DAFTAR PUSTAKA

Antonopoulos, A. M., & Wood, G. (2018). *Mastering ethereum: building smart contracts and dapps*. O'reilly Media.

Ardipandanto, A. (2019). Permasalahan Penyelenggaraan Pemilu Serentak Tahun 2019. *Jurnal Ilmu Pemerintahan*, *11*(11), 25-30.

Arpialim, F. (2020). PENERAPAN BLOCKCHAIN DENGAN INTEGRASI SMART CONTRACT PADA SISTEM CROWDFUNDING. MAKASSAR; UNIVERSITAS HASANUDDIN.

Dib, O., Brousmiche, K. L., Durand, A., Thea, E., & Hamida, E. B. (2018). Consortium *blockchain*s: Overview, applications and challenges. *Int. J. Adv. Telecommun*, *11*(1), 51-64

Ethereum. (2022, Agustus 16). https://ethereum.org/id/developers/docs/*block*s/

Ethereum. (2023, Maret 15). https://ethereum.org/en/developers/docs/transactions/

Gao, W., Hatcher, W. G., & Yu, W. (2018, July). A survey of *blockchain*: Techniques, applications, and challenges. In *2018 27th international conference on computer communication and networks (ICCCN)* (pp. 1-11). IEEE.

Habibi, M. (2018). Dinamika Implementasi E-Voting di Berbagai Negara.

Hu, S. D. K., Palit, H. N., & Handojo, A. (2019). Implementasi *Blockchain*: Studi Kasus e-Voting. *Jurnal Infra*, *7*(1), 183-189.

Jani, S. (2020). Smart contracts: Building *block*s for digital transformation. *Indira Gandhi National Open University*.

Liu, M., Wu, K., & Xu, J. J. (2019). How will *blockchain* technology impact auditing and accounting: Permissionless versus permissioned *blockchain*. *Current Issues in auditing*, *13*(2), A19-A29.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, 21260.

Pawar, D., Sarode, P., Santpure, S., Thore, P., & Nimbalkar, P. Implementation of Secure Voting System using *Blockchain*.

Pratama, Y. W., & Kurniadi, D. (2021). Implementasi *Blockchain* dalam Aplikasi Pemilu. *Incare, International Journal of Educational Resources*, *2*(2), 242-254.

Rahardja, U., Mulyati, M., & Budiarty, F. (2021). Pengaruh Teknologi *blockchain* TERHADAP Keabsahan Ijazah. *Jurnal Manajemen Retail Indonesia*, *2*(1), 11-19. doi:10.33050/jmari.v2i1.1428

Sayeed, S., Marco-Gisbert, H., & Caira, T. (2020). Smart contract: Attacks and protections. *IEEE Access*, *8*, 24416-24427.

Setia, T. E. H., & Susanto, A. (2019). Smart Contract *Blockchain* pada E-Voting. *Jurnal Informatika Upgris*, *5*(2).

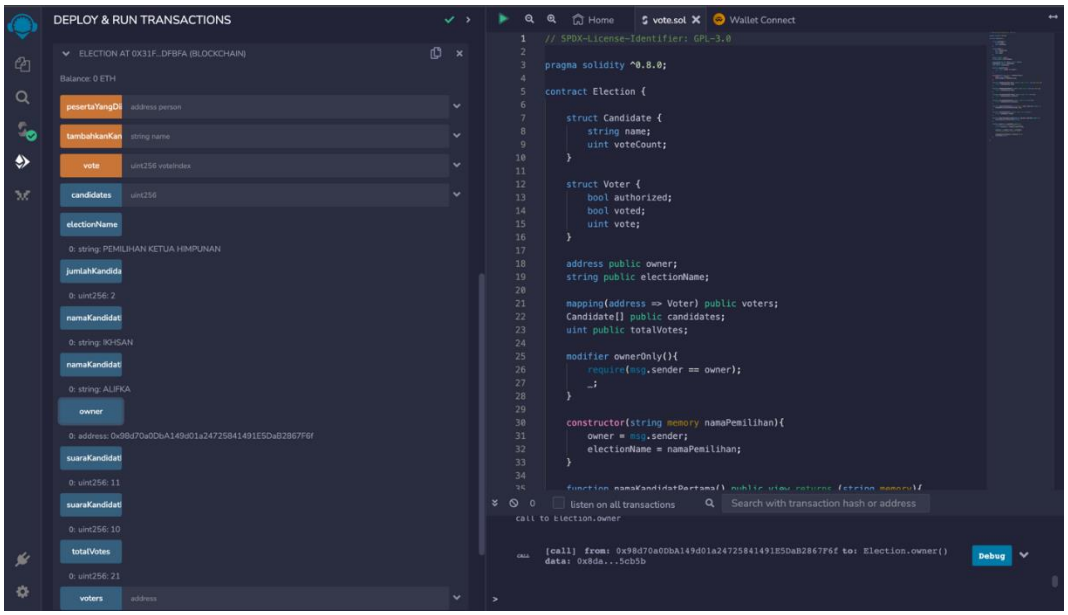
Vujičić, D., Jagodić, D., & Ranđić, S. (2018, March). *Blockchain* technology, bitcoin, and Ethereum: A brief overview. In *2018 17th international symposium infoteh-jahorina (infoteh)* (pp. 1-6). IEEE.

