Ambientes de Computação Segura

Laboratório de Arquitetura e Redes de Computadores (LARC)
Departamento de Engenharia de Computação e Sistemas Digitais (PCS)
Escola Politécnica da Universidade de São Paulo (EPUSP)

Ambientes de Computação Segura – Equipe:

Prof. Dr. Wilson Ruggiero

Prof. Dr. Marcos Simplício Junior

Prof. Dr. Stephan Kovach

Prof. Dr. Julião Braga

Nelson Yamamoto

Romeo Bulla Junior

Agradecimentos:
Leonardo Toshinobu Kimura
Lucas Cupertino

Roteiro

- Conceitual
  - Introdução e Conceitos
  - Aspectos Relevantes de Segurança
  - Superfícies de Ataque
- Visão Geral sobre Algumas Tecnologias Existentes
- Tecnologia Intel SGX
- Prática: Ferramenta de *Dump* (Autenticador pyhton usando MongoDB)
- Prática: Apresentação de Autenticador em C++ para Experimentos

Roteiro (continuação)

- Visão Geral das Ferramentas Apresentadas no Minicurso
- Ferramenta e Prática - Intel SGX SDK
- Ferramenta e Prática - Open Enclave SDK
- Ferramenta e Prática - Occlum
- Case de Exemplo: Autenticador OAuth na nuvem com Intel SGX
- Tecnologia AMD SEV/SNP
- Tecnologia ARM CCA
- Ataques CacheZoom e CrossLine
- Outras Tecnologias: RISC-V Sanctum e Keystone

Introdução

Introdução

- A utilização de ambientes em nuvem veio a facilitar a implantação de infraestruturas que suportam a execução de sistemas para as mais diversas atividades e propósitos.

- Como resultado dessa facilidade, muitas empresas utilizam a computação em nuvem de modo a ampliar sua infraestrutura ou, até mesmo, abrigá-la por completo em nuvem.

Servidores empresa XYZ

Introdução

- Porém, emergem desse fato novas preocupações:

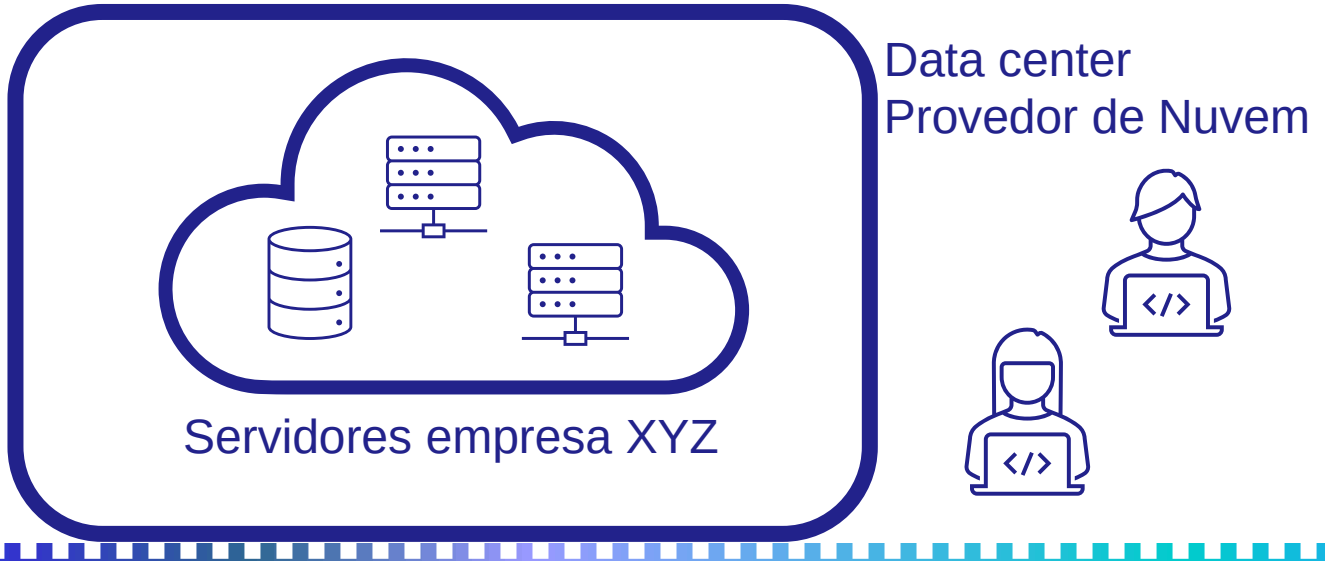
Servidores empresa XYZ

Introdução

- Porém, emergem desse fato novas preocupações:
  - Como executar softwares em um ambiente remoto (nuvem) de modo seguro, considerando-se que o computador físico se encontra em um ambiente controlado por outra empresa?

Introdução

- Porém, emergem desse fato novas preocupações:
  - Como executar softwares em um ambiente remoto (nuvem) de modo seguro, considerando-se que o computador físico se encontra em um ambiente controlado por outra empresa?
  - E como garantir a privacidade do processamento sabendo que a empresa provedora de nuvem possui acesso privilegiado a todos ambientes de computação hospedados?



Introdução

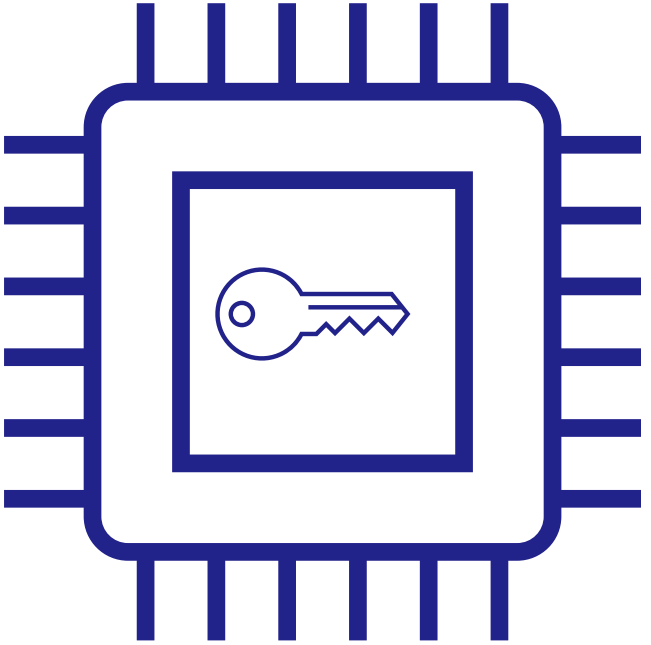
- Os fabricantes de hardware responderam a essas preocupações criando meios de se executar programas (e máquinas virtuais) na nuvem de forma privada e segura.
- Contudo, a simples existência do hardware não resolve os problemas.
  - É necessário ter um ponto de vista mais crítico, pois existem diversas observações e boas práticas a seguir já comunicadas por diversos pesquisadores em seus trabalhos práticos.
  - Algumas são abordadas nessa apresentação.

AMD RISC-V® intel® arm

Conceitos de TEEs

Conceitos de TEEs

- TEE (*Trusted Execution Environment*) é um ambiente de computação segura, implementado em áreas isoladas dentro dos microprocessadores que permitem a execução segura de programas com os objetivos de garantir a integridade e confidencialidade da execução.



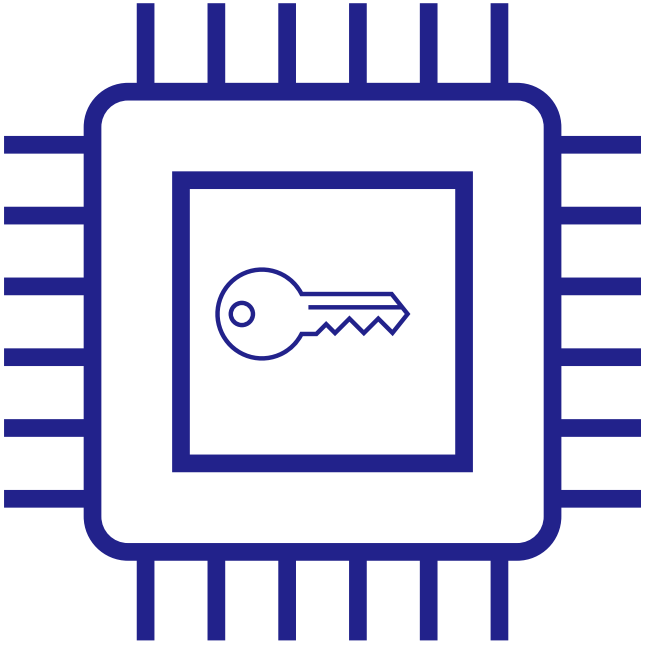
Conceitos de TEEs

- TEE (*Trusted Execution Environment*) é um ambiente de computação segura, implementado em áreas isoladas dentro dos microprocessadores que permitem a execução segura de programas com os objetivos de garantir a integridade e confidencialidade da execução.

- Esses ambientes também incluem alguma forma de proteger a memória RAM. Seja por:

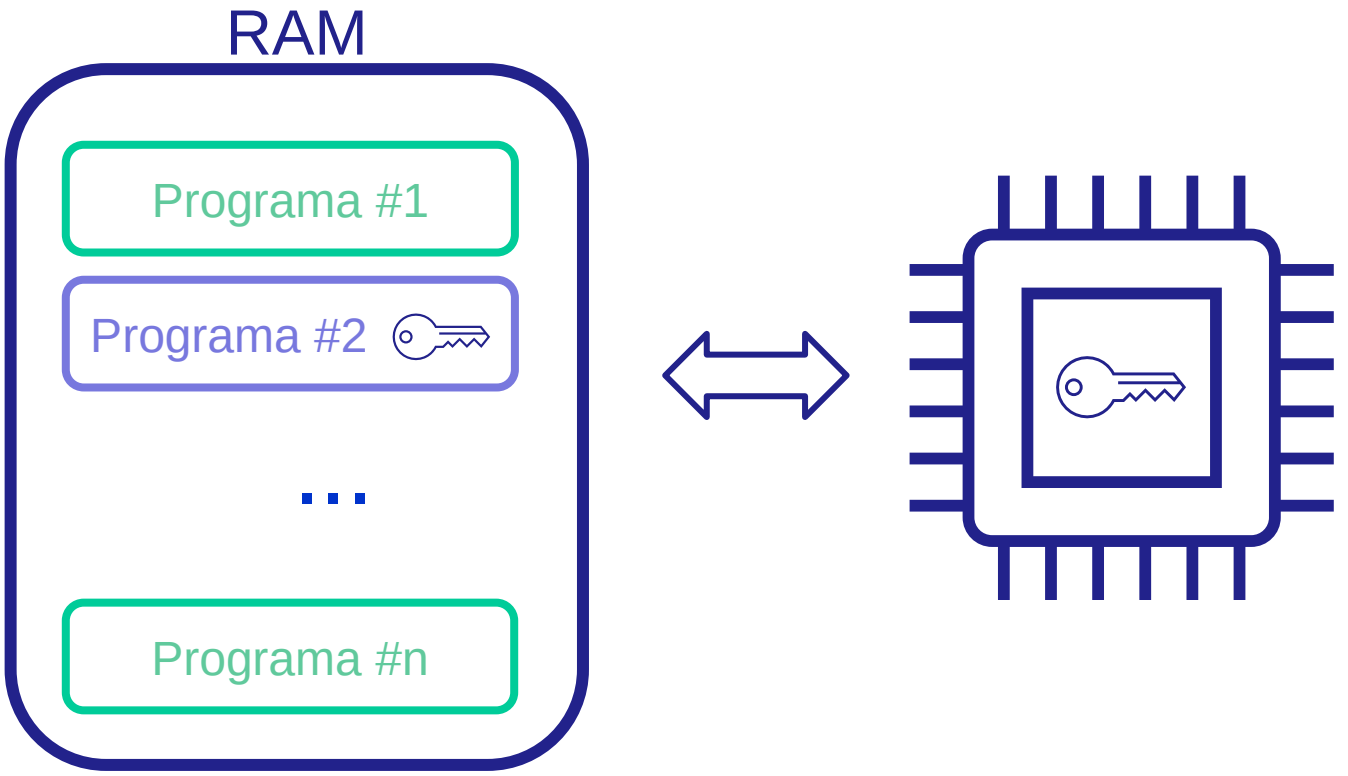
  - restrição de acesso a determinados endereços; e/ou

  - pela cifração (criptografia) dos dados armazenados.



