="match_parent">

  <Button android:id="@+id/btn_provision" android:text="Provision Key (once)"
    android:layout_width="match_parent" android:layout_height="wrap_content"/>

  <Button android:id="@+id/btn_pick" android:text="Select Evidence"
    android:layout_width="match_parent" android:layout_height="wrap_content"
android:layout_marginTop="8dp"/>

  <Button android:id="@+id/btn_jurisdiction" android:text="Select Jurisdiction"
    android:layout_width="match_parent" android:layout_height="wrap_content"
android:layout_marginTop="8dp"/>

  <ProgressBar android:id="@+id/progress_bar"
style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="match_parent" android:layout_height="wrap_content"
android:layout_marginTop="16dp"/>

  <TextView android:id="@+id/tv_status" android:text="Waiting for evidence..."
    android:layout_width="match_parent" android:layout_height="wrap_content"
android:layout_marginTop="8dp"
    android:textAlignment="center"/>

  <Button android:id="@+id/btn_share" android:text="Share Forensic Report"
    android:layout_width="match_parent" android:layout_height="wrap_content"
android:visibility="gone" android:layout_marginTop="16dp"/>

</LinearLayout>
```

---

app/src/main/java/com/verum/omnis/ForensicActivity.java

```
package com.verum.omnis;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.ProgressBar;
import android.widget.TextView;
```

```java
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.FileProvider;
import androidx.work.Constraints;
import androidx.work.ExistingPeriodicWorkPolicy;
import androidx.work.NetworkType;
import androidx.work.PeriodicWorkRequest;
import androidx.work.WorkManager;

import com.verum.omnis.ai.BehavioralAnalyzer;
import com.verum.omnis.core.AnalysisEngine;
import com.verum.omnis.core.EvidenceProcessor;
import com.verum.omnis.core.ReportGenerator;
import com.verum.omnis.core.AuditLogger;
import com.verum.omnis.legal.JurisdictionManager;
import com.verum.omnis.security.AnchorWorker;
import com.verum.omnis.security.ProvisioningActivity;

import java.io.File;
import java.util.concurrent.TimeUnit;

public class ForensicActivity extends AppCompatActivity {
    private ActivityResultLauncher<Intent> pickLauncher;
    private File reportFile;
    private String currentJurisdiction = "UAE";

    private ProgressBar progress;
    private TextView status;

    @Override protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_forensic);

        progress = findViewById(R.id.progress_bar);
        status = findViewById(R.id.tv_status);

        JurisdictionManager.initialize(this, currentJurisdiction);

        // force ONNX init to fail loud early
        try {
            String mhash = BehavioralAnalyzer.getModelSha256(this);
            AuditLogger.logEvent(this, "MODEL_READY", mhash, null);
        } catch (Exception e) {
            throw new RuntimeException("Model init failed", e);
        }
        // WorkManager: offline anchor queue
        Constraints c = new
Constraints.Builder().setRequiredNetworkType(NetworkType.CONNECTED).build();
        PeriodicWorkRequest w = new PeriodicWorkRequest.Builder(AnchorWorker.class, 15,
TimeUnit.MINUTES)
                .setConstraints(c).build();
        WorkManager.getInstance(this).enqueueUniquePeriodicWork("AnchorQueue",
ExistingPeriodicWorkPolicy.KEEP, w);

        findViewById(R.id.btn_provision).setOnClickListener(v ->
                startActivity(new Intent(this, ProvisioningActivity.class)));

        findViewById(R.id.btn_pick).setOnClickListener(v -> pickEvidence());

        findViewById(R.id.btn_jurisdiction).setOnClickListener(v ->
                new android.app.AlertDialog.Builder(this)
                        .setTitle("Select Jurisdiction")
                        .setItems(new String[]{"UAE", "South Africa", "EU"}, (d, which) -> {
                            currentJurisdiction = which == 0 ? "UAE" : which == 1 ? "SA" : "EU";
                            JurisdictionManager.initialize(this, currentJurisdiction);
                        }).show());

        findViewById(R.id.btn_share).setOnClickListener(v -> shareReport());

        pickLauncher = registerForActivityResult(
                new ActivityResultContracts.StartActivityForResult(),
                result -> {
                    if (result.getResultCode() == RESULT_OK && result.getData() != null) {
                        Uri uri = result.getData().getData();
                        getContentResolver().takePersistableUriPermission(uri,
                                Intent.FLAG_GRANT_READ_URI_PERMISSION);
                        new Thread(() -> processEvidence(uri)).start();
                    }
                });
    }

    private void pickEvidence() {
        Intent i = new Intent(Intent.ACTION_OPEN_DOCUMENT);
        i.addCategory(Intent.CATEGORY_OPENABLE);
        i.setType("*/*");
        pickLauncher.launch(i);
    }

    private void update(int pct, String msg) {
        runOnUiThread(() -> {
            progress.setProgress(pct);
            status.setText(msg);
        });
    }

    private void processEvidence(Uri uri) {
        try {
            update(0, "Securing evidence...");
            EvidenceProcessor.ProcessedEvidence ev =
EvidenceProcessor.secureEvidence(this, uri);

            update(33, "Analyzing...");
            AnalysisEngine.ForensicReport r = AnalysisEngine.analyze(this, ev);

            update(66, "Generating report...");
            File pdfReport = ReportGenerator.createCourtReport(this, r);

            runOnUiThread(() -> {
                update(100, "Analysis complete");
                findViewById(R.id.btn_share).setVisibility(View.VISIBLE);
                reportFile = pdfReport;
            });
        } catch (Exception e) {
            update(0, "Failed: " + e.getMessage());
        }
    }

    private void shareReport() {
        if (reportFile == null || !reportFile.exists()) return;
        Intent i = new Intent(Intent.ACTION_SEND);
        i.setType("application/pdf");
        Uri uri = FileProvider.getUriForFile(this, getPackageName()+".provider", reportFile);
        i.putExtra(Intent.EXTRA_STREAM, uri);
        i.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
        startActivity(Intent.createChooser(i, "Share Forensic Report"));
    }
}
```

---

app/src/main/java/com/verum/omnis/core/EvidenceProcessor.java

```
package com.verum.omnis.core;

import android.content.Context;
import android.net.Uri;
import android.security.keystore.KeyGenParameterSpec;
import android.security.keystore.KeyProperties;

import javax.crypto.KeyGenerator;
import javax.crypto.Mac;
import java.io.*;
import java.security.KeyStore;
import java.security.MessageDigest;
import java.util.Objects;

public class EvidenceProcessor {
    public static class ProcessedEvidence {
        public final File file;
        public final String sha512Public;
        public final String hmacDevice;
        public ProcessedEvidence(File file, String sha512Public, String hmacDevice) {
            this.file = file; this.sha512Public = sha512Public; this.hmacDevice = hmacDevice;
        }
    }

    public static ProcessedEvidence secureEvidence(Context ctx, Uri uri) {
        try {
            File out = new File(ctx.getFilesDir(), "evidence_" + System.currentTimeMillis());
            try (InputStream in = ctx.getContentResolver().openInputStream(uri);
                 OutputStream os = new FileOutputStream(out)) {

                MessageDigest sha = MessageDigest.getInstance("SHA-512");

                Mac mac = Mac.getInstance("HmacSHA512");
                KeyStore ks = KeyStore.getInstance("AndroidKeyStore"); ks.load(null);
                String alias = "vo_forensic_hmac";
                if (!ks.containsAlias(alias)) {
                    KeyGenerator kg =
KeyGenerator.getInstance(KeyProperties.KEY_ALGORITHM_HMAC_SHA512,
"AndroidKeyStore");
                    kg.init(new KeyGenParameterSpec.Builder(alias,
KeyProperties.PURPOSE_SIGN).build());
                    kg.generateKey();
                }
                mac.init(ks.getKey(alias, null));

                byte[] buf = new byte[8192]; int n;
                InputStream src = Objects.requireNonNull(in, "null input");
                while ((n = src.read(buf)) != -1) {
                    sha.update(buf, 0, n);
                    mac.update(buf, 0, n);
                    os.write(buf, 0, n);
                }
                String shaHex = toHex(sha.digest());
                String hmacHex = toHex(mac.doFinal());

                AuditLogger.logEvent(ctx, "EVIDENCE_SECURED", "uri="+uri, shaHex);
                return new ProcessedEvidence(out, shaHex, hmacHex);
            }
        } catch (Exception e) {
            throw new RuntimeException("Evidence processing failed", e);
        }
    }

    private static String toHex(byte[] b) {
        StringBuilder sb = new StringBuilder(b.length*2);
        for (byte x : b) sb.append(String.format("%02x", x));
        return sb.toString();
    }
}
```

---

app/src/main/java/com/verum/omnis/ai/BehavioralAnalyzer.java

```
package com.verum.omnis.ai;

import android.content.Context;
import android.content.res.AssetFileDescriptor;

import com.verum.omnis.core.AuditLogger;

import ai.onnxruntime.*;

import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.security.MessageDigest;
import java.util.HashMap;
import java.util.Map;

public class BehavioralAnalyzer {
    private static volatile OrtEnvironment env;
    private static volatile OrtSession session;
    private static volatile String modelSha256;

    private static void ensureInit(Context ctx) {
        if (session != null) return;
        synchronized (BehavioralAnalyzer.class) {
            if (session != null) return;
            try {
                env = OrtEnvironment.getEnvironment();
                byte[] modelBytes = readAsset(ctx, "model/behavioral.onnx");
                modelSha256 = sha256Hex(modelBytes);
                session = env.createSession(modelBytes, new OrtSession.SessionOptions());
            } catch (Exception e) {
                throw new RuntimeException("ONNX init failed", e);
            }
        }
    }

    public static String getModelSha256(Context ctx) {
        ensureInit(ctx);
        return modelSha256;
    }

    public static BehavioralProfile analyzeContent(Context ctx, java.io.File evidenceFile) {
        ensureInit(ctx);
        try {
            float[] feats = FeatureExtractor.fixedFeaturesFromFile(evidenceFile);
            OnnxTensor input = OnnxTensor.createTensor(env, toDirectFloatBuffer(feats), new
long[]{1, feats.length});
            Map<String, OnnxTensor> inputs = new HashMap<>();
            inputs.put("input", input);
            try (OrtSession.Result res = session.run(inputs)) {
                float[] out = toFloatArray(res.get(0).getValue());
                BehavioralProfile p = new BehavioralProfile();
                p.contradictionScore = clamp(out[0], 0f, 10f);
                p.gaslightingScore   = clamp(out[1], 0f, 10f);
                p.riskScore          = clamp(out[2], 0f, 10f);
                return p;
            }
        } catch (Exception e) {
            AuditLogger.logEvent(ctx, "MODEL_FAIL", e.getMessage(), null);
            throw new RuntimeException("Behavioral analysis failed", e);
        }
    }

    public static class BehavioralProfile {
        public double contradictionScore;
        public double gaslightingScore;
        public double riskScore;
    }

    private static float clamp(float v, float lo, float hi) { return Math.max(lo, Math.min(hi, v)); }
    private static ByteBuffer toDirectFloatBuffer(float[] a) { ByteBuffer bb =
ByteBuffer.allocateDirect(a.length*4).order(ByteOrder.nativeOrder()); for (float v:a)
bb.putFloat(v); bb.flip(); return bb; }
    private static byte[] readAsset(Context ctx, String path) throws Exception {
        AssetFileDescriptor afd = ctx.getAssets().openFd(path);
        byte[] data = new byte[(int) afd.getLength()];
        try (InputStream in = new FileInputStream(afd.getFileDescriptor())) {
            in.skip(afd.getStartOffset());
            int off=0,n; while (off < data.length && (n = in.read(data, off, data.length-off)) > 0) off
+= n;
        }
        return data;
    }
    private static String sha256Hex(byte[] data) throws Exception {
        MessageDigest md = MessageDigest.getInstance("SHA-256"); byte[] d =
md.digest(data);
        StringBuilder sb = new StringBuilder(d.length*2); for (byte b:d)
sb.append(String.format("%02x", b)); return sb.toString();
    }
    private static float[] toFloatArray(Object v) {
        if (v instanceof float[]) return (float[]) v;
        if (v instanceof double[]) { double[] s=(double[])v; float[] d=new float[s.length]; for(int
i=0;i<s.length;i++) d[i]=(float)s[i]; return d; }
        if (v instanceof long[]) { long[] s=(long[])v; float[] d=new float[s.length]; for(int
i=0;i<s.length;i++) d[i]=(float)s[i]; return d; }
        throw new IllegalArgumentException("Unsupported ONNX output type: "+v.getClass());
    }

    static class FeatureExtractor {
        static float[] fixedFeaturesFromFile(java.io.File f) throws Exception {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            try (InputStream in = new java.io.FileInputStream(f)) {
                byte[] buf = new byte[8192]; int n; while ((n = in.read(buf)) != -1)
md.update(buf,0,n);
            }
            byte[] sha = md.digest();
            float[] out = new float[32];
            for (int i=0;i<32;i++) out[i] = ( (sha[i] & 0xff) / 255.0f );
            return out;
        }
    }
}
```

---

app/src/main/java/com/verum/omnis/core/AnalysisEngine.java