Wijaya, D. A., & Darmawan, O. (2018). *Blockchain* Dari Bitcoin Untuk Dunia. *Jakarta: Jasakom*.

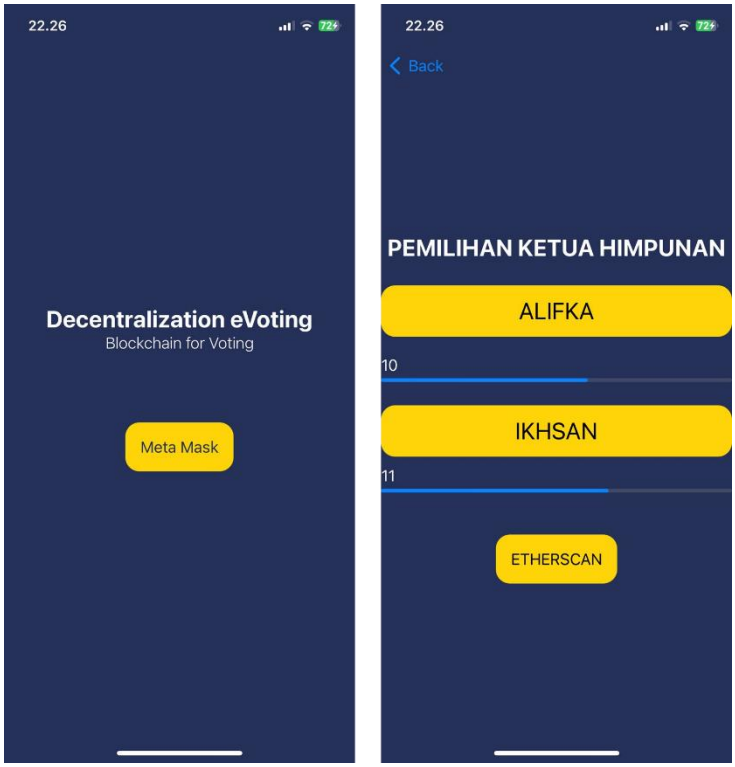
Wood, G. (2022). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, *151*(2022), 1-41.

Lampiran 1

- Tampilan halaman untuk Penyelenggara



- Tampilan halaman *login* dan *voting* untuk pemilih



Lampiran 2

- Kode *SmartContract* sebelum dianalisis *vulnerability*

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.0;

contract Election {

    struct Candidate {
        string name;
        uint voteCount;
    }

    struct Voter {
        bool authorized;
        bool voted;
        uint vote;
    }

    address public owner;
    string public electionName;

    mapping(address => Voter) public voters;
    Candidate[] public candidates;
    uint public totalVotes;

    modifier ownerOnly(){
        require(msg.sender == owner);
        _;
    }

    constructor(string memory namaPemilihan){
        owner = msg.sender;
        electionName = namaPemilihan;
    }

    function namaKandidatPertama() public view returns (string memory){
        return candidates[0].name;
    }

    function namaKandidatKedua() public view returns (string memory){
        return candidates[1].name;
    }

    function suaraKandidatPertama() public view returns(uint){
        return candidates[0].voteCount;
    }
```