Conceitos de TEEs

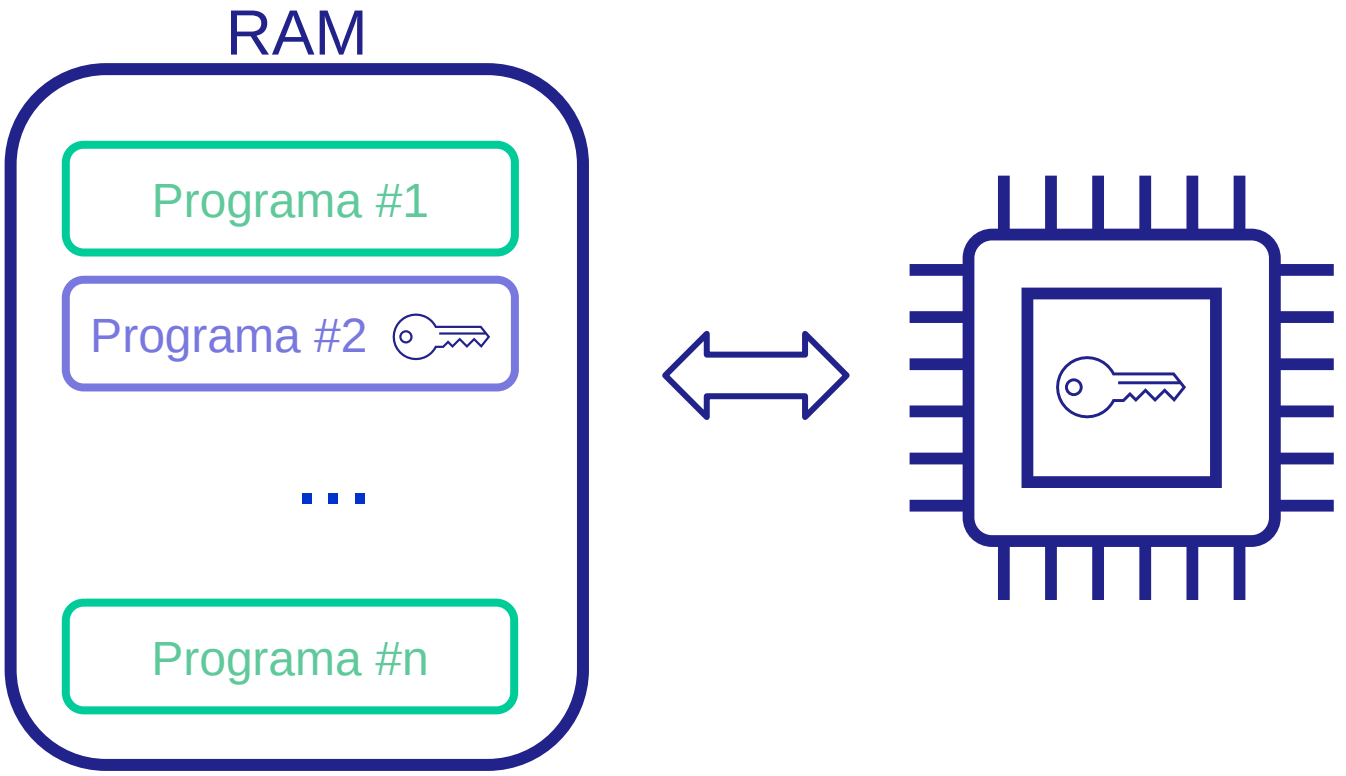
- Deste modo, temos:
  - Segurança na execução;
  - Segurança dos dados



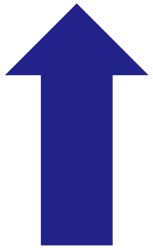
Conceitos de TEEs

- É importante observar que nem todos os programas (ou máquinas virtuais) são executados em ambientes seguros.

- Somente os programas críticos são executados em TEEs, por exemplo, programas que realizam: autenticação, cifração/decifração, ...



Conceitos de TEEs



Benefícios:

- Confidencialidade;
- Integridade;

Desvantagens:

- Desempenho geral de execução reduzido;
- Limitação no uso ou acesso de alguns recursos de hardware.

Aspectos Relevantes de Segurança

Aspectos Relevantes de Segurança

- É necessário abordar alguns pilares de segurança dos TEEs de modo a esclarecer o que está seguro diferenciando do que somente aparenta estar seguro.
- Os fabricantes criaram soluções que, aparentemente, resolvem todo o problema de confidencialidade e integridade.
- Mas é preciso avaliar como os TEEs implementam as soluções, pois somente a execução segura e a proteção dos dados na memória RAM não resolvem tudo.
- Por exemplo:
  - Como é possível saber que o atacante não substituiu o software que se deseja executar na nuvem?

Aspectos Relevantes de Segurança

De acordo com o (Li et al, 2023) é possível se orientar pelas seguintes características, ou requisitos, a fim de averiguar se uma instância de TEE está segura:

- **(1) Secure Measurement:** o *hardware* que implementa o TEE e suas bibliotecas devem prover um meio de se "medir" (obter o *measurement*) o hardware e os estados iniciais de uma instância de TEE para que seja possível a **atestação remota**, ou seja, aferir se o *software* desejado é efetivamente o executado;

- **(2) Confidencialidade:** o *hardware* que implementa o TEE e suas bibliotecas devem manter os dados da instância de TEE confidenciais contra atacantes que utilizam *software* e/ou opcionalmente, *hardware*;

- **(3) Integridade:** o *hardware* que implementa o TEE e suas bibliotecas devem proteger instâncias de TEEs de acesso não autorizado e manipulação de dados e código.

Aspectos Relevantes de Segurança

De acordo com o (Li et al, 2023) é possível se orientar pelas seguintes características, ou requisitos, a fim de averiguar se uma instância de TEE está segura:

- **(1) Secure Measurement:** o *hardware* que implementa o TEE e suas bibliotecas devem prover um meio de se "medir" (obter o *measurement*) o hardware e os estados iniciais de uma instância de TEE para que seja possível a **atestação remota**, ou seja, aferir se o *software* desejado é efetivamente o executado;
- **(2) Confidencialidade:** o *hardware* que [illegible] bibliotecas devem manter os dados da instância de [illegible] tacantes que utilizam *software* e/ou opcionalmente
- **(3) Integridade:** o *hardware* que imple[illegible]tecas devem proteger instâncias de TEEs de ac[illegible]lação de dados e código.

Realizado pelo hardware ou software do fabricante

Realizado pelo hardware ou software do fabricante

Aspectos Relevantes de Segurança

De acordo com o (Li et al, 2023) é possível se orientar pelas seguintes características, ou requisitos, a fim de averiguar se uma instância de TEE está segura:

- **(1) Secure Measurement:** o *hardware* qu… bibliotecas devem prover um meio de se "r… *nt*) o hardware e os estados iniciais de uma inst… a possível a **atestação remota**, ou seja, aferir se o *software* desejado é efetivamente o executado;

- **(2) Confidencialidade:** o *hardware* que … bibliotecas devem manter os dados da instância de … tacantes que utilizam *software* e/ou opcionalmente…

- **(3) Integridade:** o *hardware* que imple… tecas devem proteger instâncias de TEEs de ac… lação de dados e código.

Necessidade de verificação: Atestação

Realizado pelo hardware ou software do fabricante

Realizado pelo hardware ou software do fabricante

Aspectos Relevantes de Segurança

Atestação:

- É um processo para averiguar a integridade e confiança em um componente e/ou sistema;
- Contém dados e informações que encapsulam e garantem que:
  - O *hardware* e/ou a plataforma para realizar a execução é realmente o pretendido; e
  - O software enviado para execução é realmente o carregado pelo hardware e/ou plataforma.
- Uma atestação pode ser:
  - **Local**: realizada e validada no próprio computador que abriga o TEE;
  - **Remota**: realizada no computador que abriga o TEE e validada em outro computador.

Aspectos Relevantes de Segurança

Exemplo de Atestação:

- Ao executar um determinado programa (**progA**) em um processador (**processadorX**), o processo de atestação poderia obter o **hash(progA)** e assiná-lo com a chave privada do processador, gerando um *report* que é transmitido junto com um certificado do processador.

- Uma vez que o programa que solicitou a execução receber o *report*, ele deve confirmar que:

  - O **hash(progA)** está correto;

  - O certificado enviado pelo processador é da cadeia correta, ou seja, é proveniente do fabricante do processador;

  - A assinatura é válida.

Aspectos Relevantes de Segurança

Exemplo de Atestação:

- Ao executar um determinado programa (**progA**) em um processador (**processadorX**), o processo de atestação poderia obter o **hash(progA)** e assiná-lo com a chave privada de processador, gerando um *report* que é transmitido junto com um certificado do processador.

- Uma vez que o programa que solicitou a execução receber o *report*, ele deve confirmar que:

  - O **hash(progA)** está correto;

  - O certificado enviado pelo processador é da cadeia correta, ou seja, é proveniente do fabricante do processador;

  - A assinatura é válida.

Autenticamos a plataforma remota

Aspectos Relevantes de Segurança

Passos da Atestação Remota:

1) A instância TEE em si ou seu proprietário (a entidade que solicitou a execução) inicia a requisição de atestação remota para um **supervisor de atestação**.
    - Por sua vez, vai depender da TCB (*Trusted Computing Base*) de cada fabricante: pode ser processador, coprocessador, uma instância específica TEE ou outro.
2) O **supervisor de atestação** gera e assina um relatório (*report*) de atestação.
    - Ele contém informações críticas do TEE como:
        - Medida (*measurement*) segura dos estados iniciais;
        - Detalhes da plataforma; e
        - Configurações
3) Os usuários verificam o relatório (*report*) antes de utilizar ou transmitir dados para a instância TEE

Aspectos Relevantes de Segurança

Ciclo de vida de uma instância TEE:

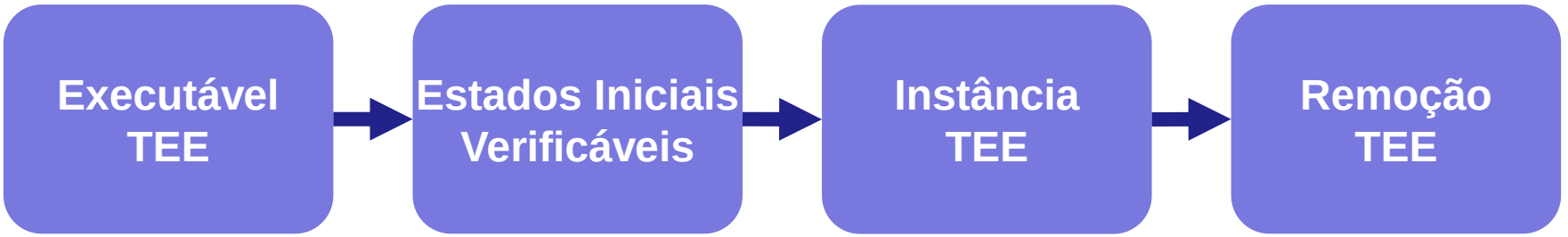


Figura extraída de (Li et al, 2024)

Aspectos Relevantes de Segurança

Ciclo de vida de uma instância TEE:

