Decentralized Video Verification System Using Blockchain to Detect and Prevent Deepfake Media

Prabhanshu Kumar Jha (EE22BT045)
Sachin Kumar (MC22BT019)
Department of Computer Science
IIT Dharwad

April 12, 2025

Abstract

The increasing prevalence of AI-generated deepfake videos has raised significant concerns regarding the trustworthiness of digital media. This project presents a decentralized platform that combines artificial intelligence with blockchain technology to detect, prevent, and trace deepfake media. By analyzing videos and images using perceptual hashing, histogram-based features, and deep learning models, the system compares uploads against previously stored media on the blockchain. Metadata is stored securely using smart contracts, and permission-based reuse is enforced. This hybrid approach ensures real-time verification, accountability, and transparency.

Contents

1 Introduction

Deepfakes are synthetic media in which a person in an existing image or video is replaced with someone else's likeness using artificial intelligence. Their increasing quality and accessibility have caused serious concerns in misinformation, political sabotage, and digital fraud. Current detection methods are largely centralized and reactive. Our system introduces a decentralized, proactive mechanism to detect and block deepfake media during the upload process itself.

2 Problem Statement

There is a lack of effective systems that verify the authenticity of videos in real time at the point of upload. Additionally, centralized databases are vulnerable to manipulation. We aim to solve this using blockchain for immutability and AI for intelligent detection.