```
    function suaraKandidatKedua() public view returns(uint){
        return candidates[1].voteCount;
    }

    function tambahkanKandidat(string memory name) ownerOnly public {
        candidates.push(Candidate(name, 0));
    }

    function jumlahKandidat() public view returns(uint) {
        return candidates.length;
    }

    function pesertaYangDiizinkan(address person) ownerOnly public {
        voters[person].authorized = true;
    }

    function vote(uint voteIndex) public {
        //require(!voters[msg.sender].voted);
        require(voters[msg.sender].authorized);

        voters[msg.sender].vote = voteIndex;
        //voters[msg.sender].voted = true;

        candidates[voteIndex].voteCount += 1;
        totalVotes += 1;
    }

}
```

- Kode *SmartContract* Setelah dilakukan analisis *vulnerability*

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.0;

contract Election {
    struct Candidate {
        string name;
        uint voteCount;
    }

    struct Voter {
        bool authorized;
        bool voted;
        uint vote;
    }

    address public owner;
    string public electionName;

    mapping(address => Voter) public voters;
```

```
    Candidate[] internal candidates;
    uint public totalVotes;

    modifier ownerOnly() {
        require(msg.sender == owner);
        _;
    }

    constructor(string memory namaPemilihan) {
        owner = msg.sender;
        electionName = namaPemilihan;
    }

    function namaKandidatPertama() public view returns (string memory) {
        return candidates[0].name;
    }

    function namaKandidatKedua() public view returns (string memory) {
        return candidates[1].name;
    }

    function suaraKandidatPertama() public view returns (uint) {
        return candidates[0].voteCount;
    }

    function suaraKandidatKedua() public view returns (uint) {
        return candidates[1].voteCount;
    }

    function tambahkanKandidat(string memory name) public ownerOnly {
        candidates.push(Candidate(name, 0));
    }

    function jumlahKandidat() public view returns (uint) {
        return candidates.length;
    }

    function pesertaYangDiizinkan(address person) public ownerOnly {
        voters[person].authorized = true;
    }

    function vote(uint voteIndex) public {
        //require(!voters[msg.sender].voted);
        require(voters[msg.sender].authorized);

        voters[msg.sender].vote = voteIndex;
        //voters[msg.sender].voted = true;

        candidates[voteIndex].voteCount += 1;
        totalVotes += 1;
    }
}
```

Lampiran 3

- Kode untuk memanggil *function* pada *smartcontract* menggunakan library Web3.Swift

```
import Web3

class SmartContractInteraction: ObservableObject {

    @Published var judul: String = ""
    @Published var namaKandidatPertama: String = ""
    @Published var namaKandidatKedua: String = ""
    @Published var suaraKandidatPertama: BigUInt = 0
    @Published var suaraKandidatKedua: BigUInt = 0

    func callSmartContract() {

        let web3 = Web3(rpcURL: "https://goerli.infura.io/v3/049172870f3942feb27cd4e6dcd933ab")

        do {
            // define contract
            let contractABI = """
                [
                    {
                        "inputs": [
                            {
                                "internalType": "address",
                                "name": "person",
                                "type": "address"
                            }
                        ],
                        "name": "pesertaYangDiizinkan",
                        "outputs": [],
                        "stateMutability": "nonpayable",
                        "type": "function"
                    },
                    {
                        "inputs": [
                            {
                                "internalType": "string",
                                "name": "name",
                                "type": "string"
                            }
                        ],
                        "name": "tambahkanKandidat",
                        "outputs": [],
                        "stateMutability": "nonpayable",
                        "type": "function"
                    },
                    {
```

```
        "inputs": [
            {
                "internalType": "string",
                "name": "namaPemilihan",
                "type": "string"
            }
        ],
        "stateMutability": "nonpayable",
        "type": "constructor"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "voteIndex",
                "type": "uint256"
            }
        ],
        "name": "vote",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "name": "candidates",
        "outputs": [
            {
                "internalType": "string",
                "name": "name",
                "type": "string"
            },
            {
                "internalType": "uint256",
                "name": "voteCount",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "electionName",
        "outputs": [
            {
```

```
                "internalType": "string",
                "name": "",
                "type": "string"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "jumlahKandidat",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "namaKandidatKedua",
        "outputs": [
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "namaKandidatPertama",
        "outputs": [
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "owner",
        "outputs": [
            {
```

```
                "internalType": "address",
                "name": "",
                "type": "address"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "suaraKandidatKedua",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "suaraKandidatPertama",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "totalVotes",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "address",
                "name": "",
```

```
                    "type": "address"
                }
            ],
            "name": "voters",
            "outputs": [
                {
                    "internalType": "bool",
                    "name": "authorized",
                    "type": "bool"
                },
                {
                    "internalType": "bool",
                    "name": "voted",
                    "type": "bool"
                },
                {
                    "internalType": "uint256",
                    "name": "vote",
                    "type": "uint256"
                }
            ],
            "stateMutability": "view",
            "type": "function"
        }
    ]
    """.data(using: .utf8)!
    let contract = try web3.eth.Contract(json: contractABI, abiKey: nil, address: EthereumAddress(ethereumValue: "0x31F0a00a455Bc377f4b92D8650C9A07C371dFbfA"))

    //try call function "electionName" in the contract
    contract["electionName"]?().call() { response, error in
        if let response = response,
           let message = response[""] as? String {
            DispatchQueue.main.async {
                self.judul = message.description
            }
        }
        else {
            print(error?.localizedDescription ?? "Failed to get response")
        }
    }

    //try call function namaKandidatPertama from contract
    contract["namaKandidatPertama"]?().call() { response, error in
        if let response = response,
           let message = response[""] as? String {
            DispatchQueue.main.async {
                self.namaKandidatPertama = message.description
            }

        }
        else {
```

```
                print(error?.localizedDescription ?? "Failed to get response")
            }
        }

        //try call function namaKandidatKedua from contract
        contract["namaKandidatKedua"]?().call() { response, error in
            if let response = response,
               let message = response[""] as? String {
                DispatchQueue.main.async {
                    self.namaKandidatKedua = message.description
                }

            }
            else {
                print(error?.localizedDescription ?? "Failed to get response")
            }
        }

        contract["suaraKandidatPertama"]?().call() { response, error in
            if let response = response,
               let message = response[""] as? BigUInt {
                DispatchQueue.main.async {
                    self.suaraKandidatPertama = message
                }

            }
            else {
                print(error?.localizedDescription ?? "Failed to get response")
            }
        }

        contract["suaraKandidatKedua"]?().call() { response, error in
            if let response = response,
               let message = response[""] as? BigUInt {
                DispatchQueue.main.async {
                    self.suaraKandidatKedua = message
                }

            }
            else {
                print(error?.localizedDescription ?? "Failed to get response")
            }
        }


    } catch {
        // Error handling
        print(error.localizedDescription)
    }

  }
}
```