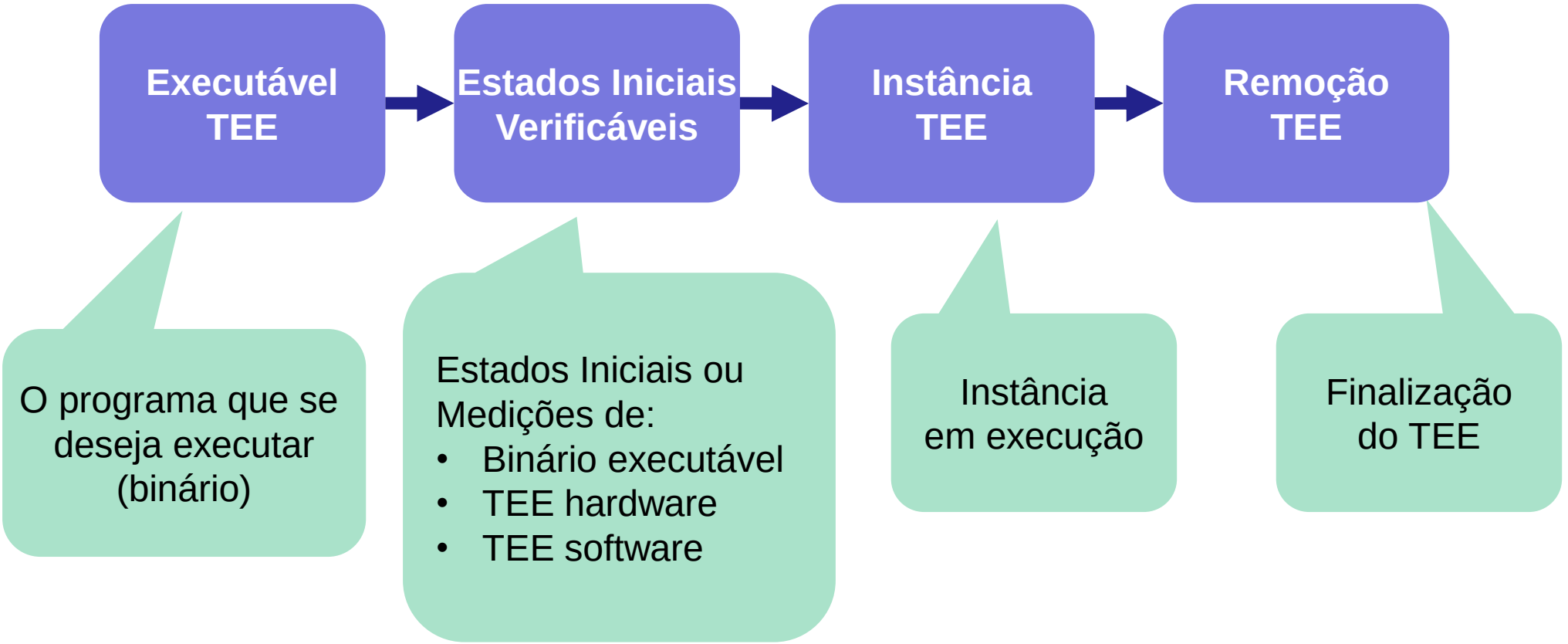


Figura extraída de (Li et al, 2024)

Aspectos Relevantes de Segurança

Ciclo de vida de uma instância TEE:

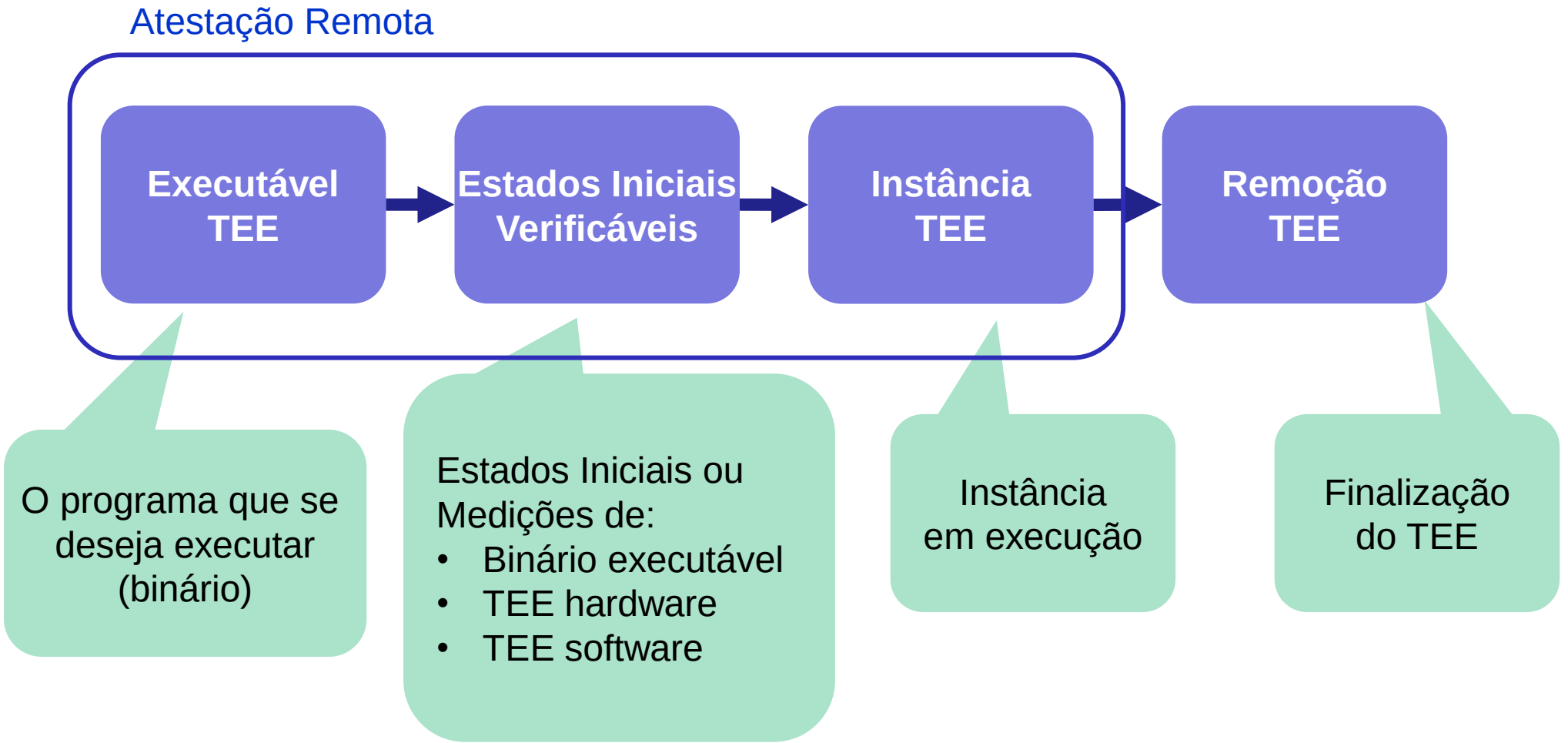


Figura extraída de (Li et al, 2024)

Aspectos Relevantes de Segurança

Ciclo de vida de uma instância TEE:

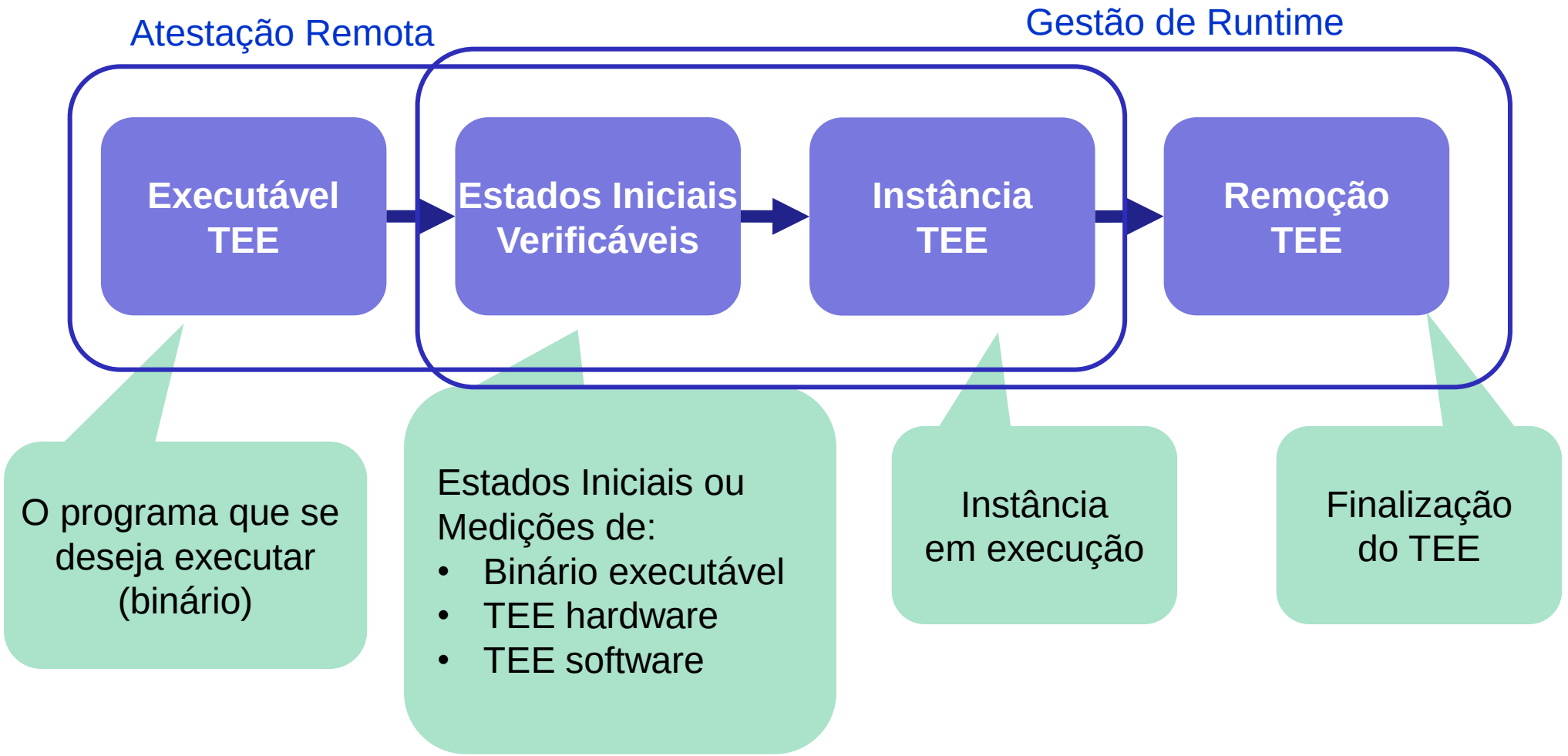
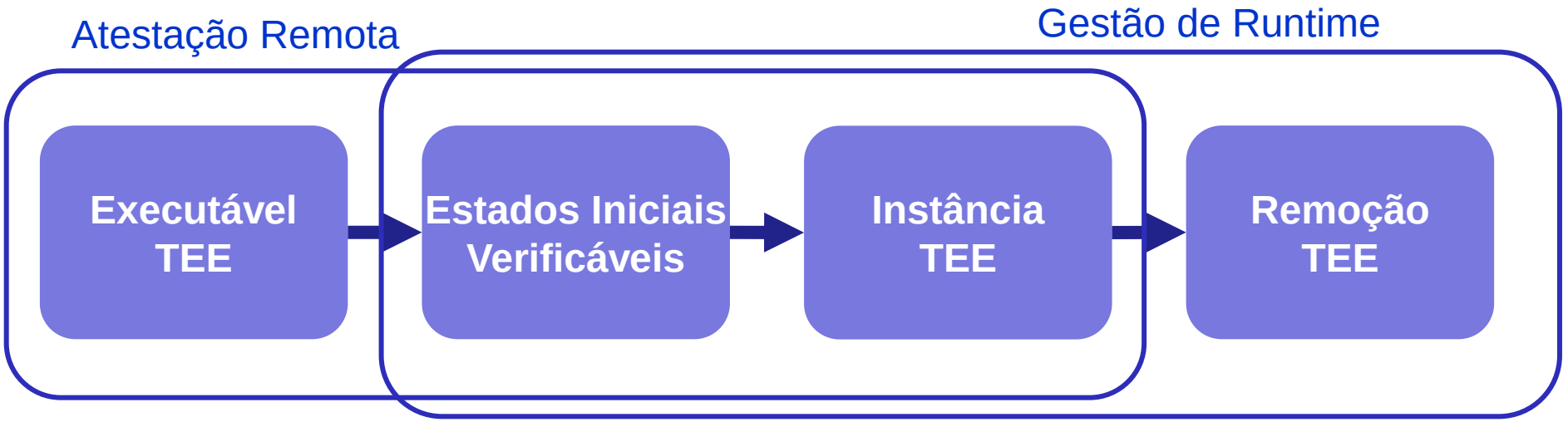


Figura extraída de (Li et al, 2024)

Aspectos Relevantes de Segurança

Ciclo de vida de uma instância TEE:



- **Atestação Remota**: um relatório (*report)* como meio de se garantir que o executável desejado foi carregado no ambiente seguro desejado ou *Trusted Computing Base* (TCB);

- **Gestão de Runtime**: Foco em dois aspectos: a gestão eficiente dos recursos e a segurança.

Figura extraída de (Li et al, 2024)

Aspectos Relevantes de Segurança

- **Observações:**
  - O fato de se utilizar um TEE não significa que se está 100% seguro
  - O SO pode ser um ponto de falha, ou seja, sempre que possível, tentar validar o SO, procurando por vírus ou outros programas maliciosos.
  - Veremos que há ataques que conseguem deduzir informações que estão dentro dos TEEs a partir de modificações nos SOs (Apresentados na seção Superfícies de Ataques)

Superfícies de Ataque

Superfícies de Ataque

Vulnerabilidades em Tradução de Endereços (*address translation vulnerabilities*):

- Nesses tipos de ataque, o atacante se utiliza de acesso privilegiado a máquina vítima.
- Por vezes, o ele já conhece os binários e o código-fonte. Então o atacante deduz informações sensíveis analisando os perfis de acesso às páginas de memória.