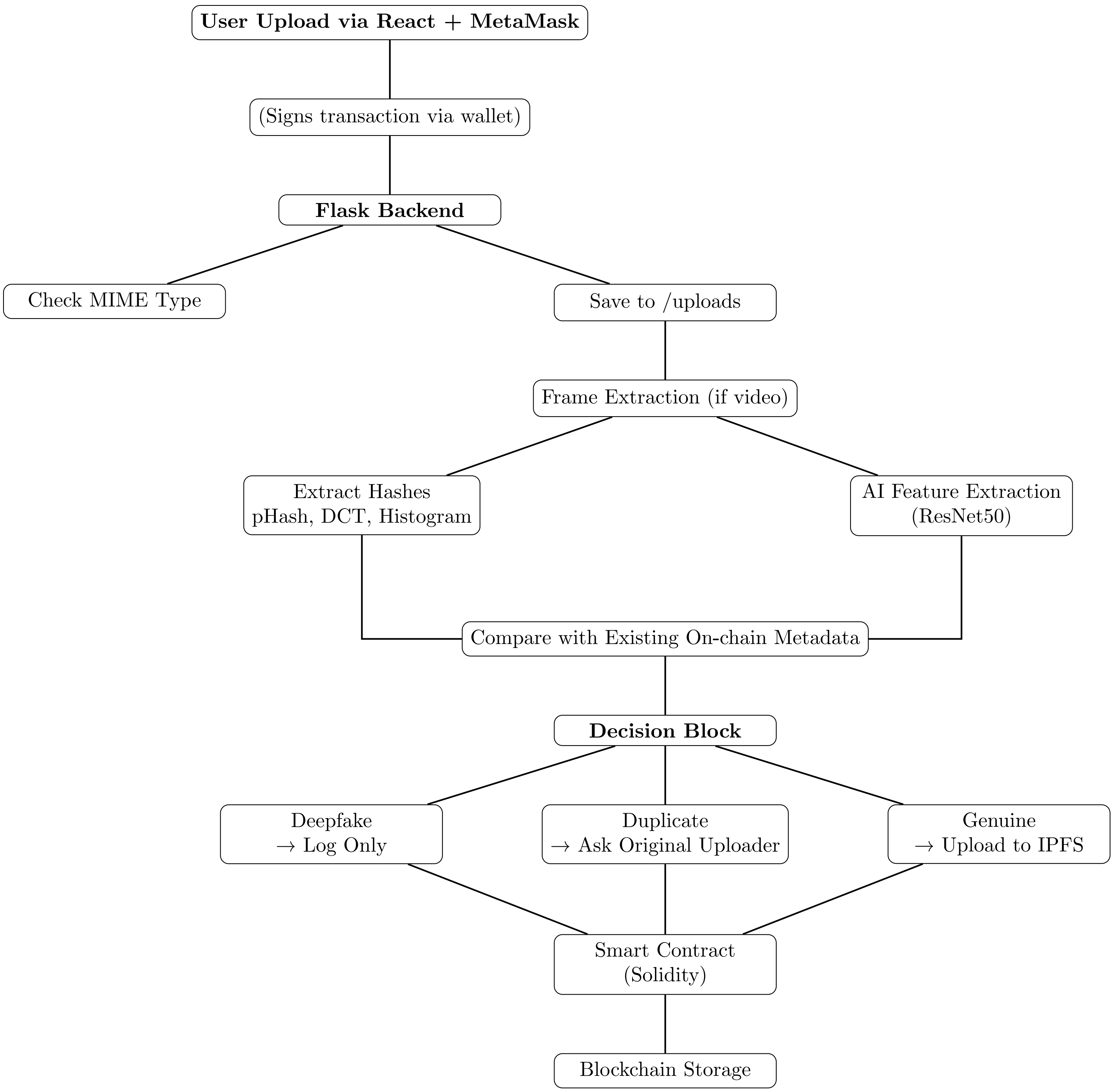
3 Proposed System

Our solution combines:

- React frontend with MetaMask for decentralized authentication
- Python Flask backend for media processing
- Smart contracts on Ethereum for recording metadata and enforcing permission-based reuse
- Perceptual hashing, DCT, Histogram, and AI features for similarity detection

