BlockSigner, uma Malha de Confiança para Documentos

—

Acadêmicos: Gabriel Marques, Murilo B. Flor
Orientador: Prof. Brivaldo Junior

Sumário

- Motivação
- Fundamentação Teórica
- Trabalhos Relacionados
- Implementação
- Conclusão
- Referências

Motivação

Evolução dos mecanismos criptográficos

- DES, 3DES, RSA.
- Apenas chaves são utilizadas para a validação (insuficiente).

Estagnação no lado servidor

- Senhas armazenadas em *hashes* criptográficos.
- Servidor recebe o par (*hash*, usuário) e autentica.

Falhas de segurança

- Utilização mal intencionada das informações dos usuários.
- Informação local do servidor passível de mudança.

Motivação

Resolvendo as falhas de segurança

- Histórico compartilhado dos dados de forma distribuída.
- Nova assinatura fraudulenta será recusada, e facilmente notada pelo usuário.
- Alterações em transações passadas facilmente detectada por outros nós.

Assinaturas digitais

- Grande parte dos arquivos em diversas áreas do mercado migraram para o ambiente digital.
- Necessidade de validar esse volume de informações.

Aplicando o *Blockchain*

- Otimiza a segurança do sistema.
- Alternativa eficaz para garantir a legalidade de transações e a segurança de instituições.
- Fuga do uso padrão da recente tecnologia.

Fundamentação Teórica

- ***Blockchain* (Cadeia de Blocos)**
  - **1. Transparência**
    - É possível ter a visualização de qualquer transação.
  - **2. Descentralização**
    - Não há necessidade de um órgão intermediário que aprove a transação ou que determine certos regulamentos de contrato.
  - **3. Segurança**
    - O banco de dados é imutável, em outra palavras, consiste em um registro que não pode ser alterado, revisado ou adulterado, nem mesmo para aqueles que operam o banco de dados.
  - **4. Consenso**
    - A validação de uma transação requer que outros computadores de outros participantes entrem em um consenso para possibilitar que essa transação ocorra.
  - **5. Automatizado**
    - O software foi desenvolvido para que não haja intereação humana, sendo assim, transações podem ser realizadas a qualquer momento..

Fundamentação Teórica

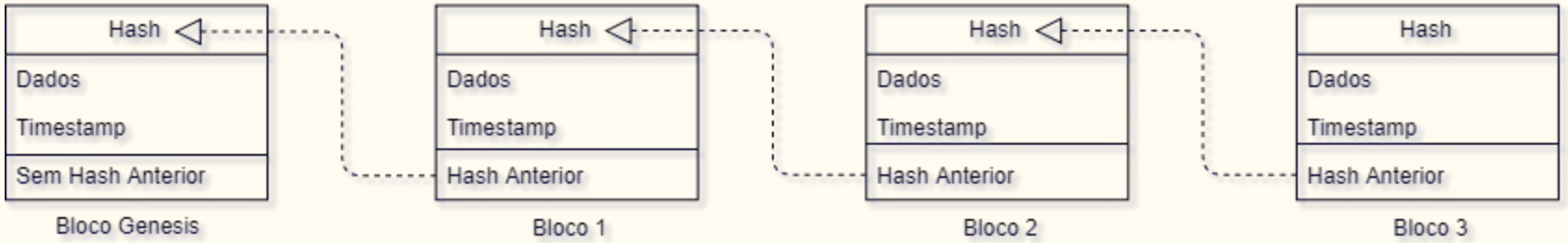


Figura 1: Blockchain (Cadeia de Blocos)

