Blockchain-Based Land Registry Verification System for Tanzania

IS 336 Project

May 25, 2025

1 Problem Statement

Land disputes in Tanzania, driven by informal ownership records, cause legal and financial issues. This project proposes a blockchain-based land registry system using Django, allowing government officials to register ownership records securely and citizens to verify them via SMS or a web portal. Stakeholders include government officials, landowners, and local authorities. Risks include fraud, disputed ownership, and data breaches.

2 Asset Valuation and Risk Assessment

Assets:

- **Land Records**: High confidentiality, high integrity, medium availability.
- **Servers/SMS Gateway**: Medium confidentiality, high integrity, high availability.
- **Verification Portal**: Low confidentiality, high integrity, high availability.

Risk Assessment:

- **STRIDE**: Spoofing (fake owners), Tampering (record alteration), Information Disclosure.
- **OWASP Top 10**: A01 (Broken Access Control), A03 (Injection).
- **Risk Matrix**:
  - High Likelihood/High Impact: Land Records.
  - Medium Likelihood/High Impact: Servers.
- **CVSS**: Tampering score = 8.2 (high integrity impact).

3 Security Principles and IAM

Principles:

- **Least Privilege**: Only officials register records.

- **Defense-in-Depth**: Blockchain, MFA, HTTPS.
- **Separation of Duties**: Registrars vs. verifiers.

IAM:

- **RBAC**: Admin (officials), Public (verification).
- **MFA**: SMS-based OTP (simulated).
- **Authentication**: Linked to NIDA IDs.

4 Cryptography and Secure Communication

- **Data at Rest**: SHA-256 hashing for records, stored in blockchain.
- **Data in Transit**: HTTPS for web; SMS encryption (simulated).
- **Digital Signatures**: Optional via `cryptography` library.
- **Blockchain**: In-memory ledger (scalable to Hyperledger).

5 Governance and Compliance

- **Framework**: Tanzanias Data Protection Act, NIST Cybersecurity Framework.
- **Measures**: Audit logs, data minimization, SMS consent.

6 Solution Architecture

Diagram: Citizen → SMS Gateway/Web Portal (HTTPS) → Django App → Blockchain → SQLite.
Tech Stack: Django, Python, SQLite, AWS Free Tier, Twilio (simulated).
Timeline:

- Month 1: Proof of concept.
- Month 23: Pilot in one district.
- Month 46: Scale to regions.

7 Prototype

A Django app simulates land record registration and verification. Admins register records via `/register/`, stored in a blockchain-like ledger and SQLite. Citizens verify via `/verify/`. SMS input is simulated via web forms.

7.1 Key Code: views.py

```python
import hashlib
import json
import time
```

```
from django.shortcuts import render, redirect
from django.contrib.auth.decorators import login_required
from django.contrib import messages
from .models import LandRecord

class Blockchain:
    def __init__(self):
        self.chain = []
        self.create_block(proof=1, previous_hash='0')

    def create_block(self, proof, previous_hash):
        block = {
            'index': len(self.chain) + 1,
            'timestamp': time.time(),
            'proof': proof,
            'previous_hash': previous_hash,
            'records': []
        }
        self.chain.append(block)
        return block

    def get_previous_block(self):
        return self.chain[-1]

    def hash_block(self, block):
        encoded_block = json.dumps(block, sort_keys=True).encode
           ()
        return hashlib.sha256(encoded_block).hexdigest()

    def add_record(self, owner_id, plot_details, issuer):
        record_data = f"{owner_id}:{plot_details}"
        record_hash = hashlib.sha256(record_data.encode()).
           hexdigest()
        block = self.get_previous_block()
        block['records'].append({
            'owner_id': owner_id,
            'plot_details': plot_details,
            'record_hash': record_hash,
            'issuer': issuer
        })
        proof = 1
        previous_hash = self.hash_block(block)
        new_block = self.create_block(proof, previous_hash)
        return new_block, record_hash

    def verify_record(self, owner_id, plot_details):
        record_data = f"{owner_id}:{plot_details}"
        record_hash = hashlib.sha256(record_data.encode()).
           hexdigest()
        for block in self.chain:
            for record in block['records']:
                if record['owner_id'] == owner_id and record['record_hash'] == record_hash:
                    return True, "Record␣is␣valid"
        return False, "Record␣not␣found␣or␣invalid"

blockchain = Blockchain()

@login_required
def register_land(request):
    if request.method == 'POST':
        owner_id = request.POST.get('owner_id')
        plot_details = request.POST.get('plot_details')
        if owner_id and plot_details:
            block, record_hash = blockchain.add_record(owner_id, plot_details, request.user.username)
            LandRecord.objects.create(
                owner_id=owner_id,
                plot_details=plot_details,
                record_hash=record_hash,
                issuer=request.user
            )
            messages.success(request, f"Land␣record␣registered␣in␣block␣{block['index']}")
            return redirect('register_land')
        messages.error(request, "Missing␣owner␣ID␣or␣plot␣details")
    return render(request, 'register_land.html')

def verify_land(request):
    if request.method == 'POST':
        owner_id = request.POST.get('owner_id')
        plot_details = request.POST.get('plot_details')
        if owner_id and plot_details:
            is_valid, message = blockchain.verify_record(owner_id, plot_details)
            messages.info(request, message)
        else:
            messages.error(request, "Missing␣owner␣ID␣or␣plot␣details")
    return render(request, 'verify_land.html')
```

8 Industry Integration

Platform: AWS Free Tier (Cognito for IAM, S3 for backups, RDS for database).
Benefits: Scalable, aligns with e-Government (eGA) standards.
Limitations: Rural connectivity; mitigated by SMS and QR codes.

9 Reflection

Skills: Django development, blockchain concepts, SMS integration.
TryHackMe: Blockchain Security, OWASP Top 10.
Ethics: Privacy (owner data protection), equity (rural access via SMS).