- Kode untuk melakukan transaksi atau voting pada kandidat pertama

```
import Web3

class SendTransaction0: ObservableObject {

    @Published var hash: String = ""

    func writeSmartContract0() {

        let web3 = Web3(rpcURL: "https://goerli.infura.io/v3/049172870f3942feb27cd4e6dcd933ab")

        do {
            let account = try! EthereumPrivateKey(hexPrivateKey:
"0x75f44eeb64f1802773eb6595062feb5f6e2063b6356a08b265de1c17a66ce050")


            let contractABI = """
                        [
                            {
                                "inputs": [
                                    {
                                        "internalType": "address",
                                        "name": "person",
                                        "type": "address"
                                    }
                                ],
                                "name": "pesertaYangDiizinkan",
                                "outputs": [],
                                "stateMutability": "nonpayable",
                                "type": "function"
                            },
                            {
                                "inputs": [
                                    {
                                        "internalType": "string",
                                        "name": "name",
                                        "type": "string"
                                    }
                                ],
                                "name": "tambahkanKandidat",
                                "outputs": [],
                                "stateMutability": "nonpayable",
                                "type": "function"
                            },
                            {
                                "inputs": [
                                    {
                                        "internalType": "string",
                                        "name": "namaPemilihan",
```

```
            "type": "string"
        }
    ],
    "stateMutability": "nonpayable",
    "type": "constructor"
},
{
    "inputs": [
        {
            "internalType": "uint256",
            "name": "voteIndex",
            "type": "uint256"
        }
    ],
    "name": "vote",
    "outputs": [],
    "stateMutability": "nonpayable",
    "type": "function"
},
{
    "inputs": [
        {
            "internalType": "uint256",
            "name": "",
            "type": "uint256"
        }
    ],
    "name": "candidates",
    "outputs": [
        {
            "internalType": "string",
            "name": "name",
            "type": "string"
        },
        {
            "internalType": "uint256",
            "name": "voteCount",
            "type": "uint256"
        }
    ],
    "stateMutability": "view",
    "type": "function"
},
{
    "inputs": [],
    "name": "electionName",
    "outputs": [
        {
            "internalType": "string",
            "name": "",
            "type": "string"
        }
```

```
    ],
    "stateMutability": "view",
    "type": "function"
},
{
    "inputs": [],
    "name": "jumlahKandidat",
    "outputs": [
        {
            "internalType": "uint256",
            "name": "",
            "type": "uint256"
        }
    ],
    "stateMutability": "view",
    "type": "function"
},
{
    "inputs": [],
    "name": "namaKandidatKedua",
    "outputs": [
        {
            "internalType": "string",
            "name": "",
            "type": "string"
        }
    ],
    "stateMutability": "view",
    "type": "function"
},
{
    "inputs": [],
    "name": "namaKandidatPertama",
    "outputs": [
        {
            "internalType": "string",
            "name": "",
            "type": "string"
        }
    ],
    "stateMutability": "view",
    "type": "function"
},
{
    "inputs": [],
    "name": "owner",
    "outputs": [
        {
            "internalType": "address",
            "name": "",
            "type": "address"
        }
```

```
    ],
    "stateMutability": "view",
    "type": "function"
},
{
    "inputs": [],
    "name": "suaraKandidatKedua",
    "outputs": [
        {
            "internalType": "uint256",
            "name": "",
            "type": "uint256"
        }
    ],
    "stateMutability": "view",
    "type": "function"
},
{
    "inputs": [],
    "name": "suaraKandidatPertama",
    "outputs": [
        {
            "internalType": "uint256",
            "name": "",
            "type": "uint256"
        }
    ],
    "stateMutability": "view",
    "type": "function"
},
{
    "inputs": [],
    "name": "totalVotes",
    "outputs": [
        {
            "internalType": "uint256",
            "name": "",
            "type": "uint256"
        }
    ],
    "stateMutability": "view",
    "type": "function"
},
{
    "inputs": [
        {
            "internalType": "address",
            "name": "",