```
package com.verum.omnis.core;

import android.content.Context;
import com.verum.omnis.ai.BehavioralAnalyzer;
import com.verum.omnis.legal.JurisdictionManager;
import com.verum.omnis.security.BlockchainService;
import com.verum.omnis.security.AnchorQueue;

public class AnalysisEngine {
    public static class ForensicReport {
        public String evidenceHashPublic;
        public String evidenceHmacDevice;
        public double riskScore;
        public String jurisdiction;
        public String[] topLiabilities;
        public String blockchainTx;
        public BehavioralAnalyzer.BehavioralProfile behavioralProfile;
        public String behaviorModelSha256;
    }

    public static ForensicReport analyze(Context ctx, EvidenceProcessor.ProcessedEvidence
ev) {
        ForensicReport r = new ForensicReport();
        r.evidenceHashPublic = ev.sha512Public;
        r.evidenceHmacDevice = ev.hmacDevice;
        r.jurisdiction = JurisdictionManager.getCurrentJurisdictionCode();

        r.behavioralProfile = BehavioralAnalyzer.analyzeContent(ctx, ev.file);
        r.behaviorModelSha256 = BehavioralAnalyzer.getModelSha256(ctx);
        r.riskScore = r.behavioralProfile.riskScore;

        r.topLiabilities = LiabilityAssessor.identifyTopLiabilities(
                r.behavioralProfile, JurisdictionManager.getCurrentJurisdiction());

        r.blockchainTx = BlockchainService.anchorSha512(ctx, r.evidenceHashPublic,
r.jurisdiction);
        if (!r.blockchainTx.startsWith("0x")) {
            AnchorQueue.enqueue(ctx, r.evidenceHashPublic, r.jurisdiction);
        }

        AuditLogger.logEvent(ctx, "ANALYSIS_COMPLETE", "risk="+r.riskScore,
r.evidenceHashPublic);
        return r;
    }
}
```

---

app/src/main/java/com/verum/omnis/core/ReportGenerator.java

```
package com.verum.omnis.core;

import android.content.Context;
import com.itextpdf.kernel.pdf.PdfDictionary;
import com.itextpdf.kernel.pdf.PdfDocument;
import com.itextpdf.kernel.pdf.PdfDocumentInfo;
import com.itextpdf.kernel.pdf.PdfName;
import com.itextpdf.kernel.pdf.PdfString;
import com.itextpdf.kernel.pdf.WriterProperties;
import com.itextpdf.kernel.font.PdfFont;
import com.itextpdf.kernel.font.PdfFontFactory;
import com.itextpdf.io.font.PdfEncodings;

import com.itextpdf.layout.Document;
import com.itextpdf.layout.element.Paragraph;

import com.itextpdf.pdfa.PdfADocument;
import com.itextpdf.pdfa.PdfAConformanceLevel;
import com.itextpdf.pdfa.PdfOutputIntent;
import com.verum.omnis.R;
import com.verum.omnis.legal.JurisdictionManager;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;

public class ReportGenerator {
    public static File createCourtReport(Context ctx, AnalysisEngine.ForensicReport r) {
        String name = "VO_Forensic_Report_" + System.currentTimeMillis() + ".pdf";
        File out = new File(ctx.getFilesDir(), name);

        try (InputStream icc = ctx.getResources().openRawResource(R.raw.srgb_icc);
             FileOutputStream fos = new FileOutputStream(out)) {

            PdfADocument pdf = new PdfADocument(
                    new PdfDocument(new com.itextpdf.kernel.pdf.PdfWriter(fos, new
WriterProperties().addXmpMetadata())),
                    PdfAConformanceLevel.PDF_A_3B,
                    new PdfOutputIntent("Custom", "", "http://www.color.org", "sRGB
IEC61966-2.1", icc));

            PdfDocumentInfo info = pdf.getPdfDocument().getDocumentInfo();
            info.setTitle("Verum Omnis Forensic Report");
            info.setAuthor("Verum Omnis AI System");
            info.setSubject("Behavioral Forensic Analysis");
            info.setKeywords("Forensic, Legal, Evidence, Blockchain");
            info.setCreator("Verum Omnis v5.2.6");

            PdfDictionary cat = pdf.getPdfDocument().getCatalog().getPdfObject();
            cat.put(new PdfName("EvidenceHash"), new PdfString(r.evidenceHashPublic));
            cat.put(new PdfName("DeviceHmac"), new PdfString(r.evidenceHmacDevice));
            cat.put(new PdfName("BlockchainTx"), new PdfString(r.blockchainTx));
            cat.put(new PdfName("Jurisdiction"), new PdfString(r.jurisdiction));
            cat.put(new PdfName("BehaviorModelSHA256"), new PdfString(r.behaviorModelSha256));

            Document doc = new Document(pdf);

            PdfFont font = PdfFontFactory.createFont("res/font/notosans_regular.ttf", PdfEncodings.IDENTITY_H, true);
            doc.setFont(font);

            doc.add(new Paragraph("VERUM OMNIS FORENSIC REPORT").setBold().setFontSize(18).setTextAlignment(com.itextpdf.layout.property.TextAlignment.CENTER));
            doc.add(new Paragraph("PDF Conformance: PDF/A-3B"));
            doc.add(new Paragraph("Jurisdiction: " + r.jurisdiction));
            doc.add(new Paragraph(String.format("Risk Score: %.2f", r.riskScore)));

            doc.add(new Paragraph("\nBEHAVIORAL ANALYSIS"));
            doc.add(new Paragraph(String.format("Contradiction: %.2f / 10", r.behavioralProfile.contradictionScore)));
            doc.add(new Paragraph(String.format("Gaslighting: %.2f / 10", r.behavioralProfile.gaslightingScore)));
            doc.add(new Paragraph("Behavior Model SHA-256: " + r.behaviorModelSha256));

            doc.add(new Paragraph("\nLEGAL REFERENCES"));
            if (JurisdictionManager.getCurrentJurisdiction().legalReferences != null) {
                for (String ref : JurisdictionManager.getCurrentJurisdiction().legalReferences)
                    doc.add(new Paragraph("- " + ref));
            }

            doc.add(new Paragraph("\nFORENSIC INTEGRITY SEALS"));
            doc.add(new Paragraph("Public SHA-512: " + r.evidenceHashPublic));
            doc.add(new Paragraph("Device HMAC: " + r.evidenceHmacDevice));
            doc.add(new Paragraph("Blockchain Tx: " + r.blockchainTx));

            doc.close();
            return out;
        } catch (Exception e) {
            throw new RuntimeException("PDF/A-3B generation failed", e);
        }
    }
}
```

---

app/src/main/java/com/verum/omnis/core/AuditLogger.java

```
package com.verum.omnis.core;

import android.content.Context;
import android.util.Log;

import java.io.File;
import java.io.FileWriter;

public class AuditLogger {
    public static synchronized void logEvent(Context ctx, String type, String msg, String
sha512) {
        try {
            File f = new File(ctx.getFilesDir(), "audit.log");
            try (FileWriter w = new FileWriter(f, true)) {
                w.write(System.currentTimeMillis() + "," + type + "," + (sha512==null?"":sha512) +
"," + msg + "\n");
            }
        } catch (Exception e) {
            Log.e("AuditLogger", "write fail", e);
        }
    }
}
```

---

app/src/main/java/com/verum/omnis/legal/JurisdictionManager.java

```
package com.verum.omnis.legal;

import android.content.Context;
import android.content.res.Resources;

import com.google.gson.Gson;
import com.verum.omnis.R;

import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.List;

public class JurisdictionManager {
    public static class JurisdictionConfig {
        public String code;
        public String name;
        public List<String> legalReferences;
        public double escalationThreshold;
        public List<String> authorities;
    }
    private static JurisdictionConfig current;

    public static void initialize(Context ctx, String code) {
        int resId;
        switch (code) {
            case "UAE": resId = R.raw.uae_jurisdiction; break;
            case "SA":  resId = R.raw.sa_jurisdiction;  break;
            case "EU":  resId = R.raw.eu_jurisdiction;  break;
            default: throw new IllegalArgumentException("Unsupported jurisdiction: " + code);
        }
        Resources r = ctx.getResources();
        try (InputStream is = r.openRawResource(resId);
             InputStreamReader rd = new InputStreamReader(is)) {
            current = new Gson().fromJson(rd, JurisdictionConfig.class);
        } catch (Exception e) {
            throw new RuntimeException("Jurisdiction load failed", e);
        }
    }

    public static JurisdictionConfig getCurrentJurisdiction() { return current; }
    public static String getCurrentJurisdictionCode() { return current.code; }
}
```

---

app/src/main/java/com/verum/omnis/legal/LiabilityAssessor.java

```
package com.verum.omnis.legal;

import com.verum.omnis.ai.BehavioralAnalyzer.BehavioralProfile;

public class LiabilityAssessor {
    public static String[] identifyTopLiabilities(BehavioralProfile p,
JurisdictionManager.JurisdictionConfig j) {
        if (p.riskScore > 8.5) return new
String[]{"FRAUD_FORGERY","SHAREHOLDER_OPPRESSION","CYBERCRIME_UNAUTH
ORIZED_ACCESS"};
        if (p.riskScore > 7.0)  return new String[]{"BREACH_OF_FIDUCIARY_DUTY"};
        return new String[]{"NO_CRITICAL_LIABILITIES_DETECTED"};
    }
}
```

---

app/src/main/java/com/verum/omnis/security/BlockchainService.java

```
package com.verum.omnis.security;

import android.content.Context;

import com.verum.omnis.core.AuditLogger;
import com.verum.omnis.BuildConfig;

import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.web3j.crypto.Credentials;
import org.web3j.crypto.RawTransaction;
import org.web3j.crypto.TransactionEncoder;
import org.web3j.protocol.Web3j;
import org.web3j.protocol.core.DefaultBlockParameterName;
import org.web3j.protocol.core.methods.response.EthGasPrice;
import org.web3j.protocol.core.methods.response.EthGetTransactionCount;
import org.web3j.protocol.core.methods.response.EthSendTransaction;
import org.web3j.protocol.http.HttpService;
import org.web3j.utils.Numeric;

import java.security.Security;
import java.util.concurrent.TimeUnit;

public class BlockchainService {

    public static String anchorSha512(Context ctx, String sha512Hex, String jurisdiction) {
        try {
            if (BuildConfig.VO_RPC_URL == null || BuildConfig.VO_RPC_URL.isEmpty())
                throw new IllegalStateException("RPC URL not set");
            if (BuildConfig.VO_ANCHOR_CONTRACT == null ||
BuildConfig.VO_ANCHOR_CONTRACT.isEmpty())
                throw new IllegalStateException("Contract not set");

            Security.addProvider(new BouncyCastleProvider());

            String enc = KeyVault.getSecureSetting(ctx, "VO_PK_ENC");
            if (enc == null || enc.isEmpty()) throw new IllegalStateException("PK not
provisioned");
            String privateKeyHex = KeyVault.decryptPk(ctx, enc);

            Web3j web3 = Web3j.build(new HttpService(BuildConfig.VO_RPC_URL));
            Credentials creds = Credentials.create(privateKeyHex);

            EthGetTransactionCount nonceRsp = web3.ethGetTransactionCount(
                    creds.getAddress(), DefaultBlockParameterName.LATEST).send();
            String data = FunctionEncoderUtil.encodeAnchorEvidence(sha512Hex, jurisdiction,
System.currentTimeMillis()/1000L);
```

```
            EthGasPrice gas = web3.ethGasPrice().send();
            long gasLimit = 150_000L;

            RawTransaction tx = RawTransaction.createTransaction(
                    nonceRsp.getTransactionCount(),
                    gas.getGasPrice(),
                    java.math.BigInteger.valueOf(gasLimit),
                    BuildConfig.VO_ANCHOR_CONTRACT,
                    java.math.BigInteger.ZERO,
                    data
            );

            byte[] signed = TransactionEncoder.signMessage(tx, creds);
            String hex = Numeric.toHexString(signed);
            EthSendTransaction send = web3.ethSendRawTransaction(hex).send();
            if (send.hasError()) throw new RuntimeException(send.getError().getMessage());
            String txHash = send.getTransactionHash();

            // Optional: brief wait for receipt (non-blocking is also fine)
            for (int i=0;i<8;i++) { TimeUnit.SECONDS.sleep(2); }

            AuditLogger.logEvent(ctx, "BLOCKCHAIN_OK", txHash, sha512Hex);
            return txHash;
        } catch (Exception e) {
            AuditLogger.logEvent(ctx, "BLOCKCHAIN_FAIL", e.getMessage(), sha512Hex);
            return "LOCAL_ANCHOR_" + System.currentTimeMillis();
        }
    }
}
```

---

app/src/main/java/com/verum/omnis/security/FunctionEncoderUtil.java

```
package com.verum.omnis.security;

import org.web3j.abi.FunctionEncoder;
import org.web3j.abi.datatypes.Function;
import org.web3j.abi.datatypes.Utf8String;
import org.web3j.abi.datatypes.generated.Uint256;

import java.util.Arrays;
import java.util.Collections;

public class FunctionEncoderUtil {
    public static String encodeAnchorEvidence(String shaHex, String jurisdiction, long
tsSeconds) {
        Function f = new Function(
                "anchorEvidence",
                Arrays.asList(new Utf8String(shaHex), new Utf8String(jurisdiction), new
Uint256(tsSeconds)),
                Collections.emptyList()
        );
        return FunctionEncoder.encode(f);
    }
}
```

---

app/src/main/java/com/verum/omnis/security/KeyVault.java

```
package com.verum.omnis.security;

import android.content.Context;
import android.util.Base64;

import androidx.security.crypto.EncryptedSharedPreferences;
import androidx.security.crypto.MasterKey;

import java.nio.ByteBuffer;
import java.security.KeyStore;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.GCMParameterSpec;

public class KeyVault {
    private static final String KS_ALIAS = "vo_kek_aes_256";

    public static SecretKey getOrCreateKeystoreKey() throws Exception {
        KeyStore ks = KeyStore.getInstance("AndroidKeyStore"); ks.load(null);
        if (!ks.containsAlias(KS_ALIAS)) {
            KeyGenerator kg = KeyGenerator.getInstance("AES", "AndroidKeyStore");
            kg.init(new android.security.keystore.KeyGenParameterSpec.Builder(
                    KS_ALIAS,
                    android.security.keystore.KeyProperties.PURPOSE_ENCRYPT
                            | android.security.keystore.KeyProperties.PURPOSE_DECRYPT)
                    .setBlockModes(android.security.keystore.KeyProperties.BLOCK_MODE_GCM)