Superfícies de Ataque

Ataques Baseados em *cache* (ou *cache timing attacks*):

- Nesses tipos de ataques, o atacante se apoia na monitoração do uso de memória cache.
- Por vezes, o atacante primeiramente preenche a memória *cache* com valores conhecidos, permite que o programa vítima seja executado por instantes e, depois, analisa o perfil de acesso da memória cache (baseado no tempo de resposta do acesso).
- Como ela está diretamente relacionada a memória principal, o atacante pode inferir dados sensíveis a partir desse perfil;

Superfícies de Ataque

Vulnerabilidades da memória DRAM:

- Esse tipo de vulnerabilidade de aplica a programas que compartilham a memória RAM como, por exemplo, máquinas virtuais.
- O perfil de acesso à memória RAM é mapeado, de modo semelhante aos ataques baseados em cache, isto é, observando-se o tempo de resposta.
- Isso ocorre, pois existe um buffer de linha na própria memória RAM.
- Contudo, há muito ruído para o atacante inferir dados sensíveis, então ele apresenta uma acurácia menor que os ataques baseados em cache;

Superfícies de Ataque

Vulnerabilidades de execução especulativa (ou *Branch Prediction Vulnerabilities*):

- Os processadores modernos realizam uma predição de próxima instrução a ser executada mesmo sem ter concluído a instrução corrente.
- É um mecanismo para aumentar a eficiência computacional. Mas ciente disso o atacante pode, por exemplo, preencher o buffer de instruções preditivas para averiguar (medir o tempo de resposta) se o programa vítima executará uma determinada instrução.
- A partir dessa medição, o ele obtém o perfil de execução do programa vítima. Com o perfil, é possível que ele obtenha algum dado sensível;

Superfícies de Ataque

Vulnerabilidades de *hardware*:

- O sucesso desse tipo de ataque depende muito do conhecimento específico que um atacante possui sobre o *hardware* utilizado na máquina que executa o programa vítima.

- Se for interessante, o atacante pode ter muito sucesso ao provocar uma negação de serviço pelo fato de conhecer bem as vulnerabilidades de um *hardware*.

Superfícies de Ataque

Observações:

- Muitas das superfícies de ataques são baseadas na observação de **perfis de acesso à memória**. Isso decorre do fato que **acessos diretos** a conteúdos cifrados ou **a endereços de memória protegidos** por ambientes de computação segura são muito **difíceis de ocorrer**.

- Importância: demonstrar meios de aferição das soluções de TEE, muitos dos quais utilizados por diversos autores.

Tecnologias

Tecnologias

- Esse item apresenta algumas das tecnologias existentes que implementam TEE.
- Toda conceituação apresentada anteriormente está presente nas tecnologias.
- Tecnologias proprietárias. Ex.:
  - Intel SGX
  - AMD SEV/SNP
  - ARM CCA (*Confidential Compute Architecture*)
- Tecnologias abertas (*open*). Ex.:
  - RISC –V Sanctum
  - RISC –V Keystone

Tecnologias

- **CCC - Confidential Computing Consortium**
  - O grupo Confidential Computing Consortium (CCC) aproxima fabricantes de hardware, provedores de nuvem e desenvolvedores de software com a finalidade de acelerar a adoção de tecnologias e padrões para TEEs.
- Importância:
  - Pode ser um local de busca por tecnologias e soluções de TEEs que sejam abertas
  - Diversidade de projetos: TEEs por hardware, bibliotecas para rodar TEEs, bibliotecas para atestação, entre outros

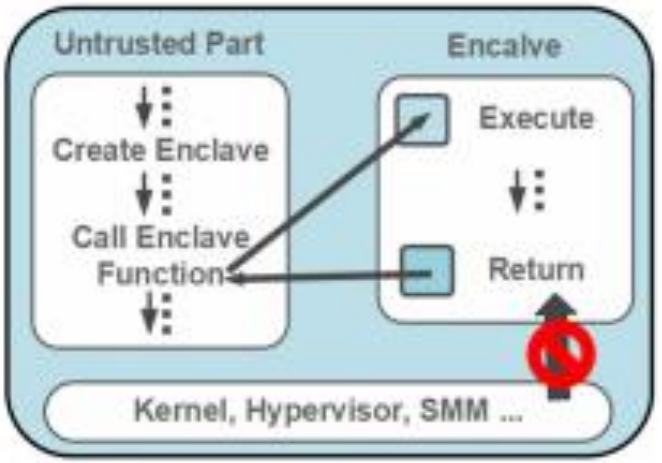
Tecnologias - SGX

SGX

- **Objetivos**
  - Permite a uma aplicação instanciar uma **área isolada** em seu espaço de endereçamento (conhecida como **enclave**)
  - A tecnologia SGX deve garantir:
    - Confidencialidade
    - Integridade
    - Multiplicidade de enclaves (podem existir diversos enclaves simultaneamente, mas isolados entre si)

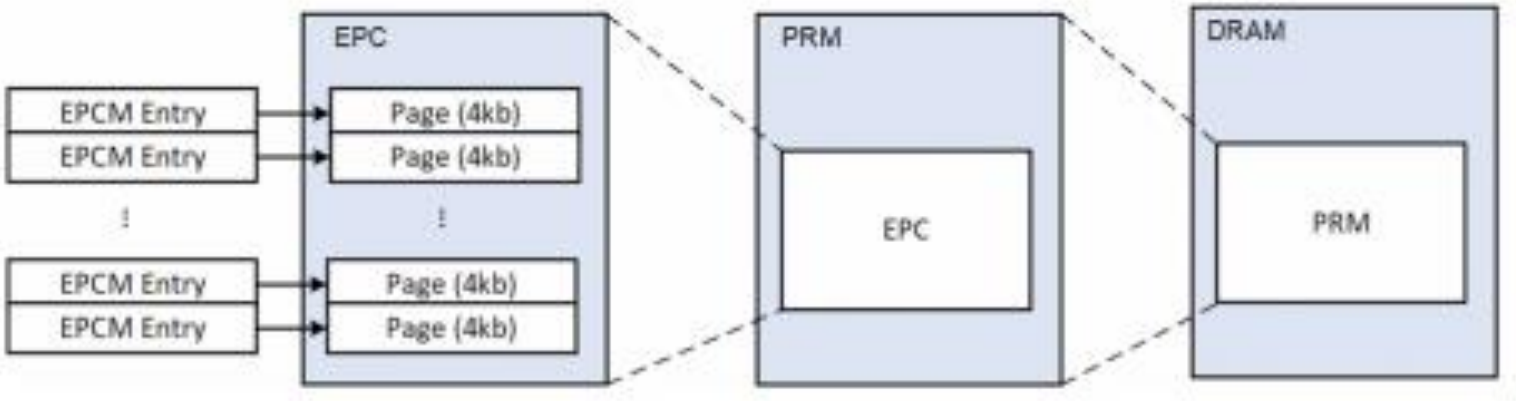
SGX

- **Funcionamento Simplificado**
  - Programa cria o enclave e carrega dentro dele código e dados
  - A execução prossegue, até que o programa chama uma função dentro do enclave
  - Somente o código dentro do enclave pode acessar a memória *Processor Reserved Memory* (PRM) – acessos externos são proibidos
  - Um enclave é parte de um processo (*software*)



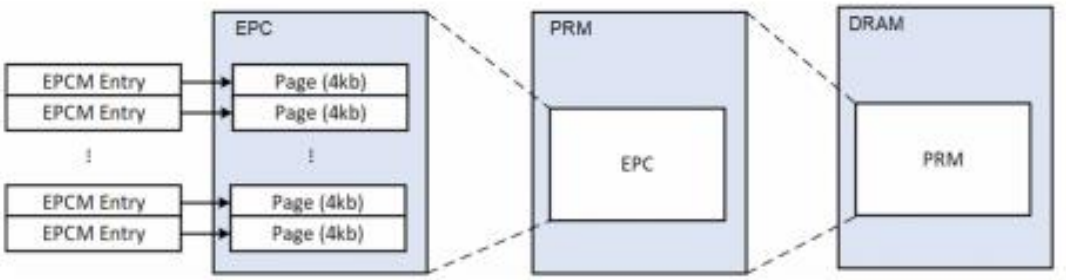
SGX

- **Organização da memória**
  - PRM - *Processor Reserved Memory*: nesta área de memória são executados os processos SGX.
  - Essa área é determinada pelo BIOS no momento do *boot.*
  - Composta por:
    - *Integrity tree*: responsável pelos TAGs de MAC (*crypto*) dos dados da EPC
    - EPC - Enclave Page Cache: contém páginas para diferentes enclaves (4Kb)



SGX

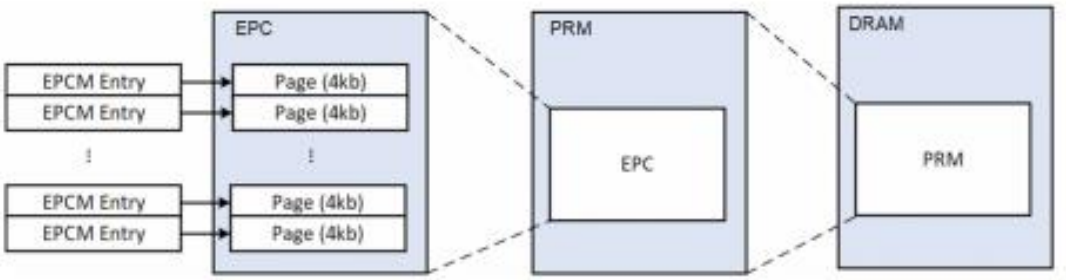
- **Organização da memória (cont.)**
  - Quem gerencia as EPCs é o OS (host ou no *hypervisor*) via instruções SGX para alocar páginas EPCs para os enclaves
  - A memória é protegida com o recurso *Memory Encryption Engine* (MEE) – extensão do *Integrated Memory Controller* (IMC).
  - Toda requisição RW de memória a um EPC é roteada do IMC ao MEE.
  - Todos acessos “non-enclave” são proibidos, incluindo acessos privilegiados: kernel, *hypervisor* e *System Management Mode* (SMM)



- EPCM - *Enclave Page Cache Map*: Dado que o software de gestão do EPC não é completamente confiável (*trusted*), o SGX registra as alocações de EPCs no EPCM.
- É um array onde cada entrada corresponde a um EPC

SGX

- **Organização da memória (cont.)**
  - SGX *Enclave Control Structure* (**SECS**): metadados de um enclave.
    - Cada um é armazenado em uma página EPC dedicada.
    - Contém o endereço virtual do enclave
  - *Thread Control Structure* (**TCS**): também é armazenado em um EPC dedicado. Contém informações sobre os threads.
    - Não pode ser acessado pelo enclave, mas pode ser acessado por instruções de *debug*.
  - Quando um processador lógico acessa essa área, ela se torna indisponível para os outros processadores lógicos



- *State Save Area* (**SSA**): área segura para armazenamento do contexto de execução. Utilizada quando há tratamento de exceção de *hardware*

SGX

- **Ciclo de Vida**
  - **Creation phase**: por meio da instrução **ECREATE**, uma **SECS** será criada em uma **EPC** livre
  - **Loading phase**: por meio de **EADD**, código e dados são carregados no enclave.
    - Já a instrução **EEXTEND** é utilizada para atualizar a **measurement** do enclave
  - **Initialization phase**: instrução **EINT** parar inicializar o enclave.
  - Uma estrutura **EINIT** (*Token Structure*) é retornada e o atributo **INIT** é alterado para *true*.
  - Nesse momento, parte das instruções não pode mais ser utilizada, como EADD, por exemplo.
  - **Teardown phase**: instrução **EREMOVE**, a página **ECP** que armazena a **SECS** é desalocada

Instruction	Description
ECREATE	Declare base and range, start build
EADD	Add 4k page
EEXTEND	Measure 256 bytes
EINT	Declare enclave built
EREMOVE	Remove page
EENTER	Enter enclave
ERESUME	Resume enclave
EEXIT	Leave enclave
AEX	Asynchronous enclave exit