Fundamentação Teórica

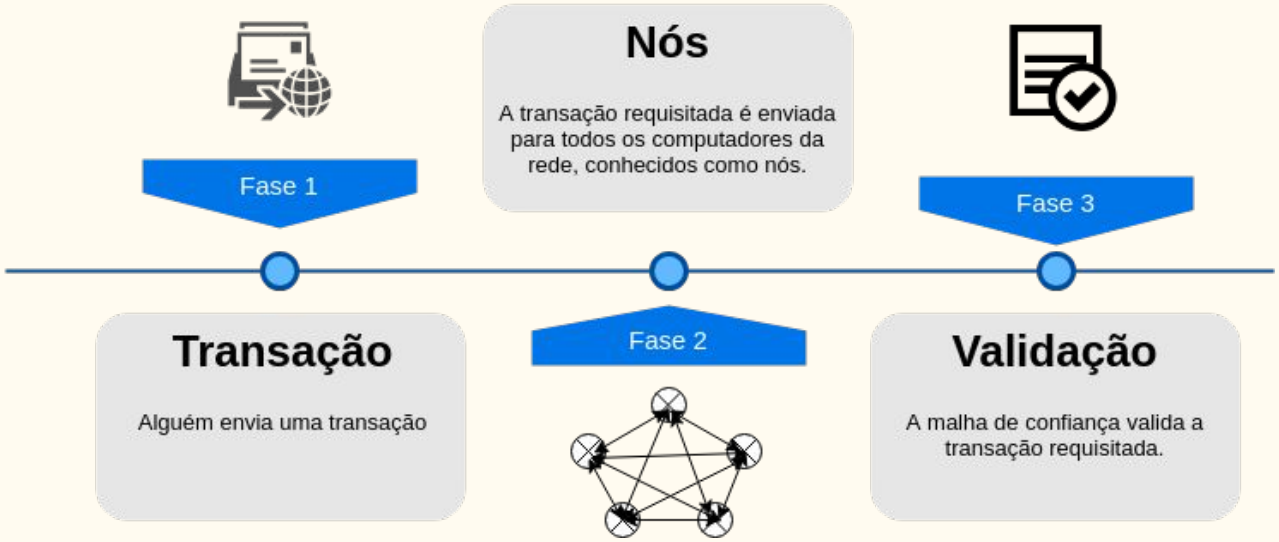


Figura 2: Ciclo de uma Transação.

Fundamentação Teórica

Dentre os mecanismos de consenso existentes destacam-se:

- ***Proof of Work (POW)***
  - É o mecanismo de consenso mais conhecido e utilizado pela rede do Bitcoin.
- ***Proof of Stake (POS)***
  - É como uma loteria: O vencedor é determinado por acaso, porém existe um peso aplicado.
- ***Proof of Authority (POA)***
  - Utilizado em *blockchains* privado, onde determinados nós são responsáveis pela mineração.
- ***Proof of Capacity (POC)***
  - Semelhante ao POS, porém o peso é capacidade de armazenamento do nó.

Os *blockchains* podem ser classificados também quanto à forma como são administrados:

- ***Públicos;***
- ***Privados;***
- ***Consórcio.***

Fundamentação Teórica

- **Criptografia**
  - **Simétrica**
    - É o modelo mais antigo de criptografia, tendo a chave compartilhada.
  - **Assimétrica**
    - É o tipo de criptografia empregada em *blockchains,* onde existe um par de chaves, pública e privada.

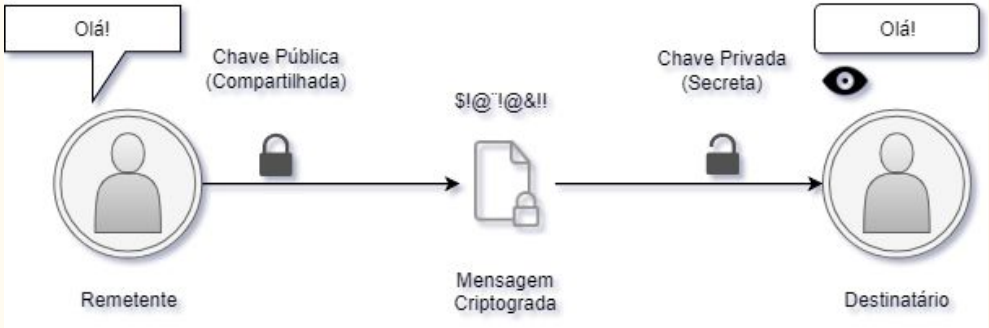


Figura 3: Par de Chaves Assimétrica.