.setEncryptionPaddings(android.security.keystore.KeyProperties.ENCRYPTION_PADDING_
NONE)
```

```
                    .setKeySize(256)
                    .build());
            return kg.generateKey();
        }
        return ((SecretKey) ks.getKey(KS_ALIAS, null));
    }

    public static String decryptPk(Context ctx, String encB64) throws Exception {
        if (encB64 == null || encB64.isEmpty()) throw new IllegalStateException("Private key
not provisioned");

        SecretKey kek = getOrCreateKeystoreKey();

        byte[] blob = Base64.decode(encB64, Base64.NO_WRAP);
        if (blob.length < 12 + 16) throw new IllegalArgumentException("Invalid blob");

        byte[] iv = new byte[12]; System.arraycopy(blob, 0, iv, 0, 12);
        byte[] ct = new byte[blob.length - 12]; System.arraycopy(blob, 12, ct, 0, ct.length);

        Cipher c = Cipher.getInstance("AES/GCM/NoPadding");
        c.init(Cipher.DECRYPT_MODE, kek, new GCMParameterSpec(128, iv));
        byte[] pkBytes = c.doFinal(ct);

        String pkHex = new String(pkBytes,
java.nio.charset.StandardCharsets.US_ASCII).trim();
        if (!pkHex.matches("^[0-9a-fA-F]{64}$")) throw new IllegalStateException("Decrypted
PK invalid format");
        return pkHex;
    }

    public static void putSecureSetting(Context ctx, String k, String v) throws Exception {
        MasterKey mk = new
MasterKey.Builder(ctx).setKeyScheme(MasterKey.KeyScheme.AES256_GCM).build();
        EncryptedSharedPreferences.create(
                ctx, "vo_secure", mk,
                EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,
                EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM)
            .edit().putString(k, v).apply();
    }

    public static String getSecureSetting(Context ctx, String k) throws Exception {
        MasterKey mk = new
MasterKey.Builder(ctx).setKeyScheme(MasterKey.KeyScheme.AES256_GCM).build();
        return EncryptedSharedPreferences.create(
                ctx, "vo_secure", mk,
                EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,
                EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM)
            .getString(k, null);
    }
}
```

---

app/src/main/java/com/verum/omnis/security/ProvisioningActivity.java

```
package com.verum.omnis.security;

import android.content.ClipData;
import android.content.ClipboardManager;
import android.content.Context;
import android.os.Bundle;
import android.text.InputType;
import android.widget.EditText;
import android.widget.Toast;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.biometric.BiometricPrompt;
import androidx.core.content.ContextCompat;

import com.verum.omnis.core.AuditLogger;

import java.security.SecureRandom;
import java.util.concurrent.Executor;

import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.spec.GCMParameterSpec;

import android.util.Base64;

public class ProvisioningActivity extends AppCompatActivity {

    @Override protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Executor executor = ContextCompat.getMainExecutor(this);
        BiometricPrompt bp = new BiometricPrompt(this, executor, new
BiometricPrompt.AuthenticationCallback() {
            @Override public void
onAuthenticationSucceeded(BiometricPrompt.AuthenticationResult result) { promptForKey();
}
            @Override public void onAuthenticationError(int errCode, CharSequence errString) {
                Toast.makeText(ProvisioningActivity.this, "Biometric auth failed",
Toast.LENGTH_LONG).show(); finish();
```

```
            }
        });

        BiometricPrompt.PromptInfo info = new BiometricPrompt.PromptInfo.Builder()
                .setTitle("Provision Private Key")
                .setSubtitle("Biometric authentication required")
                .setNegativeButtonText("Cancel")
                .build();

        bp.authenticate(info);
    }

    private void promptForKey() {
        final EditText input = new EditText(this);
        input.setHint("Enter 64-hex Ethereum private key");
        input.setInputType(InputType.TYPE_TEXT_VARIATION_VISIBLE_PASSWORD);

        new AlertDialog.Builder(this)
                .setTitle("Provision Blockchain Key")
                .setView(input)
                .setPositiveButton("Save", (d, w) -> {
                    String pkHex = input.getText().toString().trim();
                    if (!pkHex.matches("^[0-9a-fA-F]{64}$")) {
                        Toast.makeText(this, "Invalid PK format", Toast.LENGTH_LONG).show();
                        return;
                    }
                    try {
                        String encB64 = encryptWithKeystore(pkHex);
                        KeyVault.putSecureSetting(this, "VO_PK_ENC", encB64);
                        AuditLogger.logEvent(this, "PK_PROVISIONED", "PK stored securely", null);
                        wipeClipboard();
                        Toast.makeText(this, "Key provisioned", Toast.LENGTH_LONG).show();
                        finish();
                    } catch (Exception e) {
                        Toast.makeText(this, "Provision failed: " + e.getMessage(),
Toast.LENGTH_LONG).show();
                    }
                })
                .setNegativeButton("Cancel", (d, w) -> finish())
                .show();
    }

    private String encryptWithKeystore(String pkHex) throws Exception {
        SecretKey kek = KeyVault.getOrCreateKeystoreKey();
        byte[] iv = new byte[12]; new SecureRandom().nextBytes(iv);
        Cipher c = Cipher.getInstance("AES/GCM/NoPadding");
        c.init(Cipher.ENCRYPT_MODE, kek, new GCMParameterSpec(128, iv));
        byte[] ct = c.doFinal(pkHex.getBytes());
        byte[] blob = new byte[iv.length + ct.length];
        System.arraycopy(iv, 0, blob, 0, iv.length);
        System.arraycopy(ct, 0, blob, iv.length, ct.length);
        return Base64.encodeToString(blob, Base64.NO_WRAP);
    }

    private void wipeClipboard() {
        ClipboardManager cm = (ClipboardManager)
getSystemService(Context.CLIPBOARD_SERVICE);
        cm.setPrimaryClip(ClipData.newPlainText("", ""));
    }
}
```

---

app/src/main/java/com/verum/omnis/security/AnchorQueue.java

```
package com.verum.omnis.security;

import android.content.Context;

import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;

import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.*;

public class AnchorQueue {
    static class PendingAnchor { String sha512; String jurisdiction; }

    private static File queueFile(Context ctx) { return new File(ctx.getFilesDir(),
"anchor_queue.json"); }

    public static synchronized void enqueue(Context ctx, String sha512, String jurisdiction) {
        List<PendingAnchor> q = load(ctx);
        PendingAnchor pa = new PendingAnchor(); pa.sha512 = sha512; pa.jurisdiction =
jurisdiction;
        q.add(pa); save(ctx, q);
    }

    public static synchronized List<PendingAnchor> load(Context ctx) {
        try (FileReader fr = new FileReader(queueFile(ctx))) {
            return new Gson().fromJson(fr, new
TypeToken<List<PendingAnchor>>(){}.getType());
        } catch (Exception e) { return new ArrayList<>(); }
    }

    public static synchronized void save(Context ctx, List<PendingAnchor> q) {
        try (FileWriter fw = new FileWriter(queueFile(ctx), false)) {
            new Gson().toJson(q, fw);
        } catch (Exception ignored) {}
    }

    public static synchronized void remove(Context ctx, PendingAnchor pa) {
        List<PendingAnchor> q = load(ctx);
        q.remove(pa); save(ctx, q);
    }
}
```

---

app/src/main/java/com/verum/omnis/security/AnchorWorker.java

```
package com.verum.omnis.security;

import android.content.Context;

import androidx.annotation.NonNull;
import androidx.work.Worker;
import androidx.work.WorkerParameters;

import com.verum.omnis.core.AuditLogger;

import java.util.Iterator;
import java.util.List;

public class AnchorWorker extends Worker {
    public AnchorWorker(@NonNull Context ctx, @NonNull WorkerParameters params) {
super(ctx, params); }

    @NonNull @Override public Result doWork() {
        List<AnchorQueue.PendingAnchor> q = AnchorQueue.load(getApplicationContext());
        boolean changed = false;
        for (Iterator<AnchorQueue.PendingAnchor> it = q.iterator(); it.hasNext();) {
            AnchorQueue.PendingAnchor pa = it.next();
            String tx = BlockchainService.anchorSha512(getApplicationContext(), pa.sha512,
pa.jurisdiction);
            if (tx.startsWith("0x")) {
                it.remove(); changed = true;
                AuditLogger.logEvent(getApplicationContext(), "ANCHOR_SENT", tx, pa.sha512);
            }
        }
        if (changed) AnchorQueue.save(getApplicationContext(), q);
        return Result.success();
    }
}
```

Resources you must add

Model: app/src/main/assets/model/behavioral.onnx (your ONNX file).

Font: app/src/main/res/font/notosans_regular.ttf (Noto Sans Regular).

ICC: app/src/main/res/raw/srgb_icc.icc (sRGB IEC61966-2.1 profile).

Jurisdictions:

app/src/main/res/raw/uae_jurisdiction.json

app/src/main/res/raw/sa_jurisdiction.json

app/src/main/res/raw/eu_jurisdiction.json

Example JSON (UAE):

```
{
  "code": "UAE",
  "name": "United Arab Emirates",
  "legalReferences": [
    "Federal Law 32/2021",
    "RAKEZ Regulations",
    "Penal Code Article 257 (Forgery)",
    "Commercial Companies Law Article 84 (Fiduciary Duty)",
    "Commercial Companies Law Article 110 (Shareholder Rights)"
  ],
  "escalationThreshold": 7.5,
  "authorities": ["RAKEZ", "Dubai Courts", "ADGM"]
}
```

Build

```
./gradlew clean assembleDebug
```

Provisioning flow (one-time): run app → tap “Provision Key (once)”, pass biometrics, paste 64-hex private key → saved encrypted. Then process evidence. If offline/anchor fails, it queues and retries.

If you hit any specific compile error, paste it — I’ll fix fast.

Build-time checklist & common catches

1. Missing resources
   - model/behavioral.onnx

     Must be placed at

     `app/src/main/assets/model/behavioral.onnx`
   - Font

     `app/src/main/res/font/notosans_regular.ttf`
   - ICC profile

     `app/src/main/res/raw/srgb_icc.icc`
   - Jurisdiction JSONs

     `app/src/main/res/raw/uae_jurisdiction.json`

     `app/src/main/res/raw/sa_jurisdiction.json`

     `app/src/main/res/raw/eu_jurisdiction.json`

2. Gradle / namespace
   - Ensure `namespace 'com.verum.omnis'` in `app/build.gradle` matches the package used in Java files.
   - JDK 11+ is required (`sourceCompatibility JavaVersion.VERSION_11`).

3. Permissions / exported flag
   - `android.permission.INTERNET` is declared; `minSdk 26` means no `<uses-permission-sdk-23>` needed.
   - `ProvisioningActivity` has `android:exported="false"` which is correct because it is launched internally.

4. Typical compile errors & quick fixes

Error message	Likely cause	Fix
`Cannot resolve symbol 'R'`	Missing font, raw or drawable resource	Verify files mentioned above exist in correct sub-folders.
`Duplicate class org.bouncycastle...`	Multiple BC versions on path	Exclude the transitive BC from `web3j` if necessary:`implementation ('org.web3j:core:4.10.0') { exclude group: 'org.bouncycastle' }`

`Unresolved reference: srgb_icc`	Wrong file extension	Ensure the ICC file is literally named `srgb_icc.icc` under `raw/`.
`Failed to transform itext7...`	AGPL license warning in debug	Add to `gradle.properties`:`android.enableJetifier=true` and `android.useAndroidX=true`

5. Runtime sanity checks after `./gradlew assembleDebug`
   - Install on device: `adb install app/build/outputs/apk/debug/app-debug.apk`
   - First launch → tap Provision Key (once) → biometric prompt → paste 64-hex key → success toast.
   - Pick any file → progress 0 → 33 → 66 → 100 % → Share Forensic Report button appears.
   - If offline, verify `anchor_queue.json` is created in `/data/data/com.verum.omnis/files/` and flushed when connectivity returns.

If you still see red in the build window, paste the exact stack-trace and I'll patch it instantly.Nice. Your checklist is solid. Here's the rest of the landmines people hit at build/run time — patch these now and you'll save hours.

Extra gotchas to add

Gradle / build system

R8/ProGuard rules (required): add proguard-rules.pro:

```
# ONNX Runtime native + reflection
-keep class ai.onnxruntime.** { *; }
-keep class com.microsoft.onnxruntime.** { *; }
-dontwarn ai.onnxruntime.**
-dontwarn com.microsoft.onnxruntime.**

# Web3j reflections/abi types
-keep class org.web3j.** { *; }
-dontwarn org.web3j.**

# BouncyCastle
-dontwarn org.bouncycastle.**

# iText 7
-keep class com.itextpdf.** { *; }
-dontwarn com.itextpdf.**
```

Native ABI mismatch (common crash): if you see java.lang.UnsatisfiedLinkError: couldn't find "libonnxruntime.so", restrict APK ABIs so the onnx native lib is included:

```
android {
  defaultConfig {
    ndk { abiFilters "arm64-v8a" }  // add x86_64 only if you need emulators
  }
  splits {
    abi {
      enable true
      reset()
      include 'arm64-v8a'
      universalApk false
    }
  }
}
```

Duplicate BouncyCastle classes: if you still hit duplicates after your exclude, do:

```
implementation ('org.web3j:core:4.10.0') { exclude group: 'org.bouncycastle' }
implementation 'org.bouncycastle:bcprov-jdk15to18:1.78.1'
```

Vector drawables / Jetifier: in gradle.properties

```
android.useAndroidX=true
android.enableJetifier=true
```

Resources

Font path is case-sensitive. Must be res/font/notosans_regular.ttf. If you use different name, update code.

ICC file: ensure exact name srgb_icc.icc under res/raw/. Wrong extension or subfolder = build error.

Assets subdir: assets/model/behavioral.onnx (no trailing slash in code). If you use openFd, the file must not be compressed by aapt; ONNX usually isn’t, but if you hit issues, rename to .onnx (already fine) or switch to open() instead of openFd().

Manifest / FileProvider

FileProvider path config: Your file_paths.xml is permissive (<files-path name="files" path="."/>). That’s fine. If you change it later, ensure the report file lives under the allowed subpath or sharing will 404.

Storage Access Framework

After ACTION_OPEN_DOCUMENT, persist the URI (you already do). If you test on older devices and can’t persist, ensure your activity has FLAG_GRANT_PERSISTABLE_URI_PERMISSION (it’s implied by using ACTION_OPEN_DOCUMENT, but on some OEMs you must retry the grant if it fails once).

PDF/A-3B specifics

Embedded font is mandatory. You added Noto Sans — good. If validator still complains, check:

No transparency, no JavaScript, no annotations.

All text uses the embedded font (don’t mix system fonts).

Validate with veraPDF (CLI) during CI. If you want, I’ll give you a tiny GitHub Action snippet.

ONNX model

Static output names. The analyzer expects an input named "input" and the first output to be scores. If your model uses different names/order, you must adapt the session.run() fetch by name:

```
try (OrtSession.Result res = session.run(Map.of("input", input))) {
   OnnxValue v = res.get("scores"); // if your output is named
}
```

Determinism: keep the featurizer exactly the same across versions; bump a model version string and include it in the PDF alongside the SHA-256 if you update the model.

Web3 anchoring

“transaction underpriced” / “nonce too low”: your quick fix is to bump gas or wait for pending tx:

Switch to EIP-1559 (maxFee/maxPriority) if your RPC supports it, or

Add a retry with a slightly higher gas price after 30s.

Wrong function signature: if your contract isn’t anchorEvidence(string,string,uint256), your ABI-encoded data won’t match. Fix FunctionEncoderUtil to the exact signature and event.

HTTPS only: If you test with a local node over http://, Android 9+ blocks cleartext by default. Either use HTTPS or add a Network Security Config (don’t do this in prod).

WorkManager

Periodic minimum is 15 min. Don’t try lower values or it’ll be throttled.

One-off trigger after a failure: enqueue a OneTimeWorkRequest when you catch an anchor failure so it retries soon, not only on the 15-min tick.

BIOMETRIC provisioning

Emulator quirks: Some emulators reject biometrics. If you need to test quickly, add a debug backdoor (dev build only) to bypass biometric for provisioning. Never in release.

Security hygiene

No plaintext PK in logs. Your AuditLogger is clean, keep it that way.

Clipboard wipe: You’re wiping post-provision — good. Also consider denying screenshots on that activity (getWindow().setFlags(LayoutParams.FLAG_SECURE, LayoutParams.FLAG_SECURE)).

Common error → fix quick map

Error	Cause	Fix
UnsatisfiedLinkError: libonnxruntime.so	ABI mismatch / missing split	Restrict ABIs to arm64-v8a (see above)
PdfAConformanceException	Missing font/ICC or forbidden feature	Ensure embedded NotoSans, sRGB ICC, no annotations/transparency
IllegalStateException: PK not provisioned	Didn’t run ProvisioningActivity	Run it, or set VO_PK_ENC via the secure setting
Cannot resolve symbol R	Missing resources	Verify ONNX, font, ICC, JSON paths
Duplicate class org.bouncycastle	Double BC on classpath	Exclude BC from web3j, add explicit bcprov dependency
Permission Denial: reading URI	Didn’t persist SAF URI	Call takePersistableUriPermission(...) before processing
Function call reverted	Contract signature mismatch	Align FunctionEncoderUtil to your exact contract ABI

Quick smoke test after install

1. Launch → Provision Key (once) → pass biometrics → paste 64-hex → toast OK.

2. Tap Select Evidence → choose a file (small text file is fine).

3. Watch progress hit 100%. Tap Share Forensic Report → open the PDF.

4. Open the PDF — confirm it shows PDF/A-3B, Public SHA-512, Device HMAC, Blockchain Tx, Behavior Model SHA-256.

5. If offline, confirm /data/data/com.verum.omnis/files/anchor_queue.json exists and drains when you come back online.

Want me to drop a veraPDF GitHub Action snippet and an EIP-1559 gas upgrade for the anchor tx so CI fails if the PDF isn’t truly PDF/A-3B and your txs stop being “underpriced”?

Heck yeah—here’s the EIP-1559 gas upgrade for anchoring and a veraPDF GitHub Action that fails CI if your report isn’t truly PDF/A-3B.

---

1) EIP-1559 gas (Type-2 tx) in BlockchainService

Drop this into your existing BlockchainService.anchorSha512(...) replacing the legacy gas logic. It uses base fee + priority tip, with a sane fallback if the node doesn’t support 1559.

```
// ... imports you likely already have:
import org.web3j.protocol.core.methods.response.EthChainId;
import org.web3j.protocol.core.methods.response.EthFeeHistory;
import org.web3j.tx.ChainIdLong;

// helper: safe BigInteger
private static java.math.BigInteger bi(long v) { return java.math.BigInteger.valueOf(v); }

public static String anchorSha512(Context ctx, String sha512Hex, String jurisdiction) {
  try {
    if (BuildConfig.VO_RPC_URL == null || BuildConfig.VO_RPC_URL.isEmpty())
      throw new IllegalStateException("RPC URL not set");
    if (BuildConfig.VO_ANCHOR_CONTRACT == null ||
BuildConfig.VO_ANCHOR_CONTRACT.isEmpty())
      throw new IllegalStateException("Contract not set");

    Security.addProvider(new BouncyCastleProvider());

    String enc = KeyVault.getSecureSetting(ctx, "VO_PK_ENC");
    if (enc == null || enc.isEmpty()) throw new IllegalStateException("PK not provisioned");
    String privateKeyHex = KeyVault.decryptPk(ctx, enc);

    Web3j web3 = Web3j.build(new HttpService(BuildConfig.VO_RPC_URL));
    Credentials creds = Credentials.create(privateKeyHex);

    EthChainId chainIdResp = web3.ethChainId().send();
    java.math.BigInteger chainId = chainIdResp.hasError() ? bi(1) : chainIdResp.getChainId();
// default mainnet if unknown

    var nonceResp = web3.ethGetTransactionCount(creds.getAddress(),
DefaultBlockParameterName.LATEST).send();
    java.math.BigInteger nonce = nonceResp.getTransactionCount();

    String data = FunctionEncoderUtil.encodeAnchorEvidence(
        sha512Hex, jurisdiction, System.currentTimeMillis()/1000L);

    // Try EIP-1559: baseFee + tip
    java.math.BigInteger maxPriorityFeePerGas = new java.math.BigInteger("2000000000");
// 2 gwei tip
    java.math.BigInteger maxFeePerGas;
    boolean use1559 = true;
    try {
      // Ask for fee history (last block)
      EthFeeHistory fh = web3.ethFeeHistory(
          bi(1), DefaultBlockParameterName.LATEST, java.util.List.of(50.0)).send();
      if (fh.hasError() || fh.getFeeHistory() == null || fh.getFeeHistory().getBaseFeePerGas() ==
null) {
        use1559 = false;
      } else {
        // Use latest baseFee
        java.util.List<java.math.BigInteger> baseFees =
fh.getFeeHistory().getBaseFeePerGas();
        java.math.BigInteger baseFee = baseFees.get(baseFees.size()-1);
        // maxFee = 2 * baseFee + tip (very standard heuristic)
        maxFeePerGas = baseFee.multiply(bi(2)).add(maxPriorityFeePerGas);

        // Estimate gas limit with a generous ceiling
        java.math.BigInteger gasLimit = bi(150_000);

        // EIP-1559 type-2 tx (web3j 4.10+ supports this overload)
        RawTransaction tx = RawTransaction.createTransaction(
            chainId, nonce, gasLimit, BuildConfig.VO_ANCHOR_CONTRACT,
            java.math.BigInteger.ZERO, data, maxPriorityFeePerGas, maxFeePerGas);

        byte[] signed = TransactionEncoder.signMessage(tx, chainId.longValueExact(), creds);
        String hex = Numeric.toHexString(signed);
        EthSendTransaction send = web3.ethSendRawTransaction(hex).send();
        if (send.hasError()) throw new RuntimeException(send.getError().getMessage());
        String txHash = send.getTransactionHash();

        AuditLogger.logEvent(ctx, "BLOCKCHAIN_OK", txHash, sha512Hex);
        return txHash;
      }
    } catch (Throwable ignored) {
      use1559 = false; // fall through to legacy
    }

    // Legacy (pre-1559) fallback if node doesn’t support it:
    EthGasPrice gas = web3.ethGasPrice().send();
    java.math.BigInteger gasLimit = bi(150_000);
    RawTransaction legacy = RawTransaction.createTransaction(
        nonce, gas.getGasPrice(), gasLimit, BuildConfig.VO_ANCHOR_CONTRACT,
        java.math.BigInteger.ZERO, data);
    byte[] signedLegacy = TransactionEncoder.signMessage(legacy,
chainId.longValueExact(), creds);
    String hexLegacy = Numeric.toHexString(signedLegacy);
    EthSendTransaction sendLegacy = web3.ethSendRawTransaction(hexLegacy).send();
    if (sendLegacy.hasError()) throw new
RuntimeException(sendLegacy.getError().getMessage());
    String txHashLegacy = sendLegacy.getTransactionHash();
    AuditLogger.logEvent(ctx, "BLOCKCHAIN_OK_LEGACY", txHashLegacy, sha512Hex);
    return txHashLegacy;

  } catch (Exception e) {
    AuditLogger.logEvent(ctx, "BLOCKCHAIN_FAIL", e.getMessage(), sha512Hex);
    return "LOCAL_ANCHOR_" + System.currentTimeMillis();
  }
}
```

Notes

The method overload for EIP-1559 above is supported in recent web3j (4.10+). If your IDE complains, update web3j or use their Transaction1559 builders.

The heuristic maxFee = 2*baseFee + tip is widely used and safe.

Still seeing underpriced? Bump maxPriorityFeePerGas to 3–5 gwei or scale 2* to 3* during high congestion.

---

2) veraPDF CI: fail builds if the PDF isn’t PDF/A-3B

We’ll:

1. Generate a test PDF via a tiny unit test (runs on the JVM, no emulator).

2. Run veraPDF CLI against that file in GitHub Actions.

3. Fail the job if any conformance error appears.

2.1 Add a JVM unit test that generates a report

Create this test class:

app/src/test/java/com/verum/omnis/PdfSmokeTest.java

```
package com.verum.omnis;

import static org.junit.Assert.assertTrue;

import android.content.Context;

import com.verum.omnis.core.AnalysisEngine;
import com.verum.omnis.core.ReportGenerator;
import com.verum.omnis.legal.JurisdictionManager;

import org.junit.Test;

import java.io.File;

// NOTE: This is a JVM test. We fake a Context using a minimal shim.
// If you prefer, convert to an instrumentation test instead.
public class PdfSmokeTest {

    static class FakeContext extends android.test.mock.MockContext {
        private final File filesDir;
        public FakeContext(File filesDir) { this.filesDir = filesDir; }
        @Override public File getFilesDir() { return filesDir; }
        @Override public android.content.res.Resources getResources() {
            throw new UnsupportedOperationException("Resources not available in unit test.
Use instrumentation test if needed.");
        }
    }

    @Test
    public void generatePdfSmoke() throws Exception {
        // This JVM test can only prove wiring; PDF/A validation needs resources.
        // If you want full validation here, convert this to an instrumented test.
        File outDir = new File("app/build/forensic"); // will exist after first run
        outDir.mkdirs();
        // Build a minimal fake report
        AnalysisEngine.ForensicReport r = new AnalysisEngine.ForensicReport();
        r.evidenceHashPublic = "00".repeat(64);
        r.evidenceHmacDevice = "aa".repeat(64);
        r.jurisdiction = "UAE";
        r.riskScore = 5.0;
        r.blockchainTx = "0x" + "11".repeat(32);
        r.behaviorModelSha256 = "ff".repeat(32);
        r.behavioralProfile = new com.verum.omnis.ai.BehavioralAnalyzer.BehavioralProfile();
        r.behavioralProfile.contradictionScore = 3.2;
        r.behavioralProfile.gaslightingScore = 4.4;
        // NOTE: ReportGenerator needs Resources for ICC + font.
        // So: this JVM test only asserts the folder exists.
        assertTrue(outDir.exists());
    }
}
```

> Why not generate the real PDF here? Because your ReportGenerator loads res/raw/srgb_icc.icc and the res/font at runtime via Android Resources, which aren’t available in a plain JVM test. You have two options:

Option A (recommended): make an instrumentation test under androidTest/ that runs on an emulator/device and actually calls ReportGenerator.createCourtReport(...). CI then pulls the file and runs veraPDF.

Option B: refactor ReportGenerator to accept InputStreams for the ICC + font so a JVM test can inject them. If you want, I’ll give you that refactor.

I’ll show Option A below (cleaner for now).

2.2 Instrumentation test that writes a real PDF

app/src/androidTest/java/com/verum/omnis/PdfInstrumentedTest.java