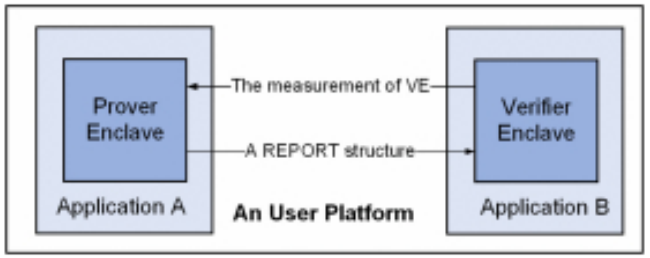
SGX

- **Runtime e Exceção**
  - Acessar enclave: **EENTER**. Se o **TCS** estiver livre (não *busy*), o processador lógico entra no "modo enclave" (SGX) e executará o código no enclave até receber a instrução **EEXIT**.
  - A partir deste momento, o controle é retornado do enclave ao processo host e o processador volta ao modo "normal"
  - Se ocorrer uma exceção quando o enclave está em execução, uma instrução **AEX** é chamada, o contexto armazenado no **SSA**, o controle retorna ao processo host, o processador volta ao modo normal e o código para tratamento de exceção é executado.
  - Após o tratamento da exceção, uma instrução **ERESUME** é chamada para retomar o processamento do enclave

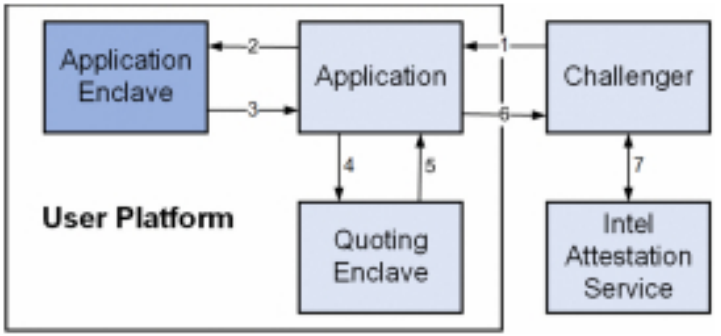
Instruction	Description
ECREATE	Declare base and range, start build
EADD	Add 4k page
EEXTEND	Measure 256 bytes
EINT	Declare enclave built
EREMOVE	Remove page
EENTER	Enter enclave
ERESUME	Resume enclave
EEXIT	Leave enclave
AEX	Asynchronous enclave exit

SGX

- **Software Attestation:**
  - Meio de verificação se o código foi corretamente instanciado. Pode ser feito por meio de:
    - **Local attestation**
    - **Remote attestation**
  - **Enclave Measurement:**
    - **Usa dois registradores:**
      - **MRENCLAVE**: representa a identidade, que é o hash das entradas das instruções ECREATE, EADD, e EEXTEND. Gerado após a instrução EINT
      - **MRSIGNER**: hash da chave pública da autoridade que que realiza o ***sealing***
  - **Data Sealing**: permite armazenar dados cifrados do enclave em disco

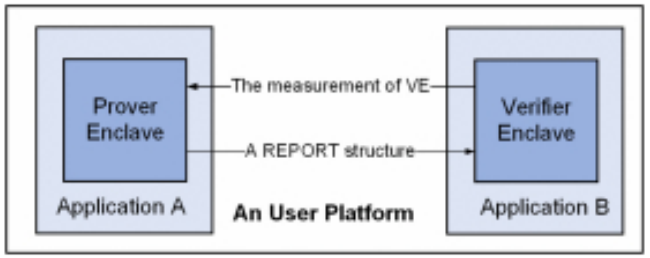


Local Attestation

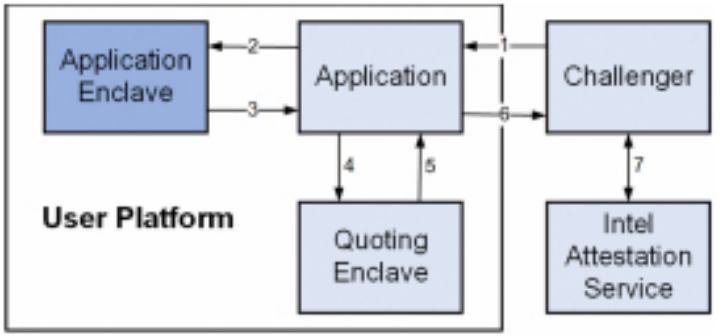


Remote Attestation

SGX



Local Attestation



Remote Attestation

Atestação Local:

1. **VE** solicita um **REPORT** ao **PE**
2. **PE** responde com um **REPORT**

Atestação Remota:

1. **Challenger** envia um desafio (**nonce**) ao **Application**
2. **Application** envia o desafio para o **AE**
3. **AE** gera:
   - um ***manifest*** contendo o **desafio** e uma **chave pública efêmera** (para comunicação futura)
   - um **REPORT** (contendo o **hash** do manifest)
4. **Application** recebe e encaminha ao **QE**, que autentica e valida os dados
5. **QE** transforma a estrutura de **REPORT** em **QUOTE** e utiliza **EPID*** para gerar envelope criptográfico final
6. Retorna ao **Challenger**
7. Validação do **REPORT** assinado junto ao serviço da Intel

SGX

- **Observações - FLC**
  - FLC (*Flexible Launch Control*) é uma característica (*feature*) dos processadores Intel com a tecnologia SGX
  - Ela provê um mecanismo para controlar a configuração de execução de um enclave, por exemplo: políticas de segurança para o enclave
  - Alguns softwares precisam de FLC para funcionar, que é o caso do framework Occlum (permite executar programas com pouca ou sem modificação dentro de enclaves)
  - Contudo, é possível que não esteja disponível para utilização, pois requer de suporte da BIOS da máquina.

SGX

- **Tecnologia relacionada: TDX (Trust Domain Extensions)**
  - Nova tecnologia da Intel para TEE, criada em 2021
  - Novo conceito - TDs (*Trust Domains*) - com paradigma diferente: proteção de VMs → Abordagem análoga a da AMD
    - VM é executada em região segura de memória
    - Importante para que provedores de nuvens possam manter as VMs de seus clientes isoladas
  - Não é possível executar aplicações usando SGX dentro de TDs
  - Mecanismo de atestação é reaproveitado do SGX (mesmo fluxo)

Ferramenta de Dump

Ferramenta de Dump

- Gcore
  - Permite realizar o *dump* de memória de um processo (PID) no linux
  - Capaz de realizar *dump* de endereços de memória marcados para impedimento de *dump* (VM_DONTDUMP)
- Toda memória das páginas de memória alocadas para o programa identificado pelo PID são colocadas em um arquivo.

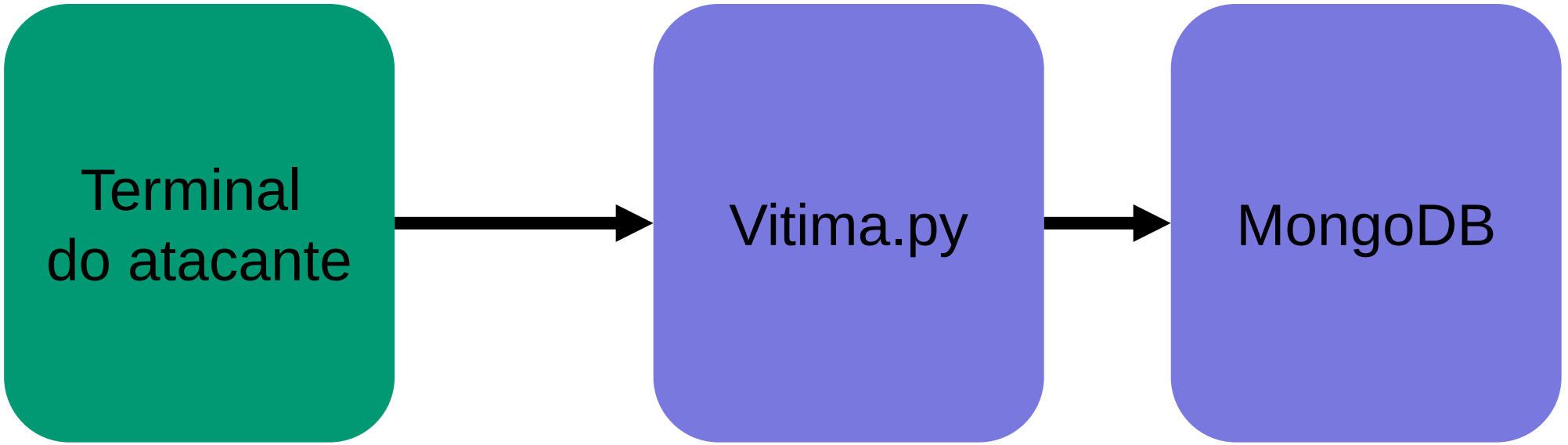
Ferramenta de Dump

Demonstração:

- **Cenário:**
  - Executar um servidor de autenticação desenvolvido em python
  - Executar um SGBD contendo os dados de autenticação
- **Ações do atacante (com acesso ao terminal e root):**
  - Solicitar que seja realizada a autenticação de um dos usuários existentes no sistema, mas com o *hash* incorreto:
    - O objetivo é fazer com que o sistema realize a busca no SGBD e carregue os dados na memória
  - Realizar *dump* para obter o *hash* do usuário

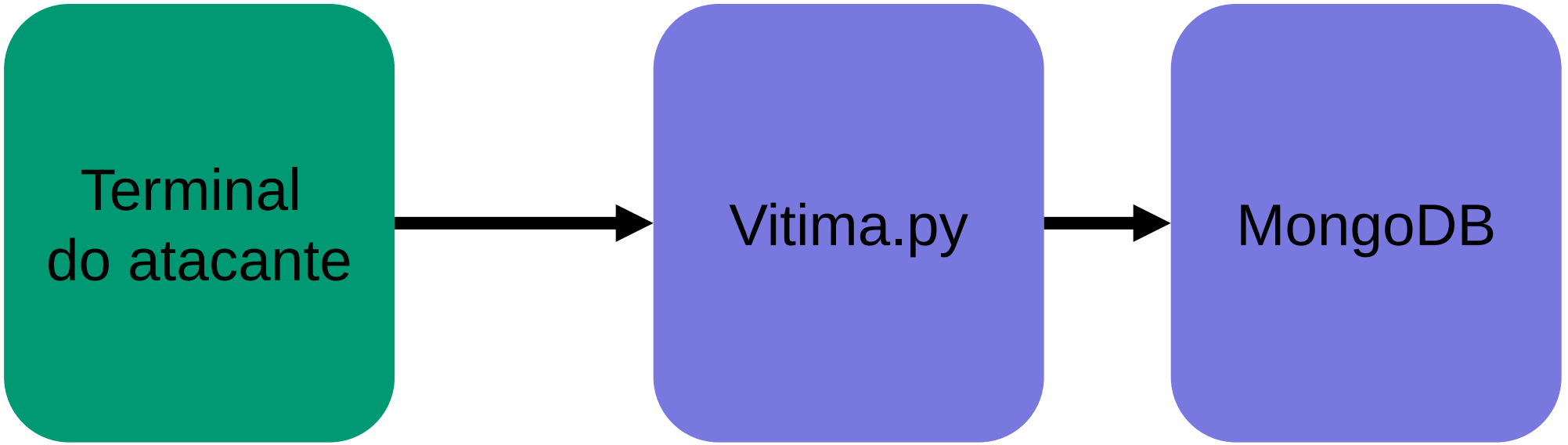
Ferramenta de Dump

Ferramenta de Dump



Solicita autenticação de
um usuário existente
(com *hash* incorreto)

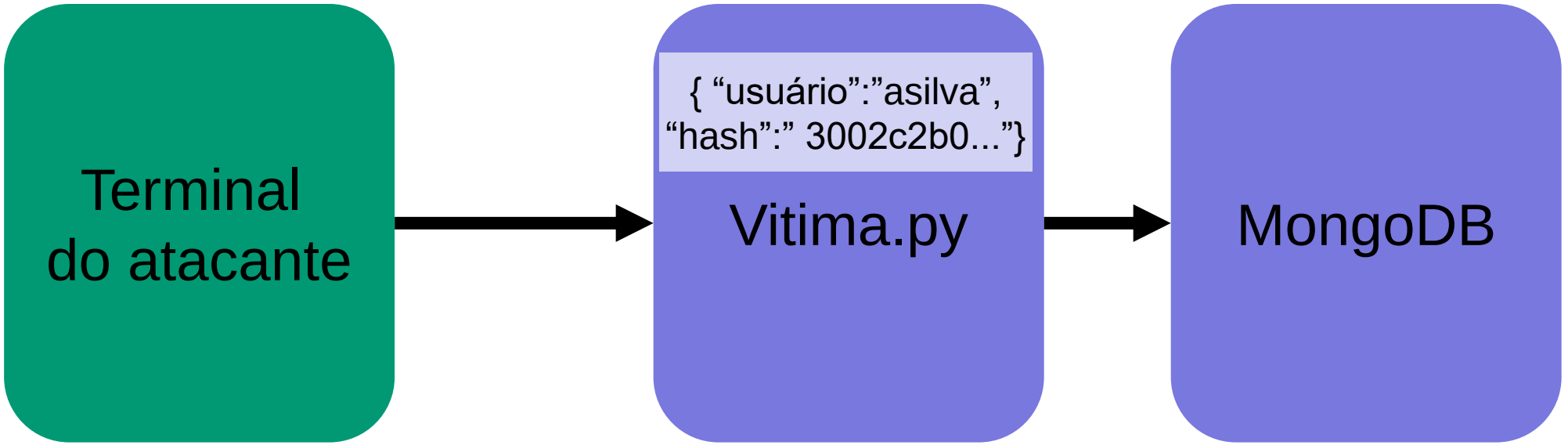
Ferramenta de Dump



Solicita autenticação de um usuário existente (com *hash* incorreto)