            "type": "address"
        }
    ],
    "name": "voters",
```

```
                            "outputs": [
                                {
                                    "internalType": "bool",
                                    "name": "authorized",
                                    "type": "bool"
                                },
                                {
                                    "internalType": "bool",
                                    "name": "voted",
                                    "type": "bool"
                                },
                                {
                                    "internalType": "uint256",
                                    "name": "vote",
                                    "type": "uint256"
                                }
                            ],
                            "stateMutability": "view",
                            "type": "function"
                        }
                    ]
            """.data(using: .utf8)!
        let contract = try web3.eth.Contract(json: contractABI, abiKey: nil, address: EthereumAddress(ethereumValue: "0x31F0a00a455Bc377f4b92D8650C9A07C371dFbfA"))



        firstly {
            web3.eth.getTransactionCount(address: account.address, block: .latest)
        }.done { nonce in
            // Contract transaction
            guard let transaction = contract["vote"]?(BigUInt(0)).createTransaction(nonce: nonce, from: account.address, value: 0, gas: EthereumQuantity(150000), gasPrice: EthereumQuantity(quantity: 21.gwei)
            ) else{
                return
            }
            let signedTX = try transaction.sign(with: account, chainId: 5)

            firstly {
                web3.eth.sendRawTransaction(transaction: signedTX)
            }.done { txHash in
                self.hash = txHash.hex()
                print(txHash.hex())
            }.catch { error in
                print(error)
            }
        }.catch { error in
            print(error)
        }

    } catch {
```

```
            print("Is : ", error.localizedDescription)
        }
    }

}


extension EthereumTransaction{
    func signX(with privateKey: EthereumPrivateKey, chainId: Int = 0) throws ->
EthereumSignedTransaction {

        // These values are required for signing
        guard let nonce = nonce, let gasPrice = gasPrice, let gasLimit = gas, let value = value else {
            throw EthereumSignedTransaction.Error.transactionInvalid
        }

        let rlp = RLPItem(
            nonce: nonce,
            gasPrice: gasPrice,
            gasLimit: gasLimit,
            to: to,
            data: data,
            chainId: UInt(chainId)
        )

        let rawRlp = try RLPEncoder().encode(rlp)

        guard let signedTransaction = try? privateKey.sign(message: rawRlp) else { throw
EthereumSignedTransaction.Error.transactionInvalid }

        let v: BigUInt
        if chainId == 0 {
            v = BigUInt(signedTransaction.v) + BigUInt(27)
        } else {
            let sigV = BigUInt(signedTransaction.v)
            let big27 = BigUInt(27)
            let chainIdCalc = (BigUInt(chainId) * BigUInt(2) + BigUInt(8))
            v = sigV + big27 + chainIdCalc
        }

        return EthereumSignedTransaction(
            nonce: nonce,
            gasPrice: gasPrice,
            gasLimit: gasLimit,
            to: to,
            value: value,
            data: data,
            v: EthereumQuantity(quantity: v),
            r: EthereumQuantity(quantity: BigUInt(signedTransaction.r)),
            s: EthereumQuantity(quantity: BigUInt(signedTransaction.s)),
            chainId: EthereumQuantity(quantity: BigUInt(chainId))
        )
```

```
        }
}

extension RLPItem {
    /**
         * Create an RLPItem representing a transaction. The RLPItem must be an array of 6 items in the proper order.
         *
         * - parameter nonce: The nonce of this transaction.
         * - parameter gasPrice: The gas price for this transaction in wei.
         * - parameter gasLimit: The gas limit for this transaction.
         * - parameter to: The address of the receiver.
         * - parameter data: Input data for this transaction.
         */
        init(
            nonce: EthereumQuantity,
            gasPrice: EthereumQuantity,
            gasLimit: EthereumQuantity,
            to: EthereumAddress?,
            data: EthereumData,
            chainId: UInt
        ) {
            self = .array(
                .bigUInt(nonce.quantity),
                .bigUInt(gasPrice.quantity),
                .bigUInt(gasLimit.quantity),
                .bytes(to?.rawAddress ?? Bytes()),
                .string(""),
                .bytes(data.bytes),
                .init(integerLiteral: chainId),
                .init(integerLiteral: 0),
                .init(integerLiteral: 0)
            )
        }

}
```