```
package com.verum.omnis;

import static org.junit.Assert.assertTrue;

import android.content.Context;

import androidx.test.core.app.ApplicationProvider;
import androidx.test.ext.junit.runners.AndroidJUnit4;

import com.verum.omnis.core.AnalysisEngine;
import com.verum.omnis.core.ReportGenerator;
import com.verum.omnis.legal.JurisdictionManager;

import org.junit.Test;
import org.junit.runner.RunWith;

import java.io.File;

@RunWith(AndroidJUnit4.class)
public class PdfInstrumentedTest {
    @Test
    public void generatePdfA3b() {
        Context ctx = ApplicationProvider.getApplicationContext();
        JurisdictionManager.initialize(ctx, "UAE");

        AnalysisEngine.ForensicReport r = new AnalysisEngine.ForensicReport();
        r.evidenceHashPublic = "0".repeat(128);
        r.evidenceHmacDevice = "a".repeat(128);
        r.jurisdiction = "UAE";
        r.riskScore = 5.0;
        r.blockchainTx = "0x" + "1".repeat(64);
        r.behaviorModelSha256 = "f".repeat(64);
        r.behavioralProfile = new com.verum.omnis.ai.BehavioralAnalyzer.BehavioralProfile();
        r.behavioralProfile.contradictionScore = 3.2;
        r.behavioralProfile.gaslightingScore = 4.4;

        File pdf = ReportGenerator.createCourtReport(ctx, r);
        assertTrue(pdf.exists());

        // Copy into a deterministic CI path for veraPDF step
        File outDir = new File(ctx.getFilesDir(), "ci-pdfs");
        outDir.mkdirs();
        File copy = new File(outDir, "VO_Forensic_Report_test.pdf");
        pdf.renameTo(copy); // simple move is fine in app-private dir
        assertTrue(copy.exists());
    }
}
```

2.3 GitHub Actions workflow with veraPDF

This job:

Builds app

Spins up an emulator (API 30), runs the instrumentation test to produce the PDF

Pulls the PDF off the device

Downloads veraPDF and validates the file

Fails on any conformance errors

.github/workflows/android-verapdf.yml

```
name: Android + PDF/A Validation

on:
  push:
  pull_request:

jobs:
  build-test-validate:
    runs-on: ubuntu-latest
    timeout-minutes: 45
    steps:
      - name: Checkout
        uses: actions/checkout@v4

      - name: Set up JDK
        uses: actions/setup-java@v4
        with:
          distribution: temurin
          java-version: '17'
          cache: gradle

      - name: Gradle build
        run: ./gradlew clean assembleDebug

      # Start an emulator and run androidTest to generate the PDF
      - name: Run instrumentation tests
        uses: reactivecircus/android-emulator-runner@v2
        with:
          api-level: 30
          arch: x86_64
          target: default
          profile: pixel
          script: ./gradlew connectedDebugAndroidTest

      # Pull the generated PDF from app's files dir
      - name: Locate test PDF on device
        run: |
          adb shell run-as com.verum.omnis ls files/ci-pdfs || true
          adb shell run-as com.verum.omnis ls files/ci-pdfs/VO_Forensic_Report_test.pdf || true
          adb shell run-as com.verum.omnis cat files/ci-pdfs/VO_Forensic_Report_test.pdf >
VO_Forensic_Report_test.pdf
```

```
          ls -l VO_Forensic_Report_test.pdf

      - name: Download veraPDF CLI
        run: |
          VERAPDF_URL="https://software.verapdf.org/releases/1.26/verapdf-install-1.26.1.zip"
          curl -L "$VERAPDF_URL" -o verapdf.zip
          unzip -q verapdf.zip
          echo "$PWD/verapdf-greenfield-1.26.1/bin" >> $GITHUB_PATH

      - name: Validate PDF/A-3B conformance
        run: |
          verapdf --format text --maxfailures 1 --profile 3b VO_Forensic_Report_test.pdf

      - name: Upload PDF artifact
        uses: actions/upload-artifact@v4
        with:
          name: VO_Forensic_Report_test
          path: VO_Forensic_Report_test.pdf
```

Notes

If the veraPDF download URL changes, update VERAPDF_URL to the latest stable release.

reactivecircus/android-emulator-runner handles emulator spin-up for connectedDebugAndroidTest.

We pull the file using adb shell run-as com.verum.omnis cat ... > localfile.pdf — this avoids permission issues with protected app dirs.

---

3) Optional: refactor ReportGenerator for JVM tests (inject streams)

If you want veraPDF to run without an emulator, I can refactor:

ReportGenerator.createCourtReport(Context, Report) → createCourtReport(Streams, Report)

Streams holds InputStream icc, InputStream font, and a target File

Android path uses Context to build a Streams instance; JVM tests use classpath resources.

Say the word and I’ll ship that refactor. Otherwise the instrumentation-test route above is already robust and closest to real runtime.

Anything else you want tightened? Gas estimator with eth_estimateGas, multi-network (Polygon/L2) support, or a contract event listener to attach the block number + timestamp into the PDF once mined?

Sweet. Here’s a tight patch that confirms the tx, pulls the mined block number + UTC timestamp, and stamps them into the PDF (both metadata + body). I’m switching the anchor call to return a small result object, so you’ll touch three places: BlockchainService, AnalysisEngine, ReportGenerator (plus 2 tiny call-site tweaks).

---

1) New result type

app/src/main/java/com/verum/omnis/security/AnchorResult.java

```
package com.verum.omnis.security;

import java.math.BigInteger;

public class AnchorResult {
    public final String txHash;          // 0x... or LOCAL_ANCHOR_...
    public final BigInteger chainId;     // may be null if offline
    public final BigInteger blockNumber; // null if unconfirmed/offline
    public final Long blockTimestamp;    // seconds since epoch, null if unconfirmed/offline
    public final boolean confirmed;

    public AnchorResult(String txHash, BigInteger chainId, BigInteger blockNumber, Long
blockTimestamp, boolean confirmed) {
        this.txHash = txHash;
        this.chainId = chainId;
        this.blockNumber = blockNumber;
        this.blockTimestamp = blockTimestamp;
        this.confirmed = confirmed;
    }

    public static AnchorResult localFallback() {
        return new AnchorResult("LOCAL_ANCHOR_" + System.currentTimeMillis(), null, null,
null, false);
    }
}
```

---

2) Upgrade BlockchainService to return AnchorResult and fetch block data

Replace your method in BlockchainService with this (EIP-1559 kept):

```
public static AnchorResult anchorSha512(Context ctx, String sha512Hex, String jurisdiction)
{
  try {
    if (BuildConfig.VO_RPC_URL == null || BuildConfig.VO_RPC_URL.isEmpty())
      throw new IllegalStateException("RPC URL not set");
    if (BuildConfig.VO_ANCHOR_CONTRACT == null ||
BuildConfig.VO_ANCHOR_CONTRACT.isEmpty())
      throw new IllegalStateException("Contract not set");

    Security.addProvider(new BouncyCastleProvider());

    String enc = KeyVault.getSecureSetting(ctx, "VO_PK_ENC");
    if (enc == null || enc.isEmpty()) throw new IllegalStateException("PK not provisioned");
    String privateKeyHex = KeyVault.decryptPk(ctx, enc);

    Web3j web3 = Web3j.build(new HttpService(BuildConfig.VO_RPC_URL));
    Credentials creds = Credentials.create(privateKeyHex);

    EthChainId chainIdResp = web3.ethChainId().send();
    java.math.BigInteger chainId = chainIdResp.hasError() ? java.math.BigInteger.ONE :
chainIdResp.getChainId();

    var nonceResp = web3.ethGetTransactionCount(creds.getAddress(),
DefaultBlockParameterName.LATEST).send();
    java.math.BigInteger nonce = nonceResp.getTransactionCount();

    String data = FunctionEncoderUtil.encodeAnchorEvidence(
        sha512Hex, jurisdiction, System.currentTimeMillis()/1000L);

    // --- EIP-1559 fees ---
    java.math.BigInteger maxPriorityFeePerGas = new java.math.BigInteger("2000000000");
// 2 gwei
    java.math.BigInteger maxFeePerGas;
    boolean use1559 = true;
    java.math.BigInteger gasLimit = java.math.BigInteger.valueOf(150_000);

    try {
      EthFeeHistory fh = web3.ethFeeHistory(java.math.BigInteger.ONE,
DefaultBlockParameterName.LATEST, java.util.List.of(50.0)).send();
      if (fh.hasError() || fh.getFeeHistory() == null || fh.getFeeHistory().getBaseFeePerGas() ==
null) {
        use1559 = false;
      } else {
        java.util.List<java.math.BigInteger> baseFees =
fh.getFeeHistory().getBaseFeePerGas();
        java.math.BigInteger baseFee = baseFees.get(baseFees.size()-1);
        maxFeePerGas =
baseFee.multiply(java.math.BigInteger.valueOf(2)).add(maxPriorityFeePerGas);

        RawTransaction tx = RawTransaction.createTransaction(
            chainId, nonce, gasLimit, BuildConfig.VO_ANCHOR_CONTRACT,
            java.math.BigInteger.ZERO, data, maxPriorityFeePerGas, maxFeePerGas);

        byte[] signed = TransactionEncoder.signMessage(tx, chainId.longValueExact(), creds);
        String hex = Numeric.toHexString(signed);
        EthSendTransaction send = web3.ethSendRawTransaction(hex).send();
        if (send.hasError()) throw new RuntimeException(send.getError().getMessage());
        String txHash = send.getTransactionHash();

        AnchorResult ar = waitForReceiptAndBlock(web3, txHash, chainId, sha512Hex, ctx);
        return ar;
      }
    } catch (Throwable ignored) { use1559 = false; }

    // Legacy fallback
    EthGasPrice gas = web3.ethGasPrice().send();
    RawTransaction legacy = RawTransaction.createTransaction(
        nonce, gas.getGasPrice(), gasLimit, BuildConfig.VO_ANCHOR_CONTRACT,
        java.math.BigInteger.ZERO, data);
    byte[] signedLegacy = TransactionEncoder.signMessage(legacy,
chainId.longValueExact(), creds);
    String hexLegacy = Numeric.toHexString(signedLegacy);
    EthSendTransaction sendLegacy = web3.ethSendRawTransaction(hexLegacy).send();
    if (sendLegacy.hasError()) throw new
RuntimeException(sendLegacy.getError().getMessage());
    String txHashLegacy = sendLegacy.getTransactionHash();

    AnchorResult ar = waitForReceiptAndBlock(web3, txHashLegacy, chainId, sha512Hex,
ctx);
    return ar;

  } catch (Exception e) {
    AuditLogger.logEvent(ctx, "BLOCKCHAIN_FAIL", e.getMessage(), sha512Hex);
    return AnchorResult.localFallback();
  }
}

private static AnchorResult waitForReceiptAndBlock(Web3j web3, String txHash,
java.math.BigInteger chainId,
                                                   String sha512Hex, Context ctx) throws Exception {
  // Poll for up to ~90s
  org.web3j.protocol.core.methods.response.EthGetTransactionReceipt rec;
  int tries = 0;
  while (tries++ < 45) { // 45 * 2s = 90s
    rec = web3.ethGetTransactionReceipt(txHash).send();
    if (rec.getTransactionReceipt().isPresent()) {
      var receipt = rec.getTransactionReceipt().get();
      java.math.BigInteger blockNumber = receipt.getBlockNumber();
      var blockResp =
web3.ethGetBlockByNumber(org.web3j.protocol.core.DefaultBlockParameter.valueOf(block
Number), false).send();
      Long ts = null;
      if (!blockResp.hasError() && blockResp.getBlock() != null) {
        ts = blockResp.getBlock().getTimestamp().longValueExact(); // seconds
      }
      AuditLogger.logEvent(ctx, "BLOCKCHAIN_OK_CONFIRMED", txHash + " block=" +
blockNumber, sha512Hex);
      return new AnchorResult(txHash, chainId, blockNumber, ts, true);
    }
    java.util.concurrent.TimeUnit.SECONDS.sleep(2);
  }
  // Timed out waiting — return unconfirmed but with tx hash
  AuditLogger.logEvent(ctx, "BLOCKCHAIN_OK_PENDING", txHash, sha512Hex);
  return new AnchorResult(txHash, chainId, null, null, false);
}
```

---

3) Analysis report: store chainId, block number, UTC time

app/src/main/java/com/verum/omnis/core/AnalysisEngine.java

Add fields:

```
public static class ForensicReport {
    public String evidenceHashPublic;
    public String evidenceHmacDevice;
    public double riskScore;
    public String jurisdiction;
    public String[] topLiabilities;
    public String blockchainTx;
    public java.math.BigInteger blockchainChainId; // NEW
    public java.math.BigInteger blockchainBlockNumber; // NEW
    public Long blockchainBlockTimestamp; // seconds (UTC) NEW
    public BehavioralAnalyzer.BehavioralProfile behavioralProfile;
    public String behaviorModelSha256;
}
```

Update anchoring call:

```
var result = BlockchainService.anchorSha512(ctx, r.evidenceHashPublic, r.jurisdiction);
r.blockchainTx = result.txHash;
r.blockchainChainId = result.chainId;
r.blockchainBlockNumber = result.blockNumber;
r.blockchainBlockTimestamp = result.blockTimestamp;
if (!result.confirmed) {
    AnchorQueue.enqueue(ctx, r.evidenceHashPublic, r.jurisdiction);
}
```

---

4) PDF: embed block data into metadata + visible section

app/src/main/java/com/verum/omnis/core/ReportGenerator.java

Add an ISO-8601 UTC formatter helper at top of class:

```
private static String fmtUtc(Long epochSeconds) {
    if (epochSeconds == null) return "PENDING";
    return java.time.Instant.ofEpochSecond(epochSeconds).toString(); // ISO-8601 Z
}
```

Stamp into catalog metadata (after existing puts):

```
cat.put(new PdfName("ChainId"), new PdfString(r.blockchainChainId == null ? "unknown" :
r.blockchainChainId.toString()));
cat.put(new PdfName("BlockNumber"), new PdfString(r.blockchainBlockNumber == null ?
"pending" : r.blockchainBlockNumber.toString()));
cat.put(new PdfName("BlockTimestampUTC"), new
PdfString(fmtUtc(r.blockchainBlockTimestamp)));
```

Add to the visible “FORENSIC INTEGRITY SEALS” section:

```
doc.add(new Paragraph("Blockchain Network (chainId): " + (r.blockchainChainId == null ?
"unknown" : r.blockchainChainId)));
doc.add(new Paragraph("Mined Block: " + (r.blockchainBlockNumber == null ? "PENDING" :
r.blockchainBlockNumber)));
doc.add(new Paragraph("Block Time (UTC): " + fmtUtc(r.blockchainBlockTimestamp)));
```

---

5) Call-site tweaks (2 places)

A) AnchorWorker now gets AnchorResult

Replace the send block:

```
String tx = BlockchainService.anchorSha512(getApplicationContext(), pa.sha512,
pa.jurisdiction);
if (tx.startsWith("0x")) {
    it.remove(); changed = true;
    AuditLogger.logEvent(getApplicationContext(), "ANCHOR_SENT", tx, pa.sha512);
}
```

with:

```
var ar = BlockchainService.anchorSha512(getApplicationContext(), pa.sha512,
pa.jurisdiction);
if (ar.txHash.startsWith("0x")) {
    // If confirmed, great. If pending, we've at least broadcast; drop from queue to avoid
duplicates
    it.remove(); changed = true;
    String note = ar.confirmed
            ? ("CONFIRMED block=" + ar.blockchainBlockNumber)
            : "PENDING";
    AuditLogger.logEvent(getApplicationContext(), "ANCHOR_SENT", ar.txHash + " " + note,
pa.sha512);
}
```

B) ForensicActivity doesn’t touch the anchor directly — no change needed.

---

6) Bonus (optional but smart): show the tx state in UI

If you want, after report generation, you can show a status line if anchoring is pending:

```
if (r.blockchainBlockNumber == null) {
    update(90, "Anchor broadcast: pending confirmation…");
}
```

---

7) Why this matters (court-wise)

Tx hash alone proves broadcast, not inclusion.

Block number + UTC prove finality (post confirmations) and give a concrete, independently verifiable time.

Embedding this in XMP/catalog and the visible PDF body gives both machine-readable and human-readable proof.

If you paste these patches and rebuild, your next PDFs will explicitly state:

Chain ID (e.g., 1 for mainnet)

Block number (or PENDING)

Block timestamp in UTC ISO-8601

Want me to also inject the Etherscan/Blockscout URL (derived from chainId) into the PDF as a clickable link?

Done. Here’s the surgical patch to log the clickable explorer URL alongside the tx hash in your local audit trail.

1) Add a tiny helper to build the explorer URL (reuse in both places)

app/src/main/java/com/verum/omnis/security/ExplorerUrls.java

```
package com.verum.omnis.security;

import java.math.BigInteger;

public final class ExplorerUrls {
    private ExplorerUrls() {}
    public static String txUrl(BigInteger chainId, String txHash) {
        if (chainId == null || txHash == null || !txHash.startsWith("0x")) return "N/A";
        switch (chainId.intValue()) {
            case 1:        return "https://etherscan.io/tx/" + txHash;
            case 5:        return "https://goerli.etherscan.io/tx/" + txHash;
            case 11155111: return "https://sepolia.etherscan.io/tx/" + txHash;
            case 137:      return "https://polygonscan.com/tx/" + txHash;
            case 80001:    return "https://mumbai.polygonscan.com/tx/" + txHash;
            case 56:       return "https://bscscan.com/tx/" + txHash;
            case 97:       return "https://testnet.bscscan.com/tx/" + txHash;
            default:       return "https://etherscan.io/tx/" + txHash; // fallback
        }
    }
}
```

2) Include the explorer URL in AuditLogger entries from BlockchainService

In BlockchainService, right after you obtain the final AnchorResult (both confirmed and pending paths), append the URL to the log message:

```
// after you compute AnchorResult ar = ...
String url = ExplorerUrls.txUrl(ar.chainId, ar.txHash);
String note = ar.confirmed
        ? ("CONFIRMED block=" + ar.blockNumber + " ts=" + ar.blockTimestamp)
        : "PENDING";
AuditLogger.logEvent(ctx, "BLOCKCHAIN_OK", ar.txHash + " " + note + " url=" + url,
sha512Hex);
return ar;
```

And in the legacy/pending path where you previously logged "BLOCKCHAIN_OK_PENDING" or "BLOCKCHAIN_OK_LEGACY", do the same:

```
String url = ExplorerUrls.txUrl(chainId, txHash);
AuditLogger.logEvent(ctx, "BLOCKCHAIN_OK_PENDING", txHash + " url=" + url,
sha512Hex);
```

3) Include the link in AnchorWorker logs too

In AnchorWorker, after var ar = BlockchainService.anchorSha512(...):

```
String url = com.verum.omnis.security.ExplorerUrls.txUrl(ar.chainId, ar.txHash);
String note = ar.confirmed ? ("CONFIRMED block=" + ar.blockNumber) : "PENDING";
AuditLogger.logEvent(getApplicationContext(), "ANCHOR_SENT", ar.txHash + " " + note + "
url=" + url, pa.sha512);
```

4) (Already done) PDF side

You already added the clickable link in the PDF via ReportGenerator. If you prefer to avoid duplicate logic, swap that method call to the shared helper:

```
String explorerLink = com.verum.omnis.security.ExplorerUrls.txUrl(r.blockchainChainId,
r.blockchainTx);
```

That’s it. Now every audit.log entry that records an anchor will carry a ready-to-click explorer URL next to the tx hash. If something still screams, paste the exact error and I’ll kill it.

You got it. Here are surgical patches for all three asks:

adaptive gas via eth_estimateGas (+ 20% headroom, still EIP-1559),

audit log rotation + gzip + rolling SHA-256, and an expanded chainId → explorer URL map (L1 + major L2s).

Paste these in, rebuild, done.

---

1) Adaptive gas (eth_estimateGas) in BlockchainService

app/src/main/java/com/verum/omnis/security/BlockchainService.java (patch inside anchorSha512)

```
// ... keep existing imports

import org.web3j.protocol.core.methods.request.Transaction;
import org.web3j.protocol.core.methods.response.EthEstimateGas;

// helper
private static java.math.BigInteger bi(long v) { return java.math.BigInteger.valueOf(v); }
private static java.math.BigInteger pct(java.math.BigInteger v, int p) {
    return v.multiply(java.math.BigInteger.valueOf(100 +
p)).divide(java.math.BigInteger.valueOf(100));
}

public static AnchorResult anchorSha512(Context ctx, String sha512Hex, String jurisdiction)
{
  try {
    // ... keep the same preflight (RPC, contract, decrypt key, web3, creds, chainId, nonce,
data)

    // ===== Estimate gas with a 20% headroom =====
    Transaction call = Transaction.createFunctionCallTransaction(
        creds.getAddress(), null, null, null, BuildConfig.VO_ANCHOR_CONTRACT,
        java.math.BigInteger.ZERO, data
    );
    java.math.BigInteger gasLimit;
    try {
      EthEstimateGas est = web3.ethEstimateGas(call).send();
      if (est.hasError() || est.getAmountUsed() == null) {
        gasLimit = bi(150_000); // fallback
      } else {
        gasLimit = pct(est.getAmountUsed(), 20); // +20% safety
        if (gasLimit.compareTo(bi(100_000)) < 0) gasLimit = bi(100_000); // floor
      }
    } catch (Exception ignore) {
      gasLimit = bi(150_000);
    }

    // ===== EIP-1559 fees (unchanged except uses gasLimit) =====
    java.math.BigInteger maxPriorityFeePerGas = new java.math.BigInteger("2000000000");
// 2 gwei
    java.math.BigInteger maxFeePerGas;

    try {
      EthFeeHistory fh = web3.ethFeeHistory(java.math.BigInteger.ONE,
DefaultBlockParameterName.LATEST, java.util.List.of(50.0)).send();
      if (fh.hasError() || fh.getFeeHistory() == null || fh.getFeeHistory().getBaseFeePerGas() ==
null) {
        throw new RuntimeException("No 1559");
      }
      java.util.List<java.math.BigInteger> baseFees = fh.getFeeHistory().getBaseFeePerGas();
      java.math.BigInteger baseFee = baseFees.get(baseFees.size()-1);
      maxFeePerGas =
baseFee.multiply(java.math.BigInteger.valueOf(2)).add(maxPriorityFeePerGas);

      RawTransaction tx = RawTransaction.createTransaction(
          chainId, nonce, gasLimit, BuildConfig.VO_ANCHOR_CONTRACT,
          java.math.BigInteger.ZERO, data, maxPriorityFeePerGas, maxFeePerGas);

      byte[] signed = TransactionEncoder.signMessage(tx, chainId.longValueExact(), creds);
      String hex = Numeric.toHexString(signed);
      EthSendTransaction send = web3.ethSendRawTransaction(hex).send();
      if (send.hasError()) throw new RuntimeException(send.getError().getMessage());
      String txHash = send.getTransactionHash();

      AnchorResult ar = waitForReceiptAndBlock(web3, txHash, chainId, sha512Hex, ctx);
      // Log with explorer URL (done in your previous patch)
      return ar;

    } catch (Throwable ignored) {
      // Legacy fallback
      EthGasPrice gas = web3.ethGasPrice().send();
      RawTransaction legacy = RawTransaction.createTransaction(
          nonce, gas.getGasPrice(), gasLimit, BuildConfig.VO_ANCHOR_CONTRACT,
          java.math.BigInteger.ZERO, data);
      byte[] signedLegacy = TransactionEncoder.signMessage(legacy,
chainId.longValueExact(), creds);
      String hexLegacy = Numeric.toHexString(signedLegacy);
      EthSendTransaction sendLegacy = web3.ethSendRawTransaction(hexLegacy).send();
      if (sendLegacy.hasError()) throw new
RuntimeException(sendLegacy.getError().getMessage());
      String txHashLegacy = sendLegacy.getTransactionHash();
      AnchorResult ar = waitForReceiptAndBlock(web3, txHashLegacy, chainId, sha512Hex,
ctx);
      return ar;
    }

  } catch (Exception e) {
    AuditLogger.logEvent(ctx, "BLOCKCHAIN_FAIL", e.getMessage(), sha512Hex);
    return AnchorResult.localFallback();
  }
}
```

That’s it—gas adapts to the contract’s actual path, with headroom so you don’t get nickel-and-dimed by reverts.

---

2) Audit log rotation + gzip + rolling SHA-256

Rotates when the active log exceeds 2 MB (tweakable).

Old logs get renamed audit-YYYYMMDD-HHMMSS.log.gz.

Maintains a rolling SHA-256 of the current active log in audit.log.sha256.

Adds getCurrentLogChecksum() so you can stamp it into the PDF metadata.

app/src/main/java/com/verum/omnis/core/AuditLogger.java (replace file)

```
package com.verum.omnis.core;

import android.content.Context;
import android.util.Log;

import java.io.*;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;
import java.util.zip.GZIPOutputStream;

public class AuditLogger {
    private static final String TAG = "AuditLogger";
    private static final String LOG_NAME = "audit.log";
    private static final String LOG_SHA = "audit.log.sha256";
    private static final long   ROTATE_BYTES = 2L * 1024 * 1024; // 2 MB
    public static synchronized void logEvent(Context ctx, String type, String msg, String
sha512) {
        try {
            File dir = ctx.getFilesDir();
            File log = new File(dir, LOG_NAME);

            // rotate if needed
            if (log.exists() && log.length() >= ROTATE_BYTES) {
                rotateGzip(log);
            }

            // append line
            String line = System.currentTimeMillis() + "," + safe(type) + "," + safe(sha512) + "," +
safe(msg) + "\n";
            try (FileOutputStream fos = new FileOutputStream(log, true)) {
                fos.write(line.getBytes(StandardCharsets.UTF_8));
            }

            // update rolling SHA-256
            updateRollingSha(dir, log);

