Universidade Federal
do Espírito Santo

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

Segurança em Computação – 2024/1
Prof. Rodolfo da Silva Villaça – rodolfo.villaca@ufes.br
Trabalho T1 – PKI Descentralizada usando Blockchains
Versão 1.1 – 17/07/2024

Objetivos:

1. Implementar um sistema distribuído, descentralizado, como alternativa à Infraestrutura de Chaves Públicas (PKI, ou *Public Key Infrastructure*) tradicional[1], baseada em Autoridades Certificadoras (AC) hierárquicas;
2. Experimentar a implementação de sistemas Web3[2], mais especificamente se sugere a utilização da tecnologia *Cartesi*[3] e *blockchains* baseadas na *Ethereum Virtual Machine* (EVM).

Requisitos:

1. O trabalho pode ser feito em grupos de até 3 alunos: sob nenhuma hipótese serão aceitos trabalhos em grupos maiores. Também não se recomenda a implementação individual;
2. Caso o grupo opte por utilizar a tecnologia *Cartesi*, sugere-se fortemente que o *backend* da aplicação seja desenvolvido na linguagem Python.

Especificação:

O trabalho consiste na implementação de uma PKI descentralizada, ou DPKI (*Decentralized Public Key Infrastructure*). Para melhor se conceitualizar sobre uma DPKI, consulte [1, 2, 3]. A implementação solicitada nesta disciplina é simplificada, baseada em alguma *blockchain* de desenvolvimento Ethereum (Devnet), com execução local e já disponibilizada no *setup* da Cartesi.

A Figura 1 traz um esboço e uma visão geral da *Cartesi* e sua relação com a *web3* e *blockchain*[4]. Destaca-se aqui a importância de se conhecer a arquitetura[3] da *Cartesi* para melhor entender a figura. Outro ponto fundamental é que a *Cartesi* não é uma *blockchain*! Tecnicamente, a *blockchain* de infraestrutura é qualquer solução de teste baseada na EVM (Ethereum).

1 https://pt.wikipedia.org/wiki/Infraestrutura_de_chaves_p%C3%BAblicas
2 https://aws.amazon.com/pt/what-is/web3/
3 https://docs.cartesi.io/cartesi-rollups/1.3/core-concepts/architecture/
4 https://www.bbc.com/portuguese/geral-59951952

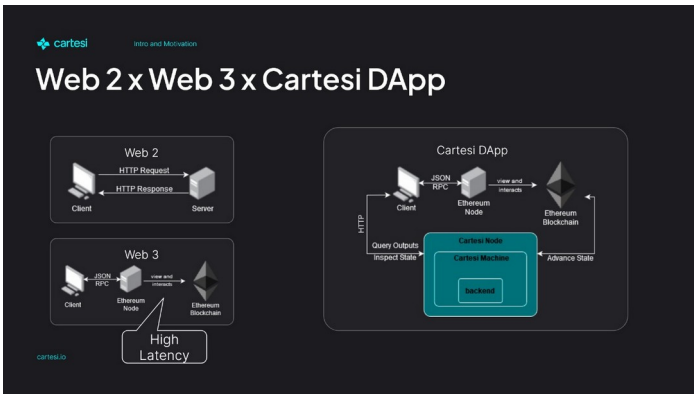

Figura 1: Visão geral da arquitetura da Cartesi, e sua relação com a Web3 e blockchains.

Na Figura 1 observa-se, no canto superior esquerdo a *web2*, que representa o desenvolvimento tradicional para serviços web, baseados no protocolo HTTP. A *web2* é a internet que conhecemos atualmente. A *web3* modifica a estrutura de base do modelo "social" da *web2* para torná-la mais descentralizada, por meio de tecnologias inovadoras como *blockchain* e criptomoedas. Destaca-se aí a alta latência das aplicações baseadas em *blockchains*. No lado direito da figura observa-se a proposta da *Cartesi* para mitigar essa limitação. Observe que as interações que fazem alterações na rede são feitas como na *web3* e com base em tecnologias como RPC e JSON. Entretanto, todas as inserções de transações na *blockchain* são registradas em um *Cartesi Node*, que "replica" (espelha) o estado da aplicação modificada. As consultas de estado são feitas diretamente no nó *Cartesi*, que mantém uma cópia fidedigna do estado desta aplicação, permitindo a verificação da inviolabilidade das transações. Mais detalhes virão a seguir...

Uma visualização sugerida para a implementação da DPKI, e é composta basicamente de 3 partes: *frontend*, *backend* e a *blockchain*, e pode ser observada na Figura 2.