O servidor **vitima.py** obtém os dados do usuário para a tentativa de autenticação

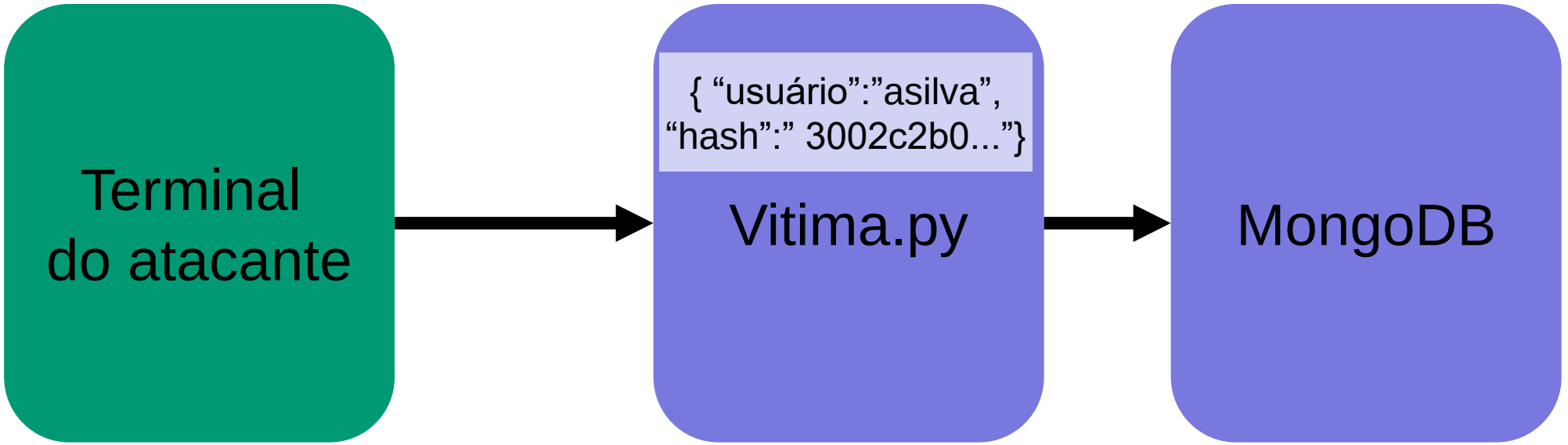
Ferramenta de Dump

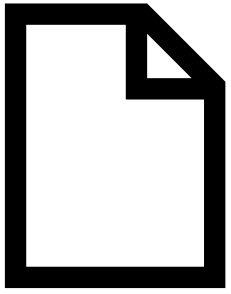


Solicita autenticação de um usuário existente (com *hash* incorreto)

O servidor **vitima.py** obtém os dados do usuário para a tentativa de autenticação

Ferramenta de Dump





Realiza dump de memória do processo para obter os dados de autenticação do usuário

Ferramenta de Dump

Demonstração da ferramenta de Dump

Fe

C

Hex Edit - [dump.bin.21758]

File Edit View Operations Template Aerial Tools Window Help

ASCII default

=asilva | 1280 E33D | 310.436.669

dump.bin.21758 | dump.bin.22472 | dump.bin.22473 | dump.bin.22474

```
            00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F  0123456789ABCDEF
1280 E270:  FF FF FF FF  FF FF FF FF  06 00 00 00  27 64 69 63  ............'dic
1280 E280:  00 00 00 00  00 00 00 00  70 0F A7 EB  01 7F 00 00  ........p.......
1280 E290:  30 6C B8 EB  01 7F 00 00  10 00 00 00  00 00 00 00  0l..............
1280 E2A0:  01 00 00 00  00 00 00 00  00 00 00 00  01 00 00 00  ................
1280 E2B0:  06 86 37 7C  D2 55 00 00  90 CA 9E EA  01 7F 00 00  ..7|.U..........
1280 E2C0:  98 CA 9E EA  01 7F 00 00  00 00 00 00  00 00 00 00  ................
1280 E2D0:  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
1280 E2E0:  10 00 00 00  00 00 00 00  01 00 00 00  00 00 00 00  ................
1280 E2F0:  D0 05 D5 EB  01 00 00 00  00 00 D5 EB  01 00 00 00  ................
1280 E300:  B0 C7 9E EA  01 7F 00 00  00 03 63 75  72 73 6F 72  ..........cursor
1280 E310:  00 A0 00 00  00 04 66 69  72 73 74 42  61 74 63 68  ......firstBatch
1280 E320:  00 65 00 00  00 03 30 00  5D 00 00 00  07 5F 69 64  .e....0.]...._id
1280 E330:  00 66 D2 1D  9C 04 C9 BD  B5 C2 4E B4  40 02 75 73  .f........N.@.us
1280 E340:  75 61 72 69  6F 00 07 00  00 00 61 73  69 6C 76 61  uario.....asilva
1280 E350:  00 02 68 61  73 68 00 29  00 00 00 33  30 30 32 63  ..hash.)...3002c
1280 E360:  32 62 30 34  32 63 64 33  33 61 31 64  63 39 62 31  2b042cd33a1dc9b1
1280 E370:  32 33 62 63  62 62 39 38  35 30 63 36  34 39 38 31  23bcbb9850c64981
1280 E380:  65 36 30 00  00 00 12 69  64 00 00 00  00 00 00 00  e60....id.......
1280 E390:  00 00 02 6E  73 00 16 00  00 00 61 75  74 65 6E 74  ...ns.....autent
1280 E3A0:  69 63 61 64  6F 72 2E 75  73 75 61 72  69 6F 73 00  icador.usuarios.
1280 E3B0:  00 01 6F 6B  00 00 00 00  00 00 00 F0  3F 00 00 00  ..ok........?...
1280 E3C0:  B0 D1 9F EA  01 7F 00 00  B0 F1 82 EA  01 7F 00 00  ................
1280 E3D0:  01 00 00 00  00 00 00 00  50 A1 FB 7C  D2 55 00 00  ........P..|.U..
1280 E3E0:  13 00 00 00  00 00 00 00  01 00 00 00  00 00 00 00  ................
1280 E3F0:  40 42 8A EA  01 7F 00 00  40 94 98 EA  01 7F 00 00  @B......@.......
1280 E400:  50 6B 87 EA  01 7F 00 00  20 00 00 00  D2 55 00 00  Pk...... ....U..
1280 E410:  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
1280 E420:  13 00 00 00  00 00 00 00  0F 00 00 00  04 00 00 00  ................
```

Ready | 12 | 02, 2, 002, 00000010, '^B' | 1280 E33Dh | 310.436.669 | Length: 1389 0598h | LE | RO

Autenticador em C++

Autenticador em C++

Objetivos:

- Simular parte de um servidor de autenticação
- Programa desenvolvido em C++ para facilitar sua execução dentro de um enclave SGX
- Extremamente simplificado para facilitar compreensão
- “Boneco” (*Toy*) para ser utilizado em mais de um experimento

Autenticador em C++

Diferenças do exemplo anterior (Autenticador em python):

- Não conecta a um database
- Mantém algumas variáveis de autenticação, isto é, usuários e seus *hashes* em memória
- Ataque é facilitado para pôr em prova as tecnologias defensoras
- Desta vez, o dump é sempre bem sucedido

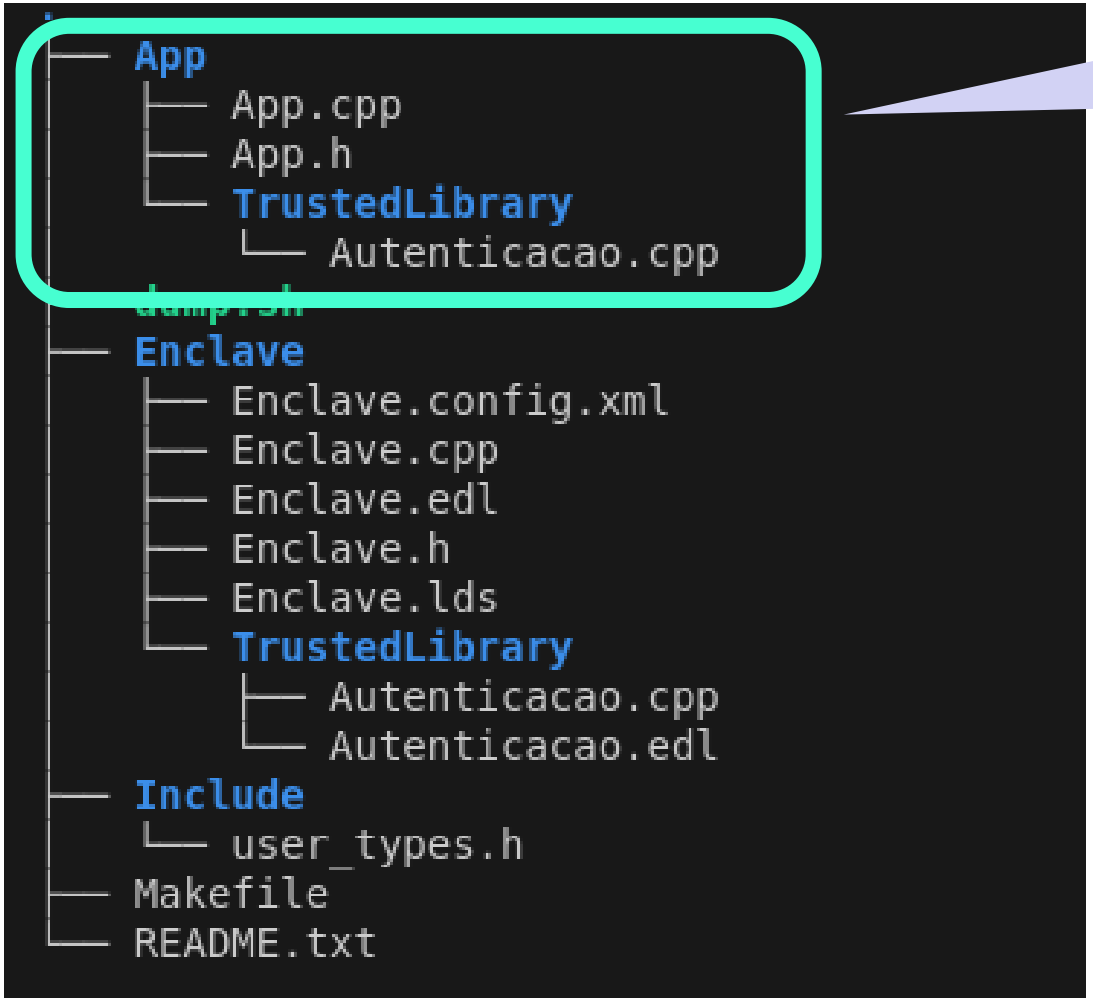
Autenticador em C++

- Estrutura do programa

```
.
├── App
│   ├── App.cpp
│   ├── App.h
│   └── TrustedLibrary
│       └── Autenticacao.cpp
├── dump.sh
├── Enclave
│   ├── Enclave.config.xml
│   ├── Enclave.cpp
│   ├── Enclave.edl
│   ├── Enclave.h
│   ├── Enclave.lds
│   └── TrustedLibrary
│       ├── Autenticacao.cpp
│       └── Autenticacao.edl
├── Include
│   └── user_types.h
├── Makefile
└── README.txt
```