4 System Architecture

4.1 System Design Flow

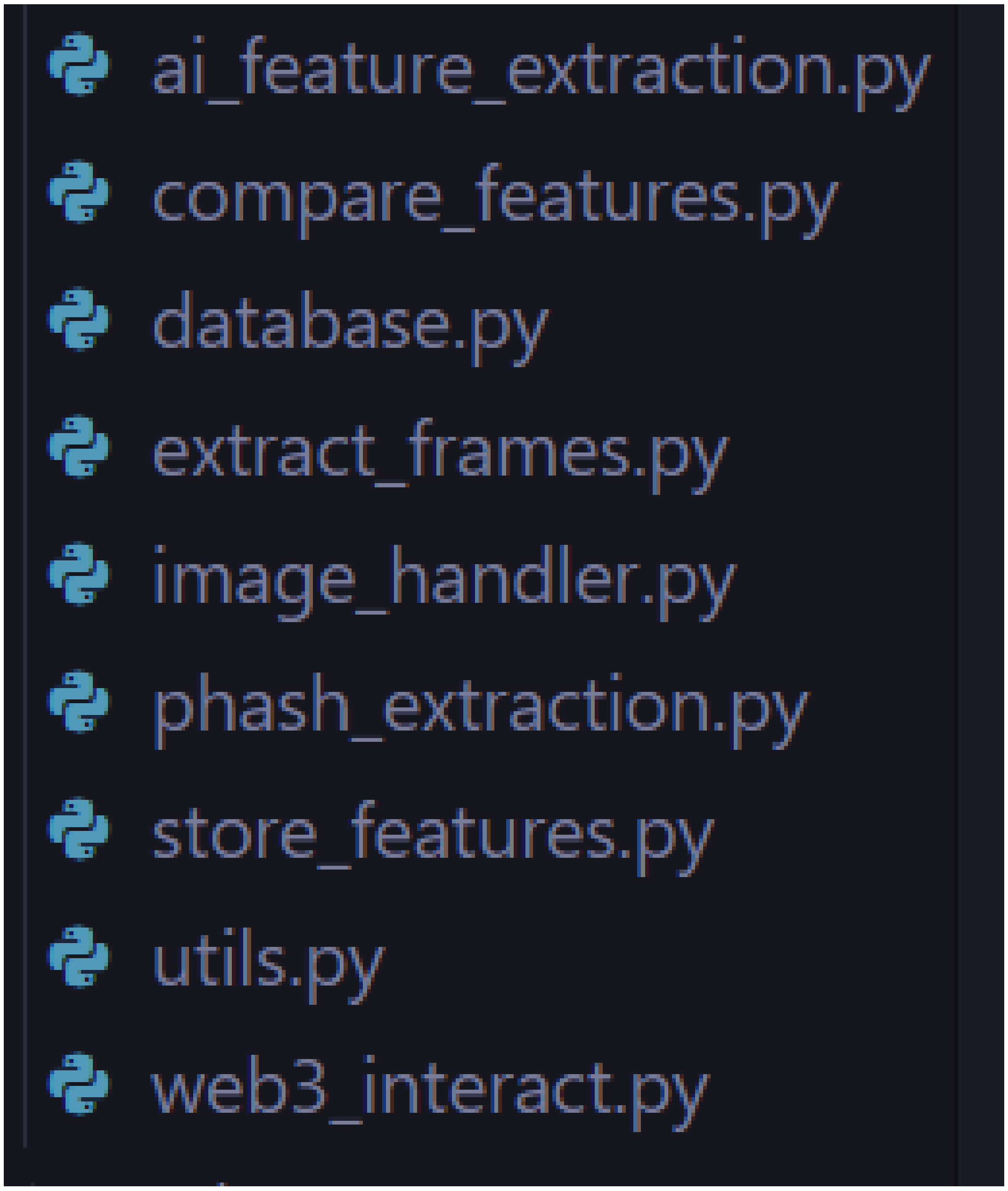

User Upload via React + MetaMask
(Signs transaction via wallet)
Flask Backend
Check MIME Type
Save to /uploads
Frame Extraction (if video)
Extract Hashes pHash, DCT, Histogram
AI Feature Extraction (ResNet50)
Compare with Existing On-chain Metadata
Decision Block
Deepfake → Log Only
Duplicate → Ask Original Uploader
Genuine → Upload to IPFS
Smart Contract (Solidity)
Blockchain Storage

5 Backend Architecture

5.1 Technologies Used

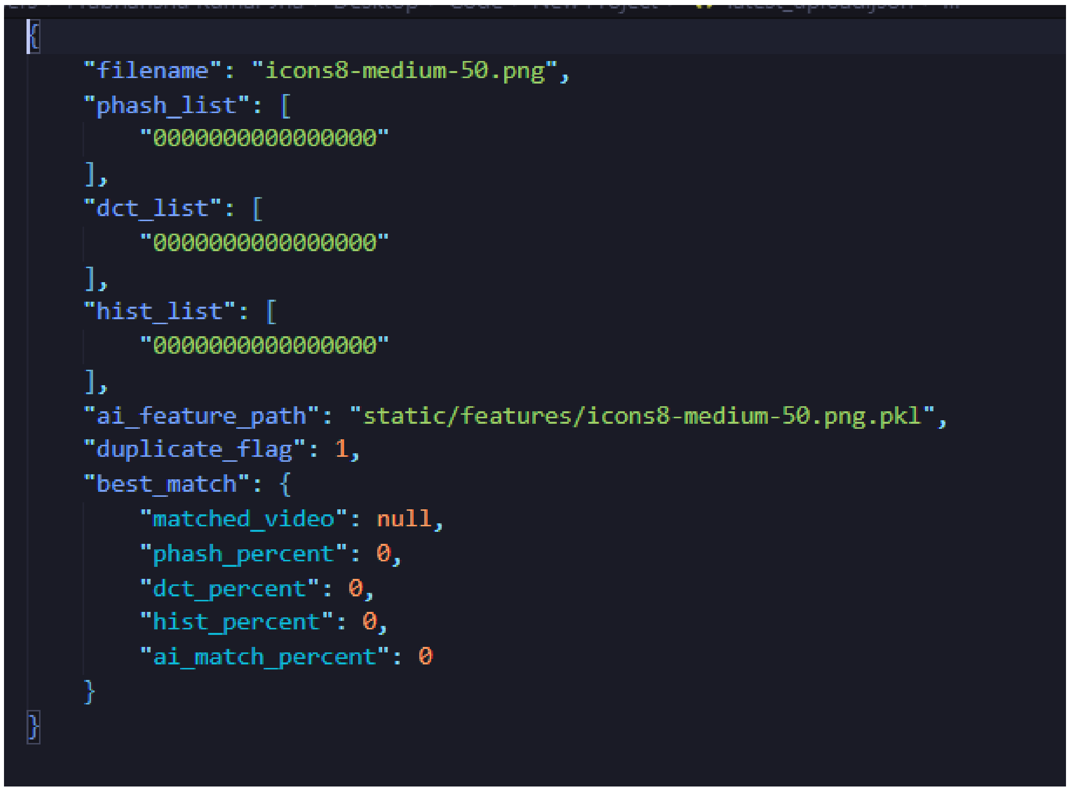
- Flask
- OpenCV for frame extraction

- ImageHash for perceptual hashing
- PyTorch for AI feature extraction
- SQLite and JSON for local storage and interim data



ai_feature_extraction.py
compare_features.py
database.py
extract_frames.py
image_handler.py
phash_extraction.py
store_features.py
utils.py
web3_interact.py