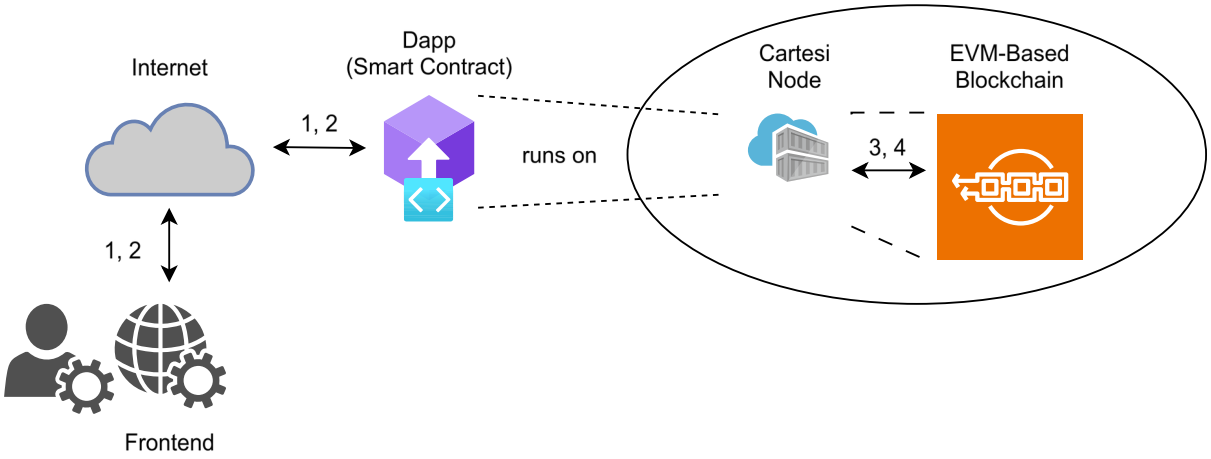

Figura 2: Arquitetura sugerida para a implementação do protótipo da DPKI baseada na Cartesi Machine e Ethereum EVM.

A interação do usuário com a DApp (*Distributed Application*) que implementada por vocês dar-se-á por meio do *frontend*. Sugere-se que esse *frontend* seja baseado em web (pontuação extra será concedida). Para melhor diferenciação entre uma DApp e um Contrato Inteligente (*Smart Contract*), veja [4].

Na aplicação sugerida nesta disciplina (ou seja, a DPKI), as interações 1 e 2, representadas na Figura 1, seriam do tipo *Advance* (ou *publish*) e *Inspect* (ou *query*). Como exemplo, o *Inspect* é uma interação que não altera o estado da aplicação, e simplesmente representa uma requisição HTTP diretamente para o *Cartesi Node*.

Para ilustrar, como exemplo imagine que você tenha um banco de dados SQL na sua Dapp. Caso você queira consultar esse banco (SELECT), envie uma interação do tipo *Inspect*. Mas, se você quiser fazer uma mudança nesse banco (INSERT, UPDATE, DELETE) você deve enviar uma interação do tipo *Advance*. O *Advance* é uma interação capaz de mudar (avançar) o estado da aplicação e por isso precisa ficar registrada na *blockchain*, Essa interação ficará disponível (registrada) na *blockchain* para que outros validadores (participantes do sistema) possam reproduzir essa computação e validar o estado da *Cartesi Machine*.

No use case previsto na disciplina de Segurança em Computação, a interação com o usuário seria por meio de APIs *web* para: i) inserir/registrar uma nova chave pública por um usuário, com mensagem

assinada; ii) solicitar alteração do estado de uma chave, inutilizando-a, por exemplo, por meio de mensagem também assinada pelo usuário. Esta ação é equivalente a revogar a chave pública do sistema, ou, torná-la inativa.

Ainda considerando o *use case* previsto na disciplina, a interação onde o usuário recebe um retorno do *frontend* seria para permitir uma consulta a uma chave registrada por um usuário. O sistema deve retornar o certificado em si, e seu status. As inserções e revogações de chaves representam alterações no estado da *Cartesi Machine* (alterações no banco de dados, por exemplo). É importante destacar aqui que o uso de um BD do tipo SQL na sua DApp é algo opcional, usado como exemplo até aqui. O uso de dicionários ou arquivos também são consideradas soluções válidas.

Em [5] pode-se consultar um exemplo de uma aplicação de calculadora, implementada em Python, como uma DApp em uma *Cartesi Machine*. Os passos de instalação da *Cartesi Machine* em sua máquina local estão disponíveis em [6].

As interações 3 e 4, representadas na Figura 1, representam as interações da *Cartesi Machine* com a *blockchain*. Mais especificamente, uma operação de *Advance* é sempre feita diretamente na *blockchain*, de tal forma que a interação fique registrada, com seus inputs e resultados (novo estado). No nosso use case, as alterações são a inclusão de novos certificados, alterações de status dos certificados já existentes e registrados; não há consultas diretamente nos registros da *blockchain*.

Por tratar-se, a *Cartesi Machine*, de uma máquina determinística baseada na arquitetura RISC-V, qualquer usuário de posse de todas as interações e de uma *Cartesi Machine* "zerada" (em seu estado inicial), é possível reproduzir todos os passos, conferir as mudanças e verificar o estado atual da aplicação, atestando-se assim a inviolabilidade das transações.

Informações Finais:

1. Data de Entrega: 19/08/2024;
2. Submeter link do Github, em modo público, com o código devidamente documentado e comentado, scripts e intruções de execução / reprodução;

3. Disponibilizar link de vídeo (ou incluir no Github) com exemplo de execução, utilização e testes realizados com a implementação submetida. Importante: deve ser possível reproduzir os exemplos contidos no vídeo disponibilizado.

Referências Bibliográficas:

[1] Decentralized Public Key Infrastructures (DPKI)
[2] A Blockchain-Based Decentralized Public Key Infrastructure Using the Web of Trust
[3] Decentralized Public Key Infrastructure
[4] Smart contracts vs. dApps—how are they different?
[5] Build a calculator dApp with Python
[6] Cartesi Quickstart

Links Adicionais:

[7] Cartesi High Level Framework for developing Dapps
[8] Cartesi community's repository of proof of concepts, hackathon projects, and experimental Dapps
[9] SolarMarket – desenvolvimento por alunos da Ufes (CC/PPGI)
[10] *Deploy* da DApp em uma rede de teste real da Ethereum
[11] Play Riscv-binaries games on a RISC-v Cartesi Machine on the browser