Autenticador em C++

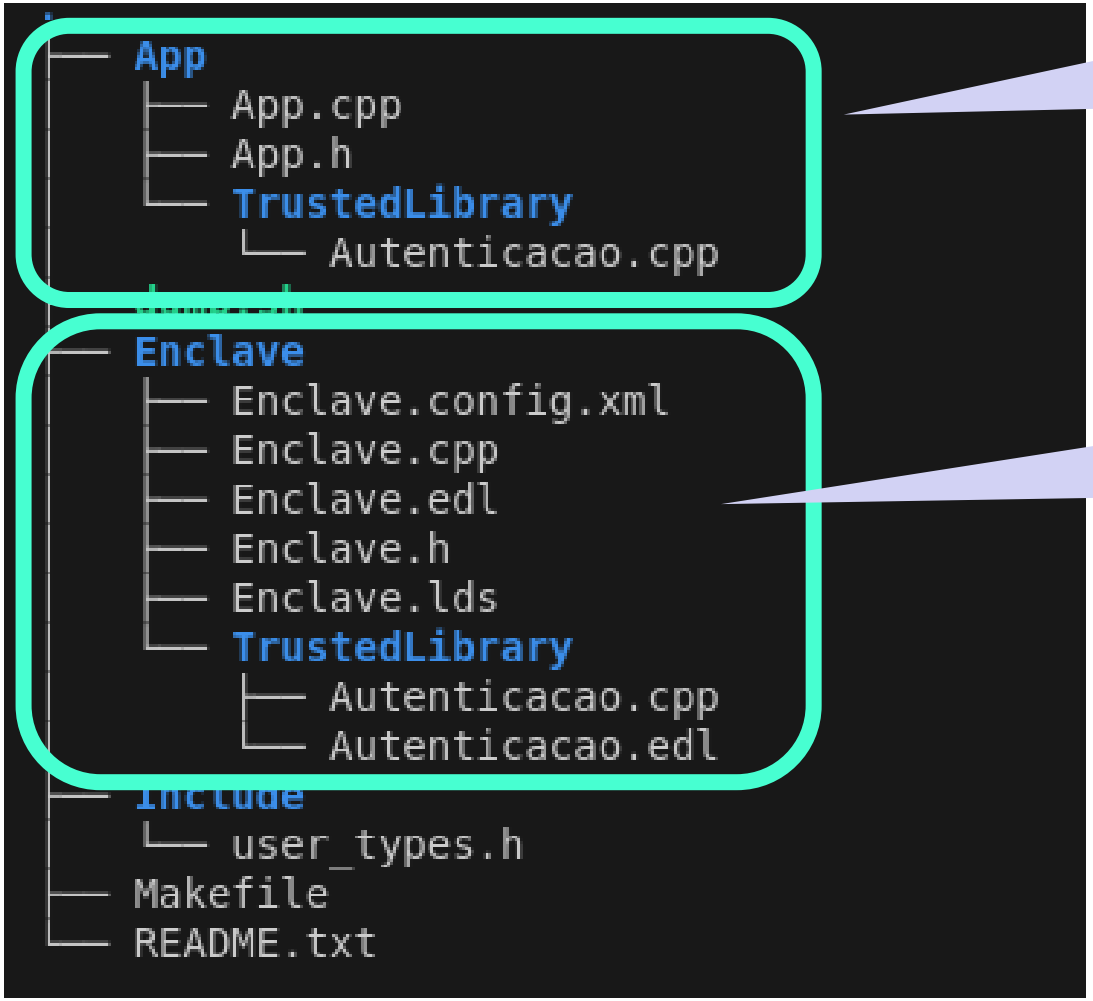
- Estrutura do programa



Programa Untrusted

Autenticador em C++

- Estrutura do programa

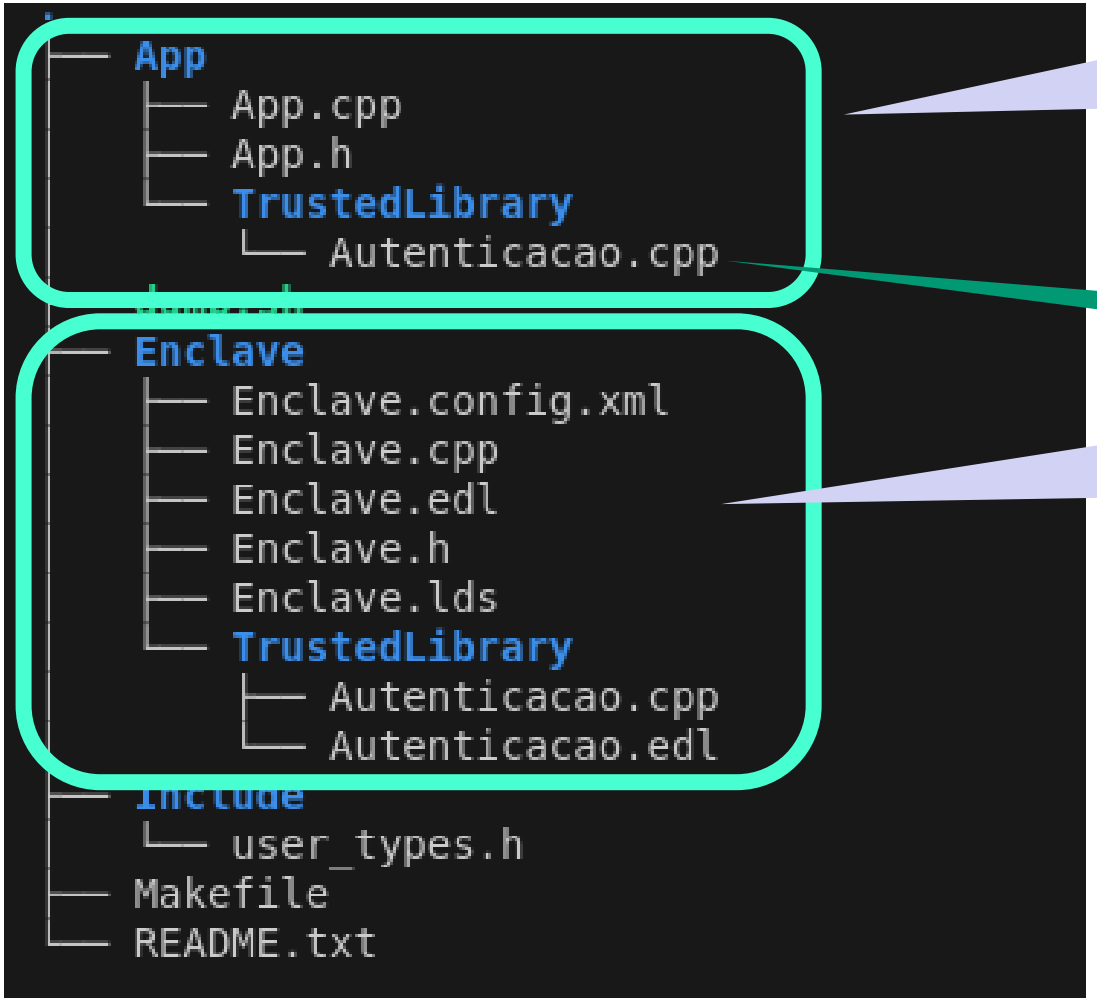


Programa
Untrusted

Programa
Trusted
(abrigado no enclave)

Autenticador em C++

- Estrutura do programa



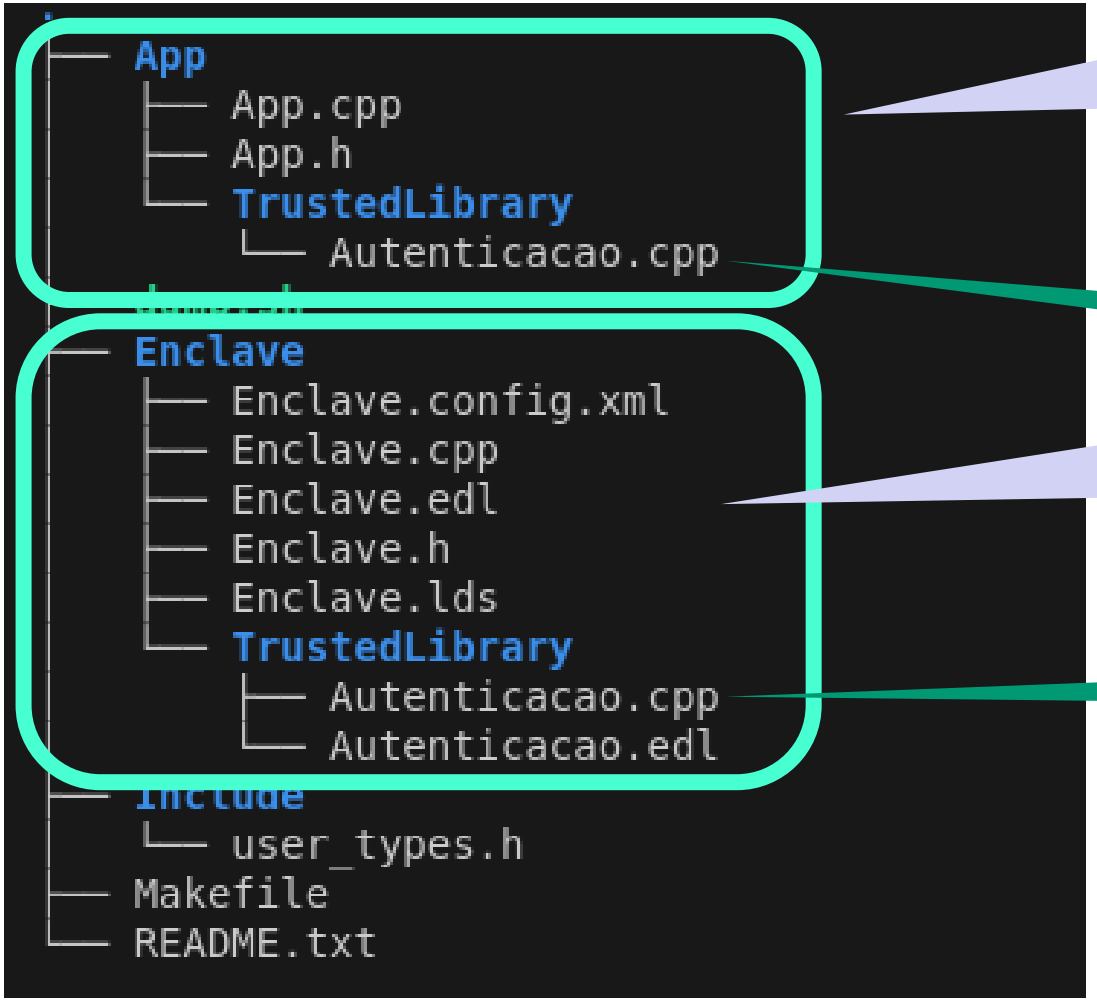
Programa Untrusted

Invoca a autenticação

Programa Trusted (abrigado no enclave)

Autenticador em C++

- Estrutura do programa



Programa Untrusted

Invoca a autenticação

Programa Trusted (abrigado no enclave)

Código específico da autenticação

Autenticador em C++

Visualização do código-fonte

Frameworks e Toolchains

Frameworks e Toolchains

- Nesse item são apresentados três ambientes de desenvolvimento e ferramentas:
  - **Intel SGX SDK**:
    - Conjunto de bibliotecas e outros softwares que permitem a criação de enclaves para processadores Intel dotados de tecnologia SGX
  - **Open Enclave SDK**
    - O Open Enclave SDK é um conjunto de desenvolvimento com software de código aberto voltado para a criação de uma única abstração unificada para enclaves, permitindo que os desenvolvedores construam aplicativos baseados em TEEs
  - **Occlum**
    - LibOS para permitir que um programa que não foi desenvolvido para Intel SGX consiga ser executado em um enclave
    - Obs.: LibOS – biblioteca capaz de se passar por um SO

Frameworks e Toolchains
Intel SGX SDK

Intel SGX SDK

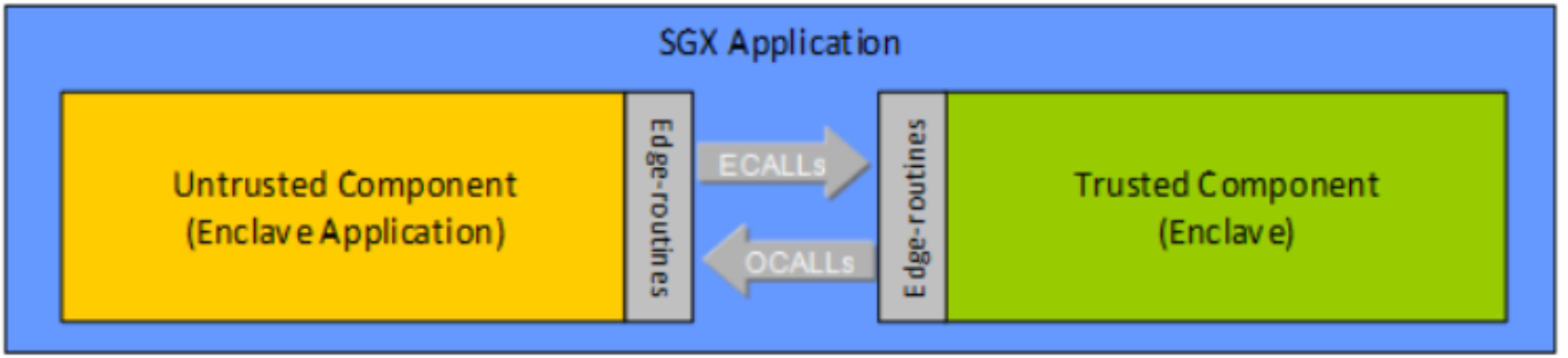
Como funciona um programa usando Intel SGX:

- Há duas partes de um programa:
  - **Untrusted**:
    - parte que fica exposta ao SO e aos demais programas em execução;
  - **Trusted**:
    - parte do programa que é executada exclusivamente pelo SGX
    - A parte *trusted* somente se comunica com a parte *untrusted* do mesmo programa
    - Não pode realizar nenhuma chamada externa

Intel SGX SDK

Como funciona um programa usando Intel SGX (cont.):

- A parte *untrusted* realiza chamadas do tipo ECALL (*Enclave call*) para acionar funções na parte *trusted*
- A parte *trusted* realiza OCALL (*Out call*) para acionar funções na parte *untrusted*.