Figure 1: Flask backend logging the analysis pipeline



```
{
    "filename": "icons8-medium-50.png",
    "phash_list": [
        "0000000000000000"
    ],
    "dct_list": [
        "0000000000000000"
    ],
    "hist_list": [
        "0000000000000000"
    ],
    "ai_feature_path": "static/features/icons8-medium-50.png.pkl",
    "duplicate_flag": 1,
    "best_match": {
        "matched_video": null,
        "phash_percent": 0,
        "dct_percent": 0,
        "hist_percent": 0,
        "ai_match_percent": 0
    }
}
```

Figure 2: Sample contents of `latest_upload.json`

5.2 Flow

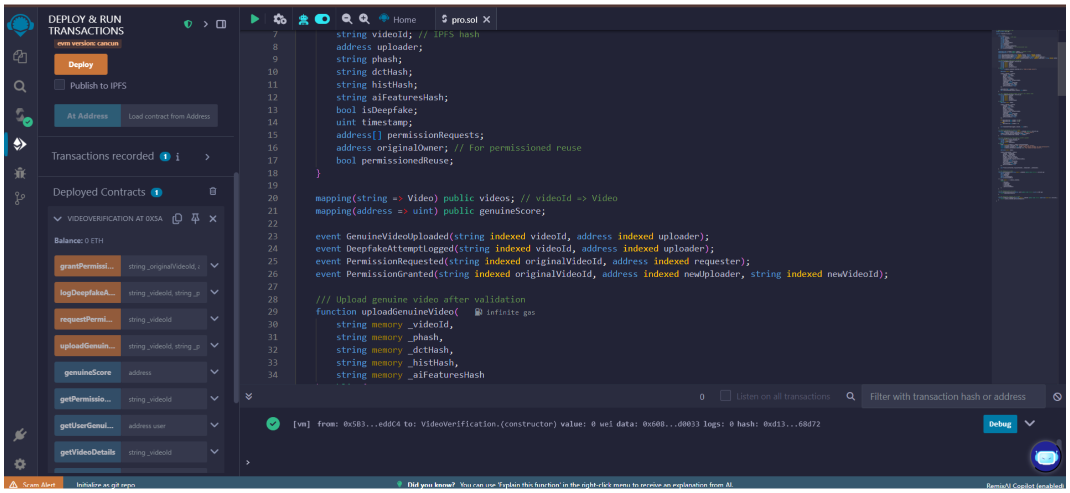
1. File uploaded via frontend
2. Flask identifies MIME type
3. If video → extract frames
4. Run pHash, DCT hash, Histogram
5. Run ResNet to extract AI features
6. Compare hashes and features with existing records
7. Save summary to `latest_upload.json`

6 Smart Contract Design

6.1 Responsibilities

- Upload metadata for genuine videos
- Log deepfake attempts (with penalty)

- Allow permission request workflow
- Track user's genuine score (`+1/-10`)


DEPLOY & RUN TRANSACTIONS
evm version: cancun
Deploy
Publish to IPFS
At Address
Load contract from Address
Transactions recorded
Deployed Contracts
VIDEOVERIFICATION AT 0X5A
Balance: 0 ETH
grantPermissi...
logDeepfakeA...
requestPermi...
uploadGenuin...
genuineScore
getPermissio...
getUserGenui...
getVideoDetails
Home
pro.sol
Listen on all transactions
Filter with transaction hash or address
Debug
Scam Alert
Initialize as git repo
Did you know? You can use 'Explain this function' in the right-click menu to receive an explanation from AI.
RemixAI Copilot (enabled)

```
        string videoId; // IPFS hash
        address uploader;
        string phash;
        string dctHash;
        string histHash;
        string aiFeaturesHash;
        bool isDeepfake;
        uint timestamp;
        address[] permissionRequests;
        address originalOwner; // For permissioned reuse
        bool permissionedReuse;
    }

    mapping(string => Video) public videos; // videoId => Video
    mapping(address => uint) public genuineScore;

    event GenuineVideoUploaded(string indexed videoId, address indexed uploader);
    event DeepfakeAttemptLogged(string indexed videoId, address indexed uploader);
    event PermissionRequested(string indexed originalVideoId, address indexed requester);
    event PermissionGranted(string indexed originalVideoId, address indexed newUploader, string indexed newVideoId);

    /// Upload genuine video after validation
    function uploadGenuineVideo(    infinite gas
        string memory _videoId,
        string memory _phash,
        string memory _dctHash,
        string memory _histHash,
        string memory _aiFeaturesHash
```

```
[vm] from: 0x5B3...eddC4 to: VideoVerification.(constructor) value: 0 wei data: 0x608...d0033 logs: 0 hash: 0xd13...68d72
```