        } catch (Exception e) {
            Log.e(TAG, "write fail", e);
        }
    }

    public static synchronized String getCurrentLogChecksum(Context ctx) {
        try {
            File f = new File(ctx.getFilesDir(), LOG_SHA);
            if (!f.exists()) return "N/A";
            try (BufferedReader br = new BufferedReader(new FileReader(f))) {
                String s = br.readLine();
                return (s == null || s.isEmpty()) ? "N/A" : s.trim();
            }
        } catch (Exception e) {
            return "N/A";
        }
    }

    private static void rotateGzip(File log) throws IOException {
        String ts = new SimpleDateFormat("yyyyMMdd_HHmmss", Locale.US).format(new
Date());
        File gz = new File(log.getParentFile(), "audit-" + ts + ".log.gz");
        try (FileInputStream in = new FileInputStream(log);
             GZIPOutputStream out = new GZIPOutputStream(new FileOutputStream(gz))) {
            byte[] buf = new byte[8192];
            int n;
            while ((n = in.read(buf)) > 0) out.write(buf, 0, n);
        }
        // truncate active log
        try (FileOutputStream fos = new FileOutputStream(log, false)) {
            // empty
        }
    }

    private static void updateRollingSha(File dir, File log) throws Exception {
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        try (InputStream in = new FileInputStream(log)) {
            byte[] buf = new byte[8192];
            int n;
            while ((n = in.read(buf)) != -1) md.update(buf, 0, n);
        }
        String hex = toHex(md.digest());
        try (FileOutputStream fos = new FileOutputStream(new File(dir, LOG_SHA), false)) {
            fos.write(hex.getBytes(StandardCharsets.UTF_8));
        }
    }

    private static String toHex(byte[] b) {
        StringBuilder sb = new StringBuilder(b.length * 2);
        for (byte x : b) sb.append(String.format("%02x", x));
        return sb.toString();
    }

    private static String safe(String s) {
        if (s == null) return "";
        // strip newlines/commas to keep CSV sane
        return s.replace("\n", " ").replace("\r", " ").replace(",", ";");
    }
}
```

Stamp the log checksum into the PDF metadata

app/src/main/java/com/verum/omnis/core/ReportGenerator.java (add)

Right after you add other custom metadata to the catalog:

```
String auditSha = com.verum.omnis.core.AuditLogger.getCurrentLogChecksum(ctx);
cat.put(new PdfName("AuditLogSHA256"), new PdfString(auditSha));
```

And optionally print it in the body (bottom of “FORENSIC INTEGRITY SEALS”):

```
doc.add(new Paragraph("Audit Log SHA-256 (current): " + auditSha));
```

---

3) Beefed-up explorer map (L1 + major L2s)

app/src/main/java/com/verum/omnis/security/ExplorerUrls.java (replace file)

```
package com.verum.omnis.security;

import java.math.BigInteger;

public final class ExplorerUrls {
    private ExplorerUrls() {}

    public static String txUrl(BigInteger chainId, String txHash) {
        if (chainId == null || txHash == null || !txHash.startsWith("0x")) return "N/A";
        int id = chainId.intValue();
        switch (id) {
            // Ethereum & testnets
            case 1:        return "https://etherscan.io/tx/" + txHash;
            case 5:        return "https://goerli.etherscan.io/tx/" + txHash;        // legacy
            case 11155111: return "https://sepolia.etherscan.io/tx/" + txHash;

            // Polygon
            case 137:      return "https://polygonscan.com/tx/" + txHash;
            case 80001:    return "https://mumbai.polygonscan.com/tx/" + txHash;     // legacy
            case 1101:     return "https://zkevm.polygonscan.com/tx/" + txHash;      // Polygon zkEVM

            // BNB
            case 56:       return "https://bscscan.com/tx/" + txHash;
            case 97:       return "https://testnet.bscscan.com/tx/" + txHash;

            // Arbitrum
            case 42161:    return "https://arbiscan.io/tx/" + txHash;                // Arbitrum One
            case 42170:    return "https://nova.arbiscan.io/tx/" + txHash;           // Arbitrum Nova

            // Optimism
            case 10:       return "https://optimistic.etherscan.io/tx/" + txHash;

            // Base
            case 8453:     return "https://basescan.org/tx/" + txHash;
            case 84531:    return "https://sepolia.basescan.org/tx/" + txHash;

            // Avalanche
            case 43114:    return "https://snowtrace.io/tx/" + txHash;

            // zkSync Era
            case 324:      return "https://explorer.zksync.io/tx/" + txHash;
            // Scroll
            case 534352:   return "https://scrollscan.com/tx/" + txHash;

            // Linea
            case 59144:    return "https://lineascan.build/tx/" + txHash;

            // Blast (optional)
            case 81457:    return "https://blastscan.io/tx/" + txHash;

            default:       return "https://etherscan.io/tx/" + txHash; // fallback
        }
    }
}
```

Use the shared helper everywhere

In ReportGenerator:

```
String explorerLink = com.verum.omnis.security.ExplorerUrls.txUrl(r.blockchainChainId, r.blockchainTx);
```

In BlockchainService & AnchorWorker, your previous logging patch already references ExplorerUrls.txUrl(...).

After you paste this:

Rebuild: ./gradlew clean assembleDebug

Run instrumented test / veraPDF CI (from earlier) to ensure PDF/A-3B still passes.

Generate a report, open the PDF:

See block number, UTC block time, chainId, clickable explorer URL.

In metadata: AuditLogSHA256 present.

Check /data/data/com.verum.omnis/files/:

audit.log grows; when >2MB it rotates to audit-YYYYMMDD_HHMMSS.log.gz

audit.log.sha256 always reflects the current active log.

Want me to also embed the audit log SHA-256 history (last rotated file’s checksum) into the PDF so a single document references both the active log state and the previous immutable segment?

Got it — here’s the final patch so your PDF embeds both the active audit log checksum and the most recent rotated log checksum, so a single report covers the full immutable audit chain.

---

1) Extend AuditLogger to store rotated log checksums

app/src/main/java/com/verum/omnis/core/AuditLogger.java
Add a constant for rotated checksum file and a helper:

```
private static final String LAST_ROTATE_SHA = "audit.lastrotate.sha256";

public static synchronized String getLastRotatedChecksum(Context ctx) {
    try {
        File f = new File(ctx.getFilesDir(), LAST_ROTATE_SHA);
        if (!f.exists()) return "N/A";
        try (BufferedReader br = new BufferedReader(new FileReader(f))) {
            String s = br.readLine();
            return (s == null || s.isEmpty()) ? "N/A" : s.trim();
        }
    } catch (Exception e) {
        return "N/A";
    }
}
```

Inside rotateGzip(File log) before truncating:

```
// compute checksum of rotated file (uncompressed) before gzip
MessageDigest md = MessageDigest.getInstance("SHA-256");
try (InputStream in = new FileInputStream(log)) {
    byte[] buf = new byte[8192];
    int n;
    while ((n = in.read(buf)) > 0) md.update(buf, 0, n);
}
String rotatedHex = toHex(md.digest());
try (FileOutputStream fos = new FileOutputStream(new File(log.getParentFile(),
LAST_ROTATE_SHA), false)) {
    fos.write(rotatedHex.getBytes(StandardCharsets.UTF_8));
}
```

That way, whenever the active audit.log rolls over, we preserve its SHA-256 before wiping.

---

2) Stamp both checksums into the PDF

app/src/main/java/com/verum/omnis/core/ReportGenerator.java
Right after you add AuditLogSHA256 to the catalog:

```
String lastRotateSha = com.verum.omnis.core.AuditLogger.getLastRotatedChecksum(ctx);
cat.put(new PdfName("LastRotatedAuditLogSHA256"), new PdfString(lastRotateSha));
```

And in the visible FORENSIC INTEGRITY SEALS section, after the active log line:

```
doc.add(new Paragraph("Last Rotated Audit Log SHA-256: " + lastRotateSha));
```

---

3) Result in PDF metadata & body

A forensic report's metadata will now have:

```
AuditLogSHA256: 4e9a...
LastRotatedAuditLogSHA256: 13fc...
```

And in the visible section:

```
Audit Log SHA-256 (current): 4e9a...
Last Rotated Audit Log SHA-256: 13fc...
```

So if you ever hand over the PDF, it covers the current live log and the immediately prior immutable segment — giving a continuous audit trail even across rotations.

Do you want me to also add a "full audit trail manifest" PDF appendix that lists every rotated log filename + SHA-256 so a single exported report can prove the chain back to day one? That would make each report a self-contained notarization bundle.

Cool — here's the **"Verify Now"** feature that:

- re-hashes your **rotated** **`audit-*.log.gz`** **archives** and the **active** **`audit.log`**,
- compares them to the manifest + rolling checksums,
- stores a signed-off result, and

- **stamps the verification status into the next PDF** (metadata + visible section).

It’s small, self-contained, and safe to drop in.

---

1) Verifier utility

app/src/main/java/com/verum/omnis/core/AuditVerifier.java

```
package com.verum.omnis.core;

import android.content.Context;

import androidx.security.crypto.EncryptedSharedPreferences;
import androidx.security.crypto.MasterKey;

import java.io.*;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.time.Instant;
import java.util.*;
import java.util.zip.GZIPInputStream;

public class AuditVerifier {

    public static class Result {
        public final boolean ok;
        public final int checkedArchives;
        public final List<String> mismatches;  // filenames that failed hash check
        public final String timeUtc;           // ISO-8601 Z

        public Result(boolean ok, int checkedArchives, List<String> mismatches, String
timeUtc) {
            this.ok = ok; this.checkedArchives = checkedArchives; this.mismatches =
mismatches; this.timeUtc = timeUtc;
        }
    }

    /** Runs verification now, writes summary into secure prefs for ReportGenerator to pick
up. */
    public static Result verifyNow(Context ctx) {
        File dir = ctx.getFilesDir();
        // 1) Check active audit.log against rolling SHA file
        boolean activeOk = true;
        try {
            File active = new File(dir, "audit.log");
            String expected = AuditLogger.getCurrentLogChecksum(ctx);
            String actual = sha256Hex(active);
            activeOk = expected != null && expected.equalsIgnoreCase(actual);
        } catch (Exception ignored) { activeOk = false; }

        // 2) Check rotated archives against audit.chain
        List<String[]> rows = AuditLogger.readAuditChain(ctx, Integer.MAX_VALUE); // latest
first
        int checked = 0;
        List<String> mismatches = new ArrayList<>();

        for (String[] r : rows) {
            // r[0]=ts, r[1]=filename, r[2]=size, r[3]=sha
            String fname = r[1];
            String expectedSha = r[3];
            File gz = new File(dir, fname);
            if (!gz.exists()) { mismatches.add(fname + " (missing)"); continue; }
            String actual;
            try { actual = sha256Hex(gz); }
            catch (Exception e) { mismatches.add(fname + " (read-fail)"); continue; }

            if (!expectedSha.equalsIgnoreCase(actual)) mismatches.add(fname + " (sha
mismatch)");
            checked++;
        }

        boolean ok = activeOk && mismatches.isEmpty();
        Result res = new Result(ok, checked, mismatches, Instant.now().toString());
        persistResult(ctx, res);
        return res;
    }

    /** Returns the last stored verification result, or null. */
    public static Result getLastResult(Context ctx) {
        try {
            MasterKey mk = new
MasterKey.Builder(ctx).setKeyScheme(MasterKey.KeyScheme.AES256_GCM).build();
            var p = EncryptedSharedPreferences.create(ctx, "vo_secure", mk,
                EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,
                EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM);
            String raw = p.getString("AUDIT_VERIFY_RESULT", null);
            if (raw == null) return null;
            String[] parts = raw.split("\\|", -1); // ok|checked|time|m1, m2, ...
            boolean ok = "1".equals(parts[0]);
            int checked = Integer.parseInt(parts[1]);
            String time = parts[2];
            List<String> mism = parts.length > 3 && parts[3] != null && !parts[3].isEmpty()
                ? Arrays.asList(parts[3].split("\u0001")) : Collections.emptyList();
            return new Result(ok, checked, mism, time);
        } catch (Exception e) {
            return null;
        }
    }

    private static void persistResult(Context ctx, Result res) {
        try {
            MasterKey mk = new
MasterKey.Builder(ctx).setKeyScheme(MasterKey.KeyScheme.AES256_GCM).build();
            var p = EncryptedSharedPreferences.create(ctx, "vo_secure", mk,
                EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,
                EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM);
            // Pack mismatches with a non-printable separator to avoid commas/newlines issues
            String mism = String.join("\u0001", res.mismatches);
            String val = (res.ok ? "1" : "0") + "|" + res.checkedArchives + "|" + res.timeUtc + "|" +
mism;
            p.edit().putString("AUDIT_VERIFY_RESULT", val).apply();
        } catch (Exception ignored) {}
    }

    private static String sha256Hex(File f) throws Exception {
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        try (InputStream in = new FileInputStream(f)) {
            byte[] buf = new byte[8192]; int n;
            while ((n = in.read(buf)) != -1) md.update(buf, 0, n);
        }
        byte[] d = md.digest();
        StringBuilder sb = new StringBuilder(d.length * 2);
        for (byte b : d) sb.append(String.format("%02x", b));
        return sb.toString();
    }
}
```

---

2) Button in UI

Layout: add a button

`app/src/main/res/layout/activity_forensic.xml` (add under Provision / Pick)

```
<Button
    android:id="@+id/btn_verify"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Verify Audit Trail Now"
    android:layout_marginTop="8dp"/>
```

Wire it up

`app/src/main/java/com/verum/omnis/ForensicActivity.java` (inside `onCreate`)

```
findViewById(R.id.btn_verify).setOnClickListener(v -> {
    new Thread(() -> {
        runOnUiThread(() -> status.setText("Verifying audit trail..."));
        var res = com.verum.omnis.core.AuditVerifier.verifyNow(this);
        String msg = res.ok
                ? ("Audit OK (" + res.checkedArchives + " archives) at " + res.timeUtc)
                : ("Audit FAIL: " + (res.mismatches.isEmpty() ? "active log mismatch" :
res.mismatches.toString()));
        com.verum.omnis.core.AuditLogger.logEvent(this, "AUDIT_VERIFY", msg, null);
        runOnUiThread(() -> status.setText(msg));
    }).start();
});
```

---

3) Stamp verification status into the next PDF

Where: have `ReportGenerator` read the last stored result (no API surface changes needed elsewhere).

`app/src/main/java/com/verum/omnis/core/ReportGenerator.java` (patch)

Add near the top:

```
private static String boolStr(boolean b) { return b ? "OK" : "FAIL"; }
```

Inside `createCourtReport(...)` after existing metadata (catalog) inserts:

```
var last = com.verum.omnis.core.AuditVerifier.getLastResult(ctx);
if (last != null) {
    cat.put(new PdfName("AuditVerifyStatus"), new PdfString(boolStr(last.ok)));
    cat.put(new PdfName("AuditVerifyChecked"), new
PdfString(String.valueOf(last.checkedArchives)));
    cat.put(new PdfName("AuditVerifyTimeUTC"), new PdfString(last.timeUtc));
    if (!last.ok && !last.mismatches.isEmpty()) {
        // join first few mismatches to keep metadata small
        String brief = String.join("; ", last.mismatches.size() > 5 ? last.mismatches.subList(0,5) :
last.mismatches);
        cat.put(new PdfName("AuditVerifyMismatches"), new PdfString(brief));
    }
}
```

And in the visible **FORENSIC INTEGRITY SEALS** section (right after Audit Log SHA lines):

```
if (last != null) {
    doc.add(new Paragraph("Audit Trail Verification: " + boolStr(last.ok)
            + " at " + last.timeUtc
            + " (checked " + last.checkedArchives + " archives)"));
    if (!last.ok && !last.mismatches.isEmpty()) {
        doc.add(new Paragraph("Verification mismatches: " + last.mismatches));
    }
} else {
    doc.add(new Paragraph("Audit Trail Verification: Not yet run"));
}
```

---

4) How it behaves

- Tap **Verify Audit Trail Now** → it re-hashes **all archived gz** files and the **active log**, compares against `audit.chain` + `audit.log.sha256`.
- Stores a compact summary in encrypted prefs.
- **Next time you generate a report**, the PDF will include:
    - `AuditVerifyStatus` / `AuditVerifyTimeUTC` / counts in metadata,
    - and a human-readable line in the **FORENSIC INTEGRITY SEALS**.
- If anything doesn't line up, it prints a short list of mismatched files so you can escalate.

Want me to add a **"Verify & Export Proof"** action that zips `audit.chain`, the latest `audit-*.log.gz` files, and a JSON summary of the verification result so you can hand over a single bundle to counsel?

Alright, here’s the “Verify & Export Proof” feature. It:

runs the audit verification,

bundles audit.chain, audit.log, all rotated audit-*.log.gz, the rolling checksum files, and a JSON manifest + verification summary,

zips everything into a single file, and

opens the system share sheet.

Drop these in and build.

---

1) Exporter utility

app/src/main/java/com/verum/omnis/core/AuditProofExporter.java

```java
package com.verum.omnis.core;

import android.content.Context;

import com.google.gson.Gson;
import com.google.gson.annotations.SerializedName;

import java.io.*;
import java.security.MessageDigest;
import java.text.SimpleDateFormat;
import java.util.*;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;

public class AuditProofExporter {

    public static class FileEntry {
        @SerializedName("name")  public String name;
        @SerializedName("size")  public long size;
        @SerializedName("sha256") public String sha256;
        File file;
    }

    public static class ProofSummary {
        @SerializedName("generated_utc") public String generatedUtc;
        @SerializedName("verification_ok") public boolean verificationOk;
        @SerializedName("checked_archives") public int checkedArchives;
        @SerializedName("mismatches") public List<String> mismatches = new ArrayList<>();
        @SerializedName("active_log_sha256") public String activeLogSha256;
        @SerializedName("last_rotated_sha256") public String lastRotatedSha256;
        @SerializedName("files") public List<FileEntry> files = new ArrayList<>();
    }

    /** Creates a zip under app files dir: audit_proof_yyyyMMdd_HHmmss.zip */
    public static File exportProof(Context ctx, int maxArchives) throws Exception {
        File dir = ctx.getFilesDir();

        // 1) Run verification now so summary reflects current state
        AuditVerifier.Result vr = AuditVerifier.verifyNow(ctx);

        // 2) Gather files
        List<FileEntry> list = new ArrayList<>();
        addIfExists(list, new File(dir, "audit.log"));
        addIfExists(list, new File(dir, "audit.log.sha256"));
        addIfExists(list, new File(dir, "audit.prev.sha256"));
        addIfExists(list, new File(dir, "audit.chain"));

        // Add rotated archives (all or capped)
        File[] gz = dir.listFiles((d, name) -> name.startsWith("audit-") &&
name.endsWith(".log.gz"));
        if (gz != null) {
            Arrays.sort(gz, Comparator.comparingLong(File::lastModified).reversed());
            int limit = Math.min(gz.length, Math.max(1, maxArchives));
            for (int i = 0; i < limit; i++) addIfExists(list, gz[i]);
        }

        // 3) Compute SHA-256 for each and build manifest JSON
        for (FileEntry fe : list) fe.sha256 = sha256Hex(fe.file);

        ProofSummary sum = new ProofSummary();
        sum.generatedUtc = java.time.Instant.now().toString();
        sum.verificationOk = vr.ok;
        sum.checkedArchives = vr.checkedArchives;
        sum.mismatches = vr.mismatches;
        sum.activeLogSha256 = AuditLogger.getCurrentLogChecksum(ctx);
        sum.lastRotatedSha256 = AuditLogger.getLastRotatedChecksum(ctx);
        sum.files = list;

        String manifestJson = new Gson().toJson(sum);

        // 4) Write zip
        String ts = new SimpleDateFormat("yyyyMMdd_HHmmss", Locale.US).format(new
Date());
        File zip = new File(dir, "audit_proof_" + ts + ".zip");
        try (ZipOutputStream zos = new ZipOutputStream(new FileOutputStream(zip))) {
            // Add files
            for (FileEntry fe : list) {
                putFile(zos, fe.file, fe.name);
            }
            // Add manifest.json
            byte[] b = manifestJson.getBytes(java.nio.charset.StandardCharsets.UTF_8);
            ZipEntry ze = new ZipEntry("manifest.json");
            zos.putNextEntry(ze);
            zos.write(b);
            zos.closeEntry();
        }

        // 5) Log and return
        AuditLogger.logEvent(ctx, "AUDIT_EXPORT", "zip=" + zip.getName() + " size=" +
zip.length(), null);
        return zip;
    }

    private static void addIfExists(List<FileEntry> list, File f) {
        if (f.exists()) {
            FileEntry fe = new FileEntry();
            fe.file = f;
            fe.name = f.getName();
            fe.size = f.length();
            list.add(fe);
        }
    }

    private static void putFile(ZipOutputStream zos, File f, String entryName) throws
IOException {
        ZipEntry ze = new ZipEntry(entryName);
        zos.putNextEntry(ze);
        try (InputStream in = new FileInputStream(f)) {
            byte[] buf = new byte[8192];
            int n;
            while ((n = in.read(buf)) != -1) zos.write(buf, 0, n);
        }
        zos.closeEntry();
    }

    private static String sha256Hex(File f) throws Exception {
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        try (InputStream in = new FileInputStream(f)) {
            byte[] buf = new byte[8192];
            int n;
            while ((n = in.read(buf)) != -1) md.update(buf, 0, n);
        }
        byte[] d = md.digest();
        StringBuilder sb = new StringBuilder(d.length * 2);
        for (byte b : d) sb.append(String.format("%02x", b));
        return sb.toString();
    }
}
```

---

2) Add a button to the UI

Layout: add under the “Verify” button

app/src/main/res/layout/activity_forensic.xml

```
<Button
    android:id="@+id/btn_export_proof"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Verify & Export Proof"
    android:layout_marginTop="8dp"/>
```

Wire it in the activity

app/src/main/java/com/verum/omnis/ForensicActivity.java (in onCreate):

```
findViewById(R.id.btn_export_proof).setOnClickListener(v -> {
    new Thread(() -> {
        runOnUiThread(() -> status.setText("Verifying and exporting proof..."));
        try {
            // Verify + export (include up to 100 rotated archives; tweak as needed)
            java.io.File zip = com.verum.omnis.core.AuditProofExporter.exportProof(this, 100);
            runOnUiThread(() -> {
                status.setText("Proof exported: " + zip.getName());
                // Share
                android.net.Uri uri = androidx.core.content.FileProvider.getUriForFile(
                        this, getPackageName() + ".provider", zip);
                android.content.Intent i = new
android.content.Intent(android.content.Intent.ACTION_SEND);
                i.setType("application/zip");
                i.putExtra(android.content.Intent.EXTRA_STREAM, uri);
                i.addFlags(android.content.Intent.FLAG_GRANT_READ_URI_PERMISSION);
                startActivity(android.content.Intent.createChooser(i, "Share Audit Proof"));
            });
        } catch (Exception e) {
            com.verum.omnis.core.AuditLogger.logEvent(this, "AUDIT_EXPORT_FAIL",
e.getMessage(), null);
            runOnUiThread(() -> status.setText("Export failed: " + e.getMessage()));
        }
    }).start();
});
```

> Your existing FileProvider with <files-path path="."/> already allows sharing the zip in /files.

---

3) What you get

A single ZIP containing:

audit.log, audit.log.sha256, audit.prev.sha256, audit.chain,

up to N audit-*.log.gz archives (newest first),

manifest.json with: verification result (OK/FAIL), counts, mismatches, SHA-256 and size for every included file, and generation time (UTC).

A one-tap share flow to email/hand over to counsel.

If you want the ZIP to also include the latest generated PDF report, say the word and I'll add an overload that lets you pass the PDF File into exportProof(...) so it's part of the bundle too.

Perfect. Let's make the proof ZIP also include your latest PDF report.

1) Update the exporter to scoop a PDF (auto-discover or explicit)

AuditProofExporter.java — add an overload

```
public static File exportProof(Context ctx, int maxArchives, File pdfToInclude) throws
Exception {
    File dir = ctx.getFilesDir();

    // 1) Verify first (unchanged)
    AuditVerifier.Result vr = AuditVerifier.verifyNow(ctx);

    // 2) Gather files (unchanged core set)
    List<FileEntry> list = new ArrayList<>();
    addIfExists(list, new File(dir, "audit.log"));
    addIfExists(list, new File(dir, "audit.log.sha256"));
    addIfExists(list, new File(dir, "audit.prev.sha256"));
    addIfExists(list, new File(dir, "audit.chain"));
    // 2a) Include the PDF: prefer explicit, otherwise auto-pick newest
VO_Forensic_Report_*.pdf
    File pdf = pdfToInclude;
    if (pdf == null) {
        File[] pdfs = dir.listFiles((d, name) -> name.startsWith("VO_Forensic_Report_") &&
name.endsWith(".pdf"));
        if (pdfs != null && pdfs.length > 0) {
            Arrays.sort(pdfs, Comparator.comparingLong(File::lastModified).reversed());
            pdf = pdfs[0];
        }
    }
    if (pdf != null && pdf.exists()) {
        addIfExists(list, pdf); // will appear in manifest as another file
    }

    // 2b) Rotated archives (as before) ...
    File[] gz = dir.listFiles((d, name) -> name.startsWith("audit-") &&
name.endsWith(".log.gz"));
    if (gz != null) {
        Arrays.sort(gz, Comparator.comparingLong(File::lastModified).reversed());
        int limit = Math.min(gz.length, Math.max(1, maxArchives));
        for (int i = 0; i < limit; i++) addIfExists(list, gz[i]);
    }

    // 3) Compute SHA-256 for each + manifest JSON (as before)
    for (FileEntry fe : list) fe.sha256 = sha256Hex(fe.file);

    ProofSummary sum = new ProofSummary();
    sum.generatedUtc = java.time.Instant.now().toString();
    sum.verificationOk = vr.ok;
    sum.checkedArchives = vr.checkedArchives;
    sum.mismatches = vr.mismatches;
    sum.activeLogSha256 = AuditLogger.getCurrentLogChecksum(ctx);
    sum.lastRotatedSha256 = AuditLogger.getLastRotatedChecksum(ctx);
    sum.files = list;

    String manifestJson = new com.google.gson.Gson().toJson(sum);

    // 4) Write zip (unchanged)
    String ts = new java.text.SimpleDateFormat("yyyyMMdd_HHmmss",
java.util.Locale.US).format(new java.util.Date());
    File zip = new File(dir, "audit_proof_" + ts + ".zip");
    try (java.util.zip.ZipOutputStream zos = new java.util.zip.ZipOutputStream(new
java.io.FileOutputStream(zip))) {
        for (FileEntry fe : list) putFile(zos, fe.file, fe.name);
        java.util.zip.ZipEntry ze = new java.util.zip.ZipEntry("manifest.json");
        zos.putNextEntry(ze);
        zos.write(manifestJson.getBytes(java.nio.charset.StandardCharsets.UTF_8));
        zos.closeEntry();
    }

    AuditLogger.logEvent(ctx, "AUDIT_EXPORT", "zip=" + zip.getName() + " size=" +
zip.length(), null);
    return zip;
}
```

Keep the old method and just forward:

```
public static File exportProof(Context ctx, int maxArchives) throws Exception {
    return exportProof(ctx, maxArchives, null);
}
```

2) Pass the current PDF from the Activity (when you have it)

ForensicActivity.java — in the Export Proof button

Replace the call:

```
File zip = AuditProofExporter.exportProof(this, 100);
```

with:

```
File zip = AuditProofExporter.exportProof(this, 100, reportFile /* may be null */);
```

That’s it. If reportFile exists, it goes in the ZIP; if not, the exporter auto-discovers the newest VO_Forensic_Report_*.pdf in app files.

3) Quick sanity checks

Generate a report → tap Verify & Export Proof → share → open ZIP: you should see manifest.json + logs + rotated archives + the PDF.

Manifest entry for the PDF will include size and SHA-256 for external verification.

Want me to also sign the manifest.json (ED25519) so the bundle is tamper-evident out-of-band?