- Kode untuk melakukan transaksi atau voting pada kandidat kedua

```
import Web3

class SendTransaction1: ObservableObject {

    @Published var hash: String = ""

    func writeSmartContract1() {

        let web3 = Web3(rpcURL: "https://goerli.infura.io/v3/049172870f3942feb27cd4e6dcd933ab")

        do {
```

```
        let account = try! EthereumPrivateKey(hexPrivateKey:
"0x75f44eeb64f1802773eb6595062feb5f6e2063b6356a08b265de1c17a66ce050")


        let contractABI = """
                        [
                            {
                                "inputs": [
                                    {
                                        "internalType": "address",
                                        "name": "person",
                                        "type": "address"
                                    }
                                ],
                                "name": "pesertaYangDiizinkan",
                                "outputs": [],
                                "stateMutability": "nonpayable",
                                "type": "function"
                            },
                            {
                                "inputs": [
                                    {
                                        "internalType": "string",
                                        "name": "name",
                                        "type": "string"
                                    }
                                ],
                                "name": "tambahkanKandidat",
                                "outputs": [],
                                "stateMutability": "nonpayable",
                                "type": "function"
                            },
                            {
                                "inputs": [
                                    {
                                        "internalType": "string",
                                        "name": "namaPemilihan",
                                        "type": "string"
                                    }
                                ],
                                "stateMutability": "nonpayable",
                                "type": "constructor"
                            },
                            {
                                "inputs": [
                                    {
                                        "internalType": "uint256",
                                        "name": "voteIndex",
                                        "type": "uint256"
                                    }
                                ],
                                "name": "vote",
```

```
    "outputs": [],
    "stateMutability": "nonpayable",
    "type": "function"
  },
  {
    "inputs": [
      {
        "internalType": "uint256",
        "name": "",
        "type": "uint256"
      }
    ],
    "name": "candidates",
    "outputs": [
      {
        "internalType": "string",
        "name": "name",
        "type": "string"
      },
      {
        "internalType": "uint256",
        "name": "voteCount",
        "type": "uint256"
      }
    ],
    "stateMutability": "view",
    "type": "function"
  },
  {
    "inputs": [],
    "name": "electionName",
    "outputs": [
      {
        "internalType": "string",
        "name": "",
        "type": "string"
      }
    ],
    "stateMutability": "view",
    "type": "function"
  },
  {
    "inputs": [],
    "name": "jumlahKandidat",
    "outputs": [
      {
        "internalType": "uint256",
        "name": "",
        "type": "uint256"
      }
    ],
    "stateMutability": "view",
```

```
        "type": "function"
    },
    {
        "inputs": [],
        "name": "namaKandidatKedua",
        "outputs": [
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "namaKandidatPertama",
        "outputs": [
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "owner",
        "outputs": [
            {
                "internalType": "address",
                "name": "",
                "type": "address"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "suaraKandidatKedua",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
```

```
        "type": "function"
    },
    {
        "inputs": [],
        "name": "suaraKandidatPertama",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "totalVotes",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "address",
                "name": "",
                "type": "address"
            }
        ],
        "name": "voters",
        "outputs": [
            {
                "internalType": "bool",
                "name": "authorized",
                "type": "bool"
            },
            {
                "internalType": "bool",
                "name": "voted",
                "type": "bool"
            },
            {
                "internalType": "uint256",
                "name": "vote",
                "type": "uint256"
```

```
                                }
                            ],
                            "stateMutability": "view",
                            "type": "function"
                        }
                    ]
            """.data(using: .utf8)!
        let contract = try web3.eth.Contract(json: contractABI, abiKey: nil, address: EthereumAddress(ethereumValue: "0x31F0a00a455Bc377f4b92D8650C9A07C371dFbfA"))



        firstly {
            web3.eth.getTransactionCount(address: account.address, block: .latest )
        }.done { nonce in
            // Contract transaction
            guard let transaction = contract["vote"]?(BigUInt(1)).createTransaction(nonce: nonce, from: account.address, value: 0, gas: EthereumQuantity(150000), gasPrice: EthereumQuantity(quantity: 21.gwei)
            ) else{
                return
            }
            let signedTX = try transaction.sign(with: account, chainId: 5)

            firstly {
                web3.eth.sendRawTransaction(transaction: signedTX)
            }.done { txHash in
                self.hash = txHash.hex()
                print(txHash.hex())
            }.catch { error in
                print(error)
            }
        }.catch { error in
            print(error)
        }

    } catch {
        print("Is : ", error.localizedDescription)
    }
  }

}
```