Figure 3: Deploying the smart contract using Remix

6.2 Core Methods

- `uploadGenuineVideo()`
- `logDeepfakeAttempt()`
- `requestPermission()`
- `grantPermission()`
- `getUserGenuineScore()`

7 Frontend Interface

- Built in ReactJS
- Wallet connect via MetaMask
- Upload panel (image/video)
- Output feedback
- Feed of previously uploaded content

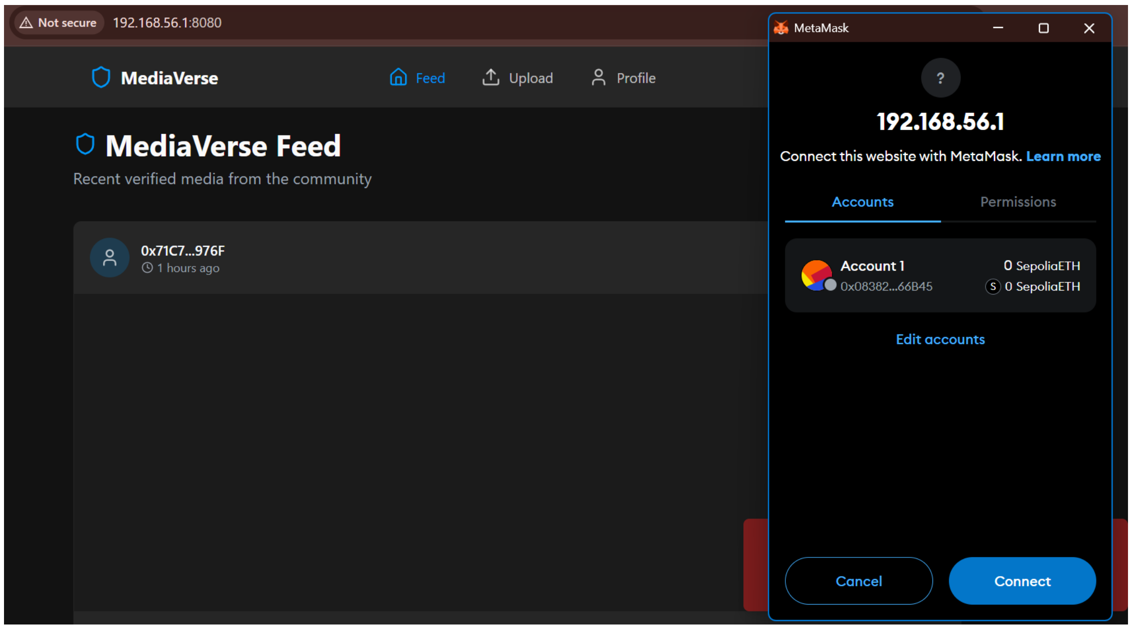
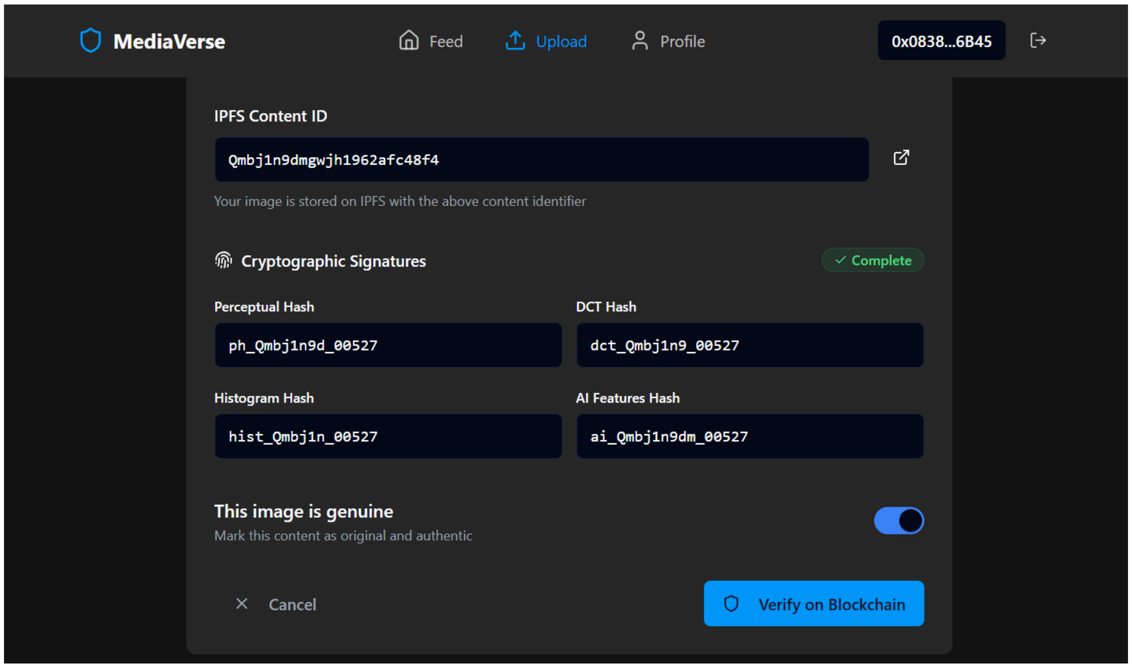