Trabalhos Relacionados

1. OriginalMy

- Pioneira no uso do *blockchain* para validação de documentos.
- Disponibiliza um carimbo de tempo a cada transação.
- Uma vez publicada, a assinatura nunca mais poderá ser removida ou alterada.
- Registra documentos em 4 *blockchains* públicos e outros privados.

2. Signatura

- Permite que várias partes assinem documentos conjuntamente, juridicamente vinculativas e autenticadas.
- Os participantes se autenticam, baixam, descriptografam, revisam e assinam digitalmente a documentação.
- Registra documentos no *blockchain* do Bitcoin.

Trabalhos Relacionados

3. BestSign

- Aplicação desenvolvida pelo *framework Exonum*.
- Utiliza o protocolo de consenso *Proof of Authority*.
- Serviços antes oferecidos de forma física pela empresa, agora mantidos de forma segura e digital via *blockchain*

Implementação e Experimentação

- **Blocksigner**
  - A implementação foi fortemente baseada na estrutura de *blockchain* do Bitcoin, com algumas particularidades:
    - **Moedas:**
      - Não existe a questão monetária relacionada ao trabalho.
    - **Rede Privada:**
      - Desenvolvida com o intuito de ser implementada em um ambiente privado, como a RNP.
    - **Custo Computacional:**
      - Não foi utilizado nenhum método de consenso que gera custo ao minerar blocos.
    - **Chaves Ocultas:**
      - As chaves são armazenadas em cada nó da rede e acessadas por CPF.

Implementação e Experimentação

- **Dependências**
  - Esta API foi desenvolvida usando a linguagem Python e ao todo foi necessário incluir 15 dependências. Dentre elas, as mais importantes são:
    - **Flask:**
      - Esta biblioteca é o coração de Blocksigner e possui várias funções para desenvolvimento web, como recebimento e tratamento de requisições HTTP;
    - **Request:**
      - Envio e tratamento de requisições HTTP;
    - **PyCrypto:**
      - Esta é uma coleção de funções hash seguras (como SHA256) e vários algoritmos de criptografia (AES, DES, RSA, etc.).
    - **etc….**

Implementação e Experimentação

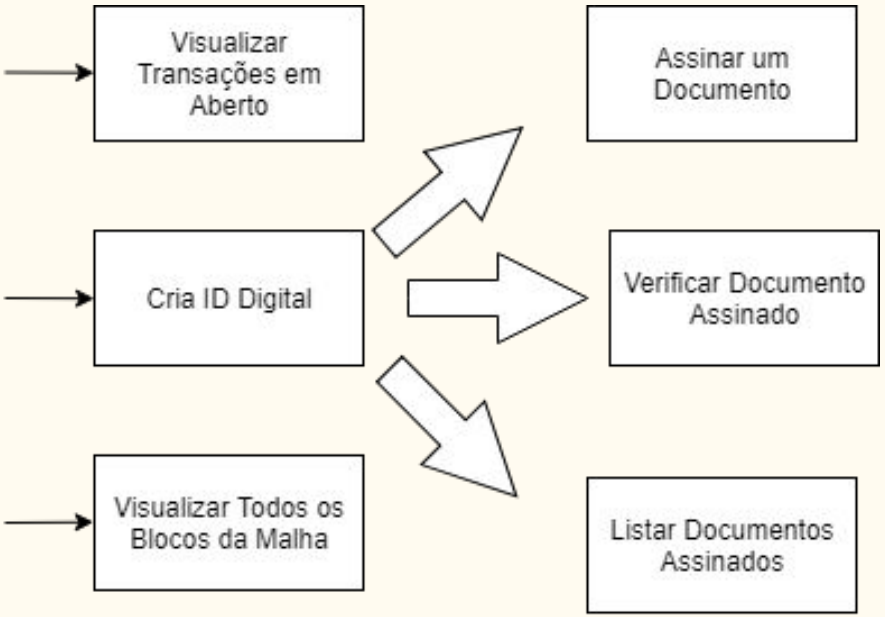


Figura 4: Diagrama de Casos de Uso.