Lampiran 4

- Kode untuk menampilkan dan menghubungkan *wallet* dengan sistem *e-voting*

```
import SwiftUI
import Glaip


struct ContentView: View {

    @ObservedObject private var glaip = Glaip(title: "Dvote Prototype", description: "Dvote Login",
supportedWallets: [.MetaMask])

    @State private var showLoginScreen = false

    var body: some View {
        NavigationView {
            ZStack {
                Color(red: 38 / 255, green: 49 / 255, blue: 89 / 255)
                    .edgesIgnoringSafeArea(.all)
                VStack {
                    Text("Decentralization eVoting")
                        .font(.system(size: 25))
                        .fontWeight(.bold)
                        .foregroundColor(Color.white)
                    Text("Blockchain for Voting")
                        .fontWeight(.light)
                        .foregroundColor(Color.white)

                    Spacer()
                        .frame(height: 75)

                    Button("Meta Mask"){
                        glaip.loginUser(type: .MetaMask) { result in
                            switch result {
                            case .success(let user):
                                print(user.wallet.address)
                                showLoginScreen = true
                            case .failure(let error):
                                print(error)
                            }
                        }

                    }
                    .padding()
                    .foregroundColor(Color(red: 38 / 255, green: 49 / 255, blue: 89 / 255))

                    .background(Color.yellow)
                    .clipShape(RoundedRectangle(cornerRadius: 15))
```

```
                NavigationLink(destination: HellowView(), isActive: $showLoginScreen){
                    EmptyView()
                }
            }
        }
        .navigationBarHidden(true)
    }


}
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

- Kode untuk menampilkan *function* yang telah dipanggil dari *blockchain*

```
import SwiftUI

struct HellowView: View {

    @ObservedObject var smartContractInteraction = SmartContractInteraction()
    @ObservedObject var smartContractTransaction0 = SendTransaction0()
    @ObservedObject var smartContractTransaction1 = SendTransaction1()
    @State private var button1 = false
    @State private var button2 = false


    var body: some View {
        ZStack {
            Color(red: 38 / 255, green: 49 / 255, blue: 89 / 255)
                .edgesIgnoringSafeArea(.all)
            VStack{
                Group{
                    Text(smartContractInteraction.judul)
                        .foregroundColor(Color.white)
                        .fontWeight(.bold)
                        .font(.system(size: 25))