- ECALL: chamada do programa *untrusted* ao enclave
- OCALL: chamada do enclave ao programa *untrusted*

Intel SGX SDK

Ambiente de Desenvolvimento:

- Linguagem C/C++, Rust e outras
- IDE incentivada: Eclipse
  - Há um plugin do eclipse desenvolvido pela Intel para facilitar
  - Alguns dos projetos de exemplo já estão preparados para abrir no Eclipse

Intel SGX SDK

Exemplo: Intel SGX SDK para Linux

- Software necessário para desenvolvimento:
  - Intel SGX Driver SGX (driver para usar instruções SGX)
    - Se a versão do kernel for >= 5.1, já estará presente nele
  - Intel SGX PSW - Platform Software:
    - Módulos que permitem a execução de enclaves (*runtime*) no Linux, a saber: Intel SGX Architecture Enclaves, Intel SGX Runtime System Library e Intel SGX Architecture Enclave Service Manager (AESM)
  - Intel SGX DCAP - Data Center Attestation Primitives
    - Funcionalidades para realização de atestação
  - Intel SDK (desenvolvimento): bibliotecas e software para auxiliar no desenvolvimento

Intel SGX SDK

Melhores Práticas:

- O enclave deve sempre checar os parâmetros recebidos para confirmar que são do tipo e tamanho corretos;
- Deve-se ter cuidado com a passagem de parâmetros por referência – os valores podem ser alterados pelo atacante após a validação do enclave.
  - Tanto para ECALL (recepção de parâmetro) como OCALL (recepção de retorno)
  - Por exemplo: uma *string* pode ter seu conteúdo alterado após ser recepcionado pelo enclave;
- Alguma chamada que o enclave realize (OCALL), talvez não produza o resultado ou execução desejada;

Intel SGX SDK

Melhores Práticas (cont.):

- Quando uma OCALL ocorre (*trusted* → *untrusted*), talvez o programa *untrusted* realize uma ECALL (*untrusted* → *trusted*) na mesma pilha de execução, o que é indesejável, pois pode expor alguma vulnerabilidade.
  - Para evitar que isso ocorra, o enclave deve controlar esse acesso internamente, por meio do uso de variáveis;

Autenticador em C++ com SGX

Autenticador em C++ com SGX

Roteiro de demonstração do SGX:

- Geração de executável com simulação (SGX desativado) e instruções de DEBUG desativadas
- Execução da ferramenta de dump (como *root*):
  - Resultado esperado: Ao avaliar o dump obtido, os dados sensíveis são encontrados
- Geração de executável com SGX ativado (em hardware) e com instruções de DEBUG desativadas
  - Resultado esperado: Ao avaliar o dump obtido, os dados sensíveis **não** são encontrados

Autenticador em C++ com SGX

Demonstração SGX desativado (simulação) e ativado

A

Hex Edit - [dump.bin.7272]

File Edit View Operations Template Aerial Tools Window Help

ASCII default | 14 2630 | 1.320.496

dump.bin.7272

```
          00 01 02 03  04 05 06 07  08 09 0A 0B  0C 0D 0E 0F  0123456789ABCDEF
014 2590: 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
014 25A0: 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
014 25B0: 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
014 25C0: 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
014 25D0: 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
014 25E0: 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
014 25F0: 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
014 2600: 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
014 2610: 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
014 2620: 02 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
014 2630: 68 61 73 68  2D 31 32 33  34 35 36 37  38 00 68 61  hash-12345678.ha
014 2640: 73 68 2D 61  62 63 64 65  36 37 38 00  68 61 73 68  sh-abcde678.hash
014 2650: 2D 31 32 33  35 34 61 62  63 65 64 00  6A 6F 61 6F  -12354abced.joao
014 2660: 00 6D 61 72  63 65 6C 61  00 72 65 6E  61 74 61 00  .marcela.renata.
014 2670: 48 20 F3 37  6A E6 B2 F2  03 4D 3B 7A  4B 48 A7 78  H .7j....M;zKH.x
014 2680: 53 39 AE 8C  93 AE 8F 3C  E4 32 DB 92  4D 0F 07 33  S9.....<.2..M..3
014 2690: 64 EA 4F 3F  A0 03 0C 36  38 3C 32 2D  4F 3A 8D 4F  d.O?...68<2-O:.O
014 26A0: 54 48 49 53  49 53 4F 57  4E 45 52 45  50 4F 43 48  THISISOWNEREPOCH
014 26B0: 0E 72 F0 FF  84 74 F0 FF  EA 76 F0 FF  F9 70 F0 FF  .r...t...v...p..
014 26C0: FC 6C F0 FF  00 00 00 00  00 00 00 00  00 00 00 00  .l..............
014 26D0: 00 11 22 33  44 55 66 77  88 99 AA BB  CC DD EE FF  .."3DUfw........
014 26E0: FF 00 FF 00  FF 00 FF 00  FF 00 FF 00  FF 00 FF 00  ................
014 26F0: AA 55 AA 55  AA 55 AA 55  AA 55 AA 55  AA 55 AA 55  .U.U.U.U.U.U.U.U
014 2700: BB AA BB EE  FF 00 00 DD  BB AA BB EE  FF 00 00 DD  ................
014 2710: 50 52 4F 56  49 53 49 4F  4E 53 45 41  4C 4B 45 59  PROVISIONSEALKEY
014 2720: A6 7F F0 FF  AF 7F F0 FF  B8 7F F0 FF  9D 7F F0 FF  ................
014 2730: 94 7F F0 FF  4C 81 F0 FF  1C 81 F0 FF  0E 81 F0 FF  ....L...........
014 2740: 0E 81 F0 FF  0E 81 F0 FF  0E 81 F0 FF  1C 81 F0 FF  ................
014 2750: 1C 81 F0 FF  AC 81 F0 FF  0E 81 F0 FF  0E 81 F0 FF  ................
```

Ready | 0 | 68, 104, 150, 01101000, 'h' | 14 2630h | 1.320.496 | Length: 12B B1D8h | LE | RO

Frameworks e Toolchains
Open Enclave SDK

Open Enclave SDK

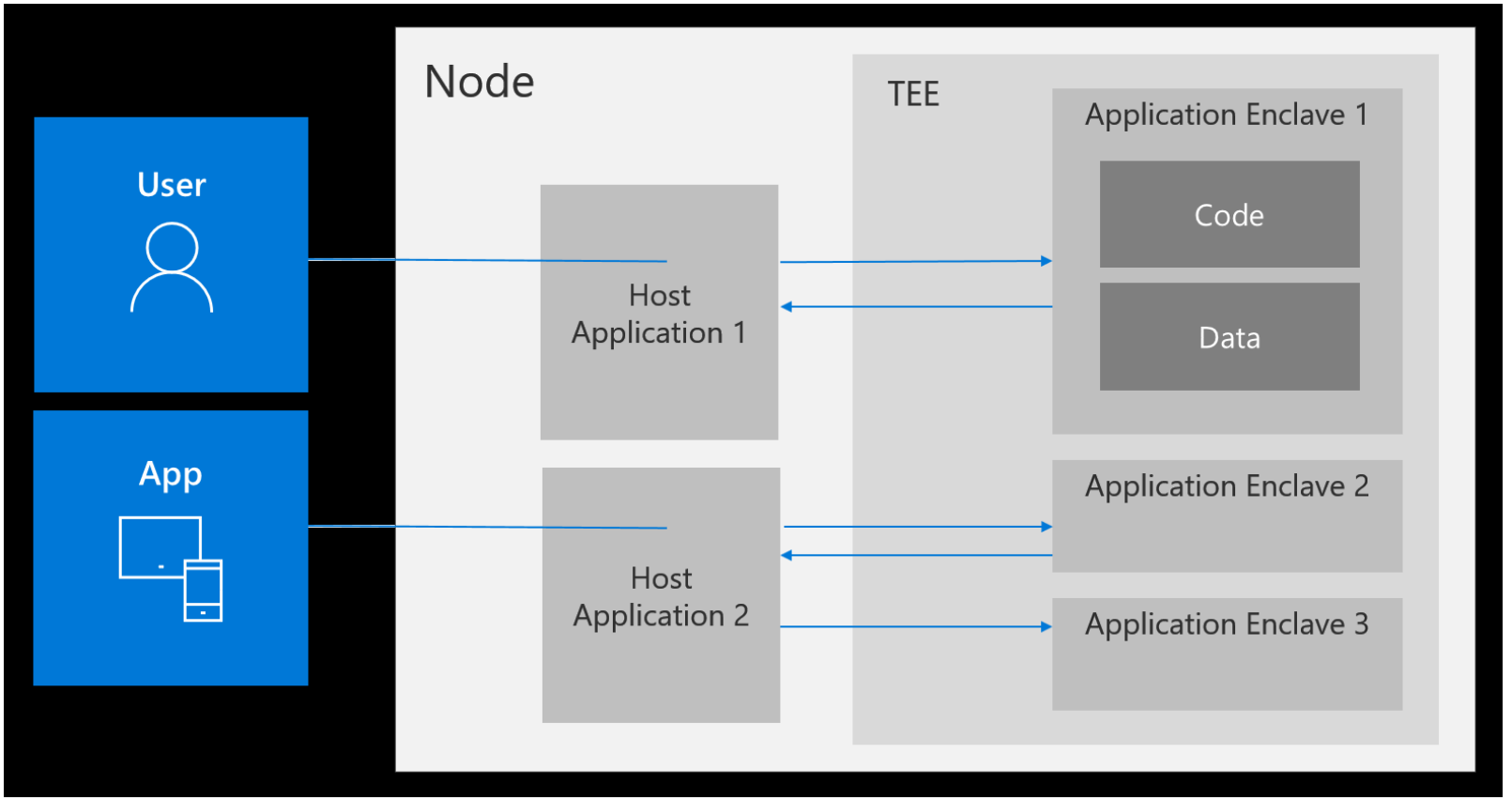
Objetivos:

- Um projeto do *Confidential Computing Consortium*
- Um SDK aberto para a criação de TEEs, de modo a abstrair a tecnologia utilizada.
- Atualmente capaz de abstrair algumas tecnologias como:
  - Intel SGX
  - OP-TEE OS (*Open Portable* TEE OS): TEE projetado para ser acoplado a um kernel Linux para ARM Cortex-A (com TrustZone)
- Linguagem de programação: C/C++
- O SDK está disponível para uso Windows e Ubuntu 20.04
- Código fonte disponível no GitHub

Open Enclave SDK

Definições:

- Componente *Unstruted*, denominado **Host**
- Componente *Trusted*, denominado **Enclave**



Open Enclave SDK

Funcionalidades:

- Criação e gestão de enclaves
- Encapsulamento de dados para chamadas do **Enclave** ao **Host** e vice-versa
- Funções para *Sealing*, isto é, para armazenamento cifrado de dados
- Bibliotecas de *runtime* para o enclave, para disponibilizar funções criptográficas ao **Enclave**
- *Measurement* e Atestação (para validação de