Implementação e Experimentação

- **Ambiente Computacional**

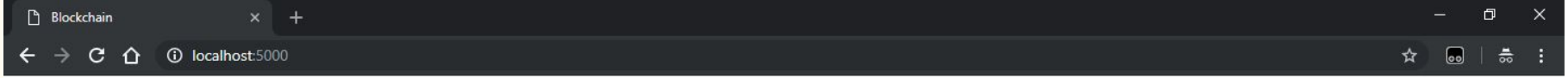
1. **Inicialização:**
   - A API carrega em sua execução uma lista de nós da rede. Note que este arquivo pode ser alterado e que, por padrão, vem configurado no ambiente de teste com 3 endereços locais de máquina, nas portas 5000, 5001 e 5002.
2. **Atualização:**
   - Em seguida, todos nós verificam na rede pela blockchain mais atualizada.
3. **Aprestado:**
   - Após inicializar e atualizar o nó já está pronto para receber e armazenar assinaturas.

Implementação e Experimentação

- **Testes**
  - Os teste foram realizados de duas formas:

- **Hardware Simulado:**
  - Debian 9 64bits
  - Processador Xeon Dual Core 2.0Ghz
  - 64MB de memória RAM

- **Hardware Físico:**
  - Ubuntu 18.04 LTS
  - Xeon dual core 2.0Ghz
  - 8GB RAM



Welcome to the digital signer system!

Sign Document | Create ID | Verify Documents

Create a digital ID before start signing documents!

Document hash

Ex: 35ab8189d4f042d82ac2 ...