                    Spacer()
                        .frame(height: 25)
                }


                Group{

                    Button(smartContractInteraction.namaKandidatPertama){
```

```
                smartContractTransaction0.writeSmartContract0()
                smartContractInteraction.suaraKandidatPertama += 1
                button1 = true
            }
            .frame(height: 55)
            .frame(maxWidth: .infinity)
            .background(Color.yellow)
            .clipShape(RoundedRectangle(cornerRadius: 15))
            .foregroundColor(Color.black)
            .font(.system(size: 25))
            Spacer()
                .frame(height: 20)

            ProgressView(String(smartContractInteraction.suaraKandidatPertama)                ,value:
Double(smartContractInteraction.suaraKandidatPertama), total: 17)

            if button1 == true {
                Text(smartContractTransaction0.hash)
                    .contextMenu(ContextMenu(menuItems: {
                        Button("Copy"){
                            UIPasteboard.general.string = smartContractTransaction0.hash
                        }
                    }))

            }

            Spacer()
                .frame(height: 25)

            Button(smartContractInteraction.namaKandidatKedua){
                smartContractTransaction1.writeSmartContract1()
                smartContractInteraction.suaraKandidatKedua += 1
                button2 = true
            }
            .frame(height: 55)
            .frame(maxWidth: .infinity)
            .background(Color.yellow)
            .clipShape(RoundedRectangle(cornerRadius: 15))
            .foregroundColor(Color.black)
            .font(.system(size: 25))
            Spacer()
                .frame(height: 10)

            ProgressView(String(smartContractInteraction.suaraKandidatKedua),                value:
Double(smartContractInteraction.suaraKandidatKedua), total: 17)

            if button2 == true {
                Text(smartContractTransaction1.hash)
                    .contextMenu(ContextMenu(menuItems: {
                        Button("Copy"){
                            UIPasteboard.general.string = smartContractTransaction1.hash
                        }
```

```
                        }))
                    }
                }

                Group{
                    Spacer()
                        .frame(height: 45)

                    NavigationLink(destination:                                    EtherscanView(url:
"https://goerli.etherscan.io/address/0x31f0a00a455bc377f4b92d8650c9a07c371dfbfa")) {
                        Text("ETHERSCAN")
                            .padding()
                            .background(Color.yellow)
                            .cornerRadius(15)
                            .foregroundColor(Color.black)
                    }
                }
            }

            .onAppear{
                //self.smartContractTransaction0
                self.smartContractInteraction.callSmartContract()


            }
        }

    }

}


struct HellowView_Previews: PreviewProvider {
    static var previews: some View {
        HellowView()
    }
}
```

- Kode untuk menampilkan Etherscan (*ledger*)

```
import SwiftUI
import WebKit

struct EtherscanView: View {

    let url: String

    var body: some View {
        WebView(urlString: url)
    }
}
```

```
struct EtherscanView_Previews: PreviewProvider {
    static var previews: some View {
        EtherscanView(url: "https://www.google.com")
    }
}

struct WebView: UIViewRepresentable {

    let urlString: String?

    func makeUIView(context: Context) -> WebView.UIViewType {
        return WKWebView()
    }

    func updateUIView(_ uiView: WKWebView, context: Context) {
        if let safeString = urlString {
            if let url = URL(string: safeString) {
                let request = URLRequest(url: url)
                uiView.load(request)
            }
        }
    }
}
```

- Lampiran Lembar Perbaikan Skripsi

LEMBAR PERBAIKAN SKRIPSI

"IMPLEMENTASI *BLOCKCHAIN* PADA SISTEM *E-VOTING* BERBASIS *MOBILE*"

OLEH:

IRFANDI KURNIAWAN ANWAR
D121171504

Skripsi ini telah dipertahankan pada Ujian Akhir Sarjana tanggal 1 Agustus 2023.

Telah dilakukan perbaikan penulisan dan isi skripsi berdasarkan usulan dari penguji dan pembimbing skripsi.

Persetujuan perbaikan oleh tim penguji:

	Nama	Tanda Tangan
Ketua	Dr. Eng. Ady Wahyudi Paundu, S.T., M.T.	
Sekretaris	Adnan, S.T., M.T., Ph. D.	
Anggota	Dr. Amil Ahmad Ilham, S.T., M.IT.	
	Mukarramah Yusuf, B.Sc., M.Sc., Ph. D.	

Persetujuan Perbaikan oleh pembimbing:

Pembimbing	Nama	Tanda Tangan
I	Dr. Eng. Ady Wahyudi Paundu, S.T., M.T.	
II	Adnan, S.T., M.T., Ph. D.	