Blockchain-Based Autonomous Notarization System

# 1. Project Overview

This project implements a blockchain-based notarization system to automate fixed-date notarizations using smart contracts on the Ethereum blockchain. It integrates national eID cards equipped with Public Key Infrastructure (PKI) for secure and reliable identity verification.

# 2. Workflow Explanation

The system works in 4 main steps:  
1. \*\*Public Key Registration\*\*: The government agency registers the user's eID public key in the Public Key Management Smart Contract (SC).  
2. \*\*Digital Signature Creation\*\*: The client generates a digital signature for the document using their eID card and PIN.  
3. \*\*Signature Verification\*\*: The Signature Verification SC validates the digital signature and records the notarization details on the blockchain.  
4. \*\*Verification\*\*: A third-party verifier confirms document authenticity and integrity using blockchain data.

# 3. Diagram Explanation

The provided diagram highlights the interaction between the components:  
- \*\*Government Agency\*\*: Registers public keys into the Public Key Management SC.  
- \*\*Client\*\*: Generates a digital signature using their eID card.  
- \*\*Verifier\*\*: Validates documents using blockchain transaction data.  
- \*\*Smart Contracts\*\*: Manage public keys and verify digital signatures.  
  
The two main smart contracts are:  
- \*\*Public Key Management SC\*\*: Registers and deletes public keys.  
- \*\*Signature Verification SC\*\*: Validates digital signatures and logs notarization timestamps.

# 4. Algorithms

### Public Key Registration  
\*\*Inputs\*\*: Public Key (P\_key), Government Authentication (G\_auth)  
\*\*Outputs\*\*: Registration status  
  
1. Government agency verifies its identity.  
2. Call `registerPublicKey()` to add the user's public key.  
3. Validate the public key is stored successfully.  
  
### Digital Signature Creation  
\*\*Inputs\*\*: Document (D), User PIN, eID Private Key (K\_private)  
\*\*Outputs\*\*: Digital Signature (S), Document Hash (H(D))  
  
1. Generate document hash: H(D) = SHA256(D).  
2. Sign the hash with the eID private key.  
3. Send the signature and hash to the Signature Verification SC.  
  
### Signature Verification  
\*\*Inputs\*\*: Document Hash (H(D)), Digital Signature (S), Public Key (P\_key)  
\*\*Outputs\*\*: Verification Status, Transaction Receipt  
  
1. Validate the public key with the Public Key Management SC.  
2. Verify the signature using ECDSA recovery.  
3. Log the notarization on the blockchain and emit a transaction receipt.

# 5. Smart Contract Methods

### Public Key Management SC  
- `registerPublicKey(address \_publicKey)`: Adds a public key.  
- `isRegistered(address \_publicKey)`: Checks if a key is registered.  
  
### Signature Verification SC  
- `verifySignature(bytes32 documentHash, bytes memory signature, address publicKey)`: Verifies the signature and logs notarization.  
- `getTimestamp(bytes32 documentHash)`: Retrieves notarization time.

# 6. Advantages

1. Eliminates dependency on third-party notary services.  
2. Ensures tamper-proof notarization using blockchain.  
3. Enhances document authenticity and traceability.  
4. Secure identity verification using PKI and eID cards.  
5. Transparent and decentralized verification.