Signer's CPF

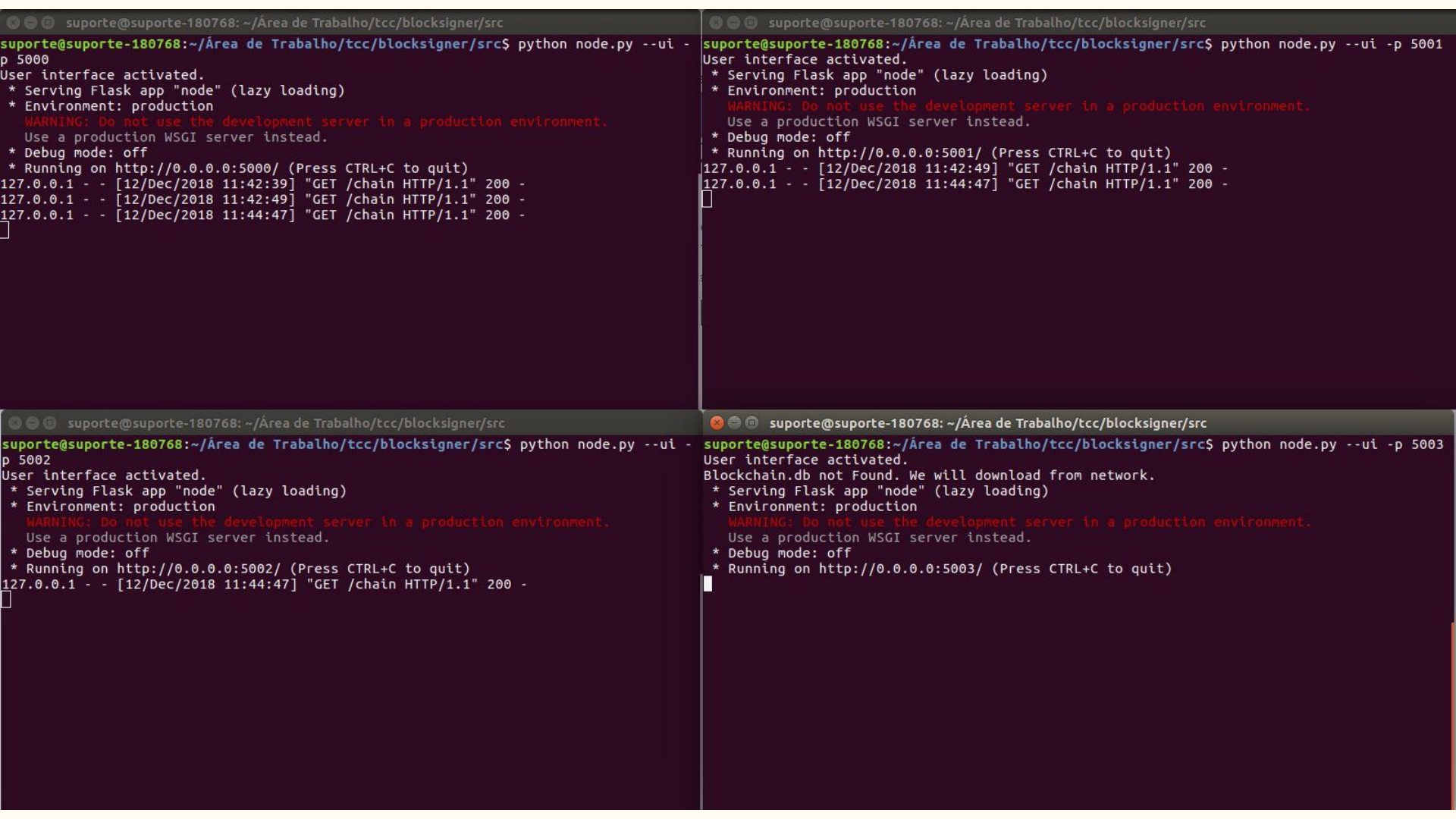
Ex: 12345678910

Only numbers

Send

Blockchain | Open Transactions

Load Blockchain


suporte@suporte-180768: ~/Área de Trabalho/tcc/blocksigner/src
suporte@suporte-180768:~/Área de Trabalho/tcc/blocksigner/src$ python node.py --ui -p 5000
User interface activated.
* Serving Flask app "node" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [12/Dec/2018 11:42:39] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [12/Dec/2018 11:42:49] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [12/Dec/2018 11:44:47] "GET /chain HTTP/1.1" 200 -
suporte@suporte-180768: ~/Área de Trabalho/tcc/blocksigner/src
suporte@suporte-180768:~/Área de Trabalho/tcc/blocksigner/src$ python node.py --ui -p 5001
User interface activated.
* Serving Flask app "node" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5001/ (Press CTRL+C to quit)
127.0.0.1 - - [12/Dec/2018 11:42:49] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [12/Dec/2018 11:44:47] "GET /chain HTTP/1.1" 200 -
suporte@suporte-180768: ~/Área de Trabalho/tcc/blocksigner/src
suporte@suporte-180768:~/Área de Trabalho/tcc/blocksigner/src$ python node.py --ui -p 5002
User interface activated.
* Serving Flask app "node" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5002/ (Press CTRL+C to quit)
127.0.0.1 - - [12/Dec/2018 11:44:47] "GET /chain HTTP/1.1" 200 -
suporte@suporte-180768: ~/Área de Trabalho/tcc/blocksigner/src
suporte@suporte-180768:~/Área de Trabalho/tcc/blocksigner/src$ python node.py --ui -p 5003
User interface activated.
Blockchain.db not Found. We will download from network.
* Serving Flask app "node" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5003/ (Press CTRL+C to quit)

ID's Management

Sucesso, ID foi criado.

Sign Documents | Create ID | Verify Documents

Create Digital ID

12345678910

Only numbers

Create ID

Sucesso, transacao adicionada.

Create a digital ID before start signing documents!