Figure 4: React frontend with MetaMask-connected video upload panel



IPFS Content ID

Qmbj1n9dmgwjh1962afc48f4

Your image is stored on IPFS with the above content identifier

Cryptographic Signatures ✓ Complete

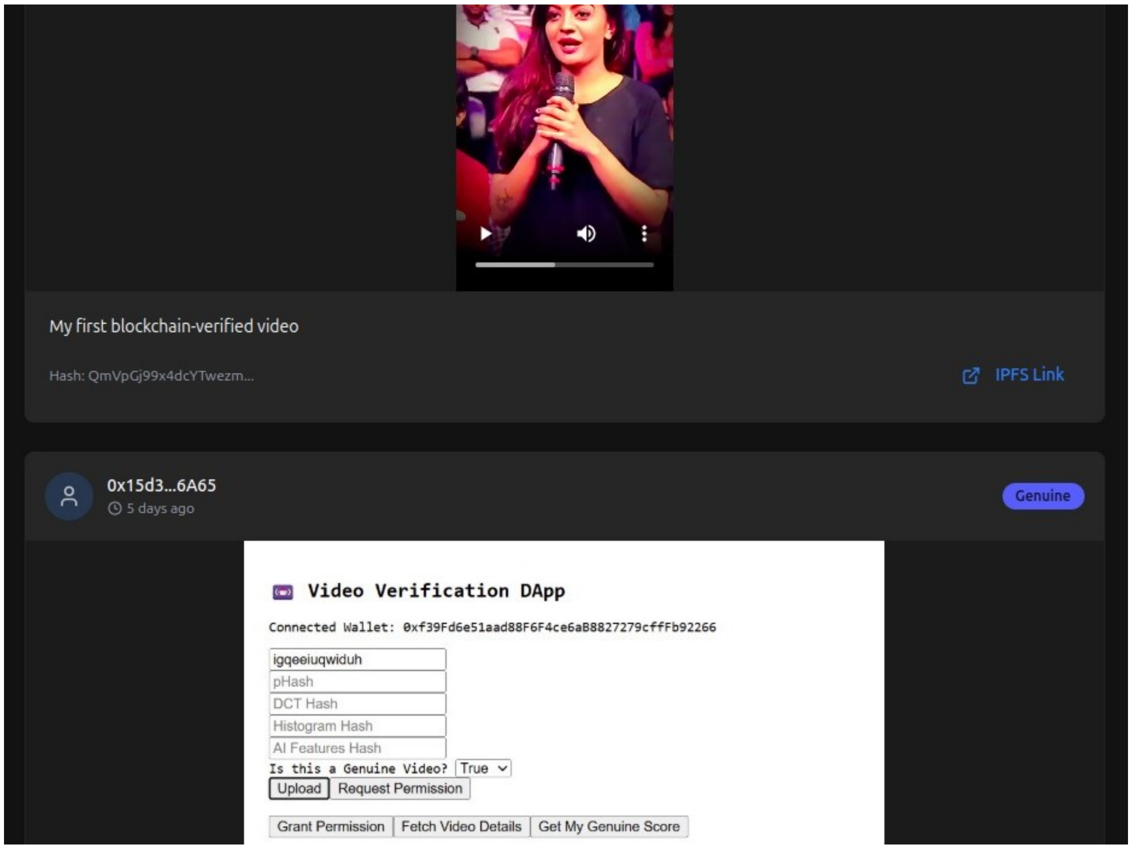
Perceptual Hash	DCT Hash
ph_Qmbj1n9d_00527	dct_Qmbj1n9_00527