Document hash

ad88130de091bc85acd2d2755145f8a1fe89c9f72bac3a326401ce1ade3045c9

Signer's CPF

12345678910

Only numbers

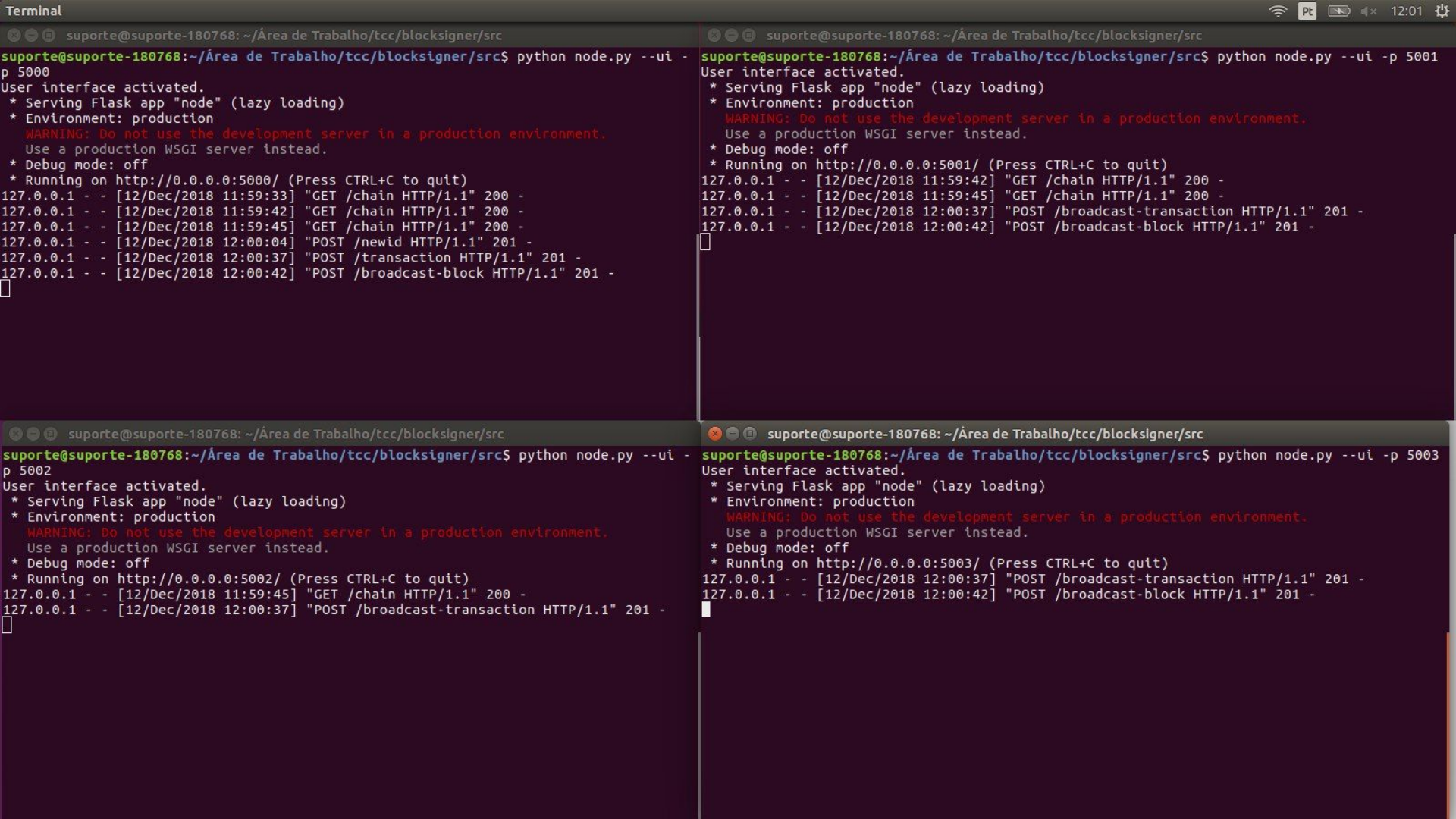
Send

Blockchain | Open Transactions

Load Transactions

Transaction #0

Sender:
30819f300d06092a864886f70d010101050003818d0030818902818100df9e04b05926729683890848cd66ac27f42b35211f2d721feff98a201c6ad641affc0551db5f9ca881d74f5c6c90cbc1ac3b310eb4916a0206cdab9943fbc35c2c9c0be50831fdd6292694f26a79b76b0c1c12d5738ff28a7d1ee257757e542cc7f574af9324eea7040856bc6ae065bb9100195090f538a94d2adc76cec5adc10203010001
Documento: ad88130de091bc85acd2d2755145f8a1fe89c9f72bac3a326401ce1ade3045c9


Terminal
12:01
suporte@suporte-180768: ~/Área de Trabalho/tcc/blocksigner/src
suporte@suporte-180768:~/Área de Trabalho/tcc/blocksigner/src$ python node.py --ui -p 5000
User interface activated.
* Serving Flask app "node" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [12/Dec/2018 11:59:33] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [12/Dec/2018 11:59:42] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [12/Dec/2018 11:59:45] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [12/Dec/2018 12:00:04] "POST /newid HTTP/1.1" 201 -
127.0.0.1 - - [12/Dec/2018 12:00:37] "POST /transaction HTTP/1.1" 201 -
127.0.0.1 - - [12/Dec/2018 12:00:42] "POST /broadcast-block HTTP/1.1" 201 -
suporte@suporte-180768: ~/Área de Trabalho/tcc/blocksigner/src
suporte@suporte-180768:~/Área de Trabalho/tcc/blocksigner/src$ python node.py --ui -p 5001
User interface activated.
* Serving Flask app "node" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5001/ (Press CTRL+C to quit)
127.0.0.1 - - [12/Dec/2018 11:59:42] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [12/Dec/2018 11:59:45] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [12/Dec/2018 12:00:37] "POST /broadcast-transaction HTTP/1.1" 201 -
127.0.0.1 - - [12/Dec/2018 12:00:42] "POST /broadcast-block HTTP/1.1" 201 -
suporte@suporte-180768: ~/Área de Trabalho/tcc/blocksigner/src
suporte@suporte-180768:~/Área de Trabalho/tcc/blocksigner/src$ python node.py --ui -p 5002
User interface activated.
* Serving Flask app "node" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5002/ (Press CTRL+C to quit)
127.0.0.1 - - [12/Dec/2018 11:59:45] "GET /chain HTTP/1.1" 200 -
127.0.0.1 - - [12/Dec/2018 12:00:37] "POST /broadcast-transaction HTTP/1.1" 201 -
suporte@suporte-180768: ~/Área de Trabalho/tcc/blocksigner/src
suporte@suporte-180768:~/Área de Trabalho/tcc/blocksigner/src$ python node.py --ui -p 5003
User interface activated.
* Serving Flask app "node" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5003/ (Press CTRL+C to quit)
127.0.0.1 - - [12/Dec/2018 12:00:37] "POST /broadcast-transaction HTTP/1.1" 201 -
127.0.0.1 - - [12/Dec/2018 12:00:42] "POST /broadcast-block HTTP/1.1" 201 -

Conclusão

Proposta inovadora

- Escassez de referências teóricas e práticas.
- Desenvolvimento desafiador e exaustivo.
- Tendencia para armazenamento distribuído.

Trabalhos Futuros

- Aplicar melhora em escalabilidade e usabilidade.
- Chord
- IPFS
- Armazenar localmente os documentos assinados.

Referências

1. Greve, F., Sampaio, L., Abijaude, J., Coutinho, A., Valcy, ´I., and Queiroz, S. Blockchain e a revolução do consenso sob demanda.
2. Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper, 151:1–32.
3. Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.
4. Oliveira, R. R. (2012). Criptografia simétrica e assimétrica-os principais algoritmos de cifragem. Segurança Digital [Revista online], 31:11–15.
5. Singh, G. (2013). A study of encryption algorithms (rsa, des, 3des and aes) for information security. International Journal of Computer Applications, 67(19).
6. Benet, J. (2014). Ipfs-content addressed, versioned, p2p file system. arXiv preprint arXiv:1407.3561.
7. Croman, K., Decker, C., Eyal, I., Gencer, A. E., Juels, A., Kosba, A., Miller, A., Saxena, P., Shi, E., Sirer, E. G., et al. (2016). On Scaling Decentralized Blockchains. In International Conference on Financial Cryptography and Data Security, pages 106– 125. Springer.
8. Thakur, J. and Kumar, N. (2011). Des, aes and blowfish: Symmetric key cryptography algorithms simulation based performance analysis. International journal of emerging technology and advanced engineering, 1(2):6–12

Questões e Comentários

BlockSigner

Acadêmicos: Gabriel Marques, Murilo B. Flor
Orientador: Prof. Brivaldo Junior