Histogram Hash	AI Features Hash
hist_Qmbj1n_00527	ai_Qmbj1n9dm_00527

This image is genuine

Mark this content as original and authentic

Figure 5: React frontend when uploading the media


My first blockchain-verified video

Hash: QmVpGj99x4dcYTwezm...

0x15d3...6A65

5 days ago

Video Verification DApp

Connected Wallet: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266

igqeeiuqwiduh

pHash

DCT Hash

Histogram Hash

AI Features Hash

Is this a Genuine Video? True

Figure 6: Feed showing previously uploaded verified videos

8 JSON Data Format

`latest_upload.json` example:

```
{
  "filename": "cat.mp4",
  "phash_list": [...],
  "dct_list": [...],
  "hist_list": [...],
  "ai_feature_path": "static/features/cat.pkl",
  "duplicate_flag": 1,
  "best_match": {
    "matched_video": "lion.mp4",
    "phash_percent": 91.3,
    "dct_percent": 89.4,
    "hist_percent": 87.2,
    "ai_match_percent": 93.1
  }
}
```

9 Edge Cases Handled Effectively

Our system is not only capable of identifying exact matches or fully synthetic videos but is also robust against many real-world edge cases. Here's how:

1. Exact Reuploads and Minor Modifications

Scenario: A user uploads the same video with slight edits (cropping, re-encoding, noise).
Detection:

- `pHash`: Matches almost exactly.
- `DCT Hash`: Captures compression artifacts.
- `ResNet50 features`: Remain similar.

Action: Video is flagged as duplicate and blocked.

2. Partial Deepfakes in Long Videos

Scenario: Only a few seconds of deepfake in an otherwise real video.
Detection:

- Frame-wise hashing highlights altered frames.
- Deep features flag semantic inconsistencies.

Action: Flag only deepfake frames, alert uploader.

3. Color-Based Manipulation (e.g., Skin Tone Changes)

Scenario: Altering color spectrum to bypass visual detection.
Detection:

- `Histogram Hash`: Sensitive to color tone changes.
- `DCT Hash`: Detects low-frequency distortions.

Action: Flag the video for review if histogram anomaly is high.

4. Same Scene Filmed by Different Users

Scenario: Multiple users record the same event from different angles.
Avoiding False Positives:

- `pHash`: May match partially.
- `DCT + AI features`: Differ clearly.

Action: Not flagged as duplicate unless at least two hash types match.

5. Deepfake Tools That Modify Only Faces

Scenario: Faces are manipulated, but backgrounds remain identical.
Detection:

- `DCT Hash`: Detects facial distortions.
- `ResNet50`: Highlights semantic facial changes.

Action: Flagged as potential deepfake.

6. Video Compression as a Bypass Method

Scenario: A genuine video is compressed and re-uploaded.
Detection:

- `pHash + DCT Hash`: Minor differences, still highly similar.

Action: Allowed if variation is minor, not flagged as deepfake.

7. Video Splicing (Combining Multiple Sources)

Scenario: Real and deepfake segments combined in one file.
Detection:

- `Frame-wise Hashing`: Detects inconsistent segments.
- `Histogram`: Highlights visual mismatches.

Action: Only fake segments are flagged; user notified.

10 Key Innovations

- Multi-modal verification before publishing media
- Local + blockchain storage combo for speed and trust
- Reusability and ownership governance using smart contract

11 Scope for Expansion

- Integrate IPFS for decentralized media storage
- Replace SQLite with decentralized DB
- Add live AI-based deepfake classifier (GAN detection)
- Token-based reward/penalty system

12 Conclusion

This project bridges the gap between blockchain and AI for responsible media verification. It lays a foundation for scalable, tamper-proof systems capable of halting deepfake spread at the source.

References

- FaceForensics++ Dataset (R"ossler et al.)
- Solidity Documentation – Ethereum.org
- PyTorch ResNet Models
- MetaMask Web3 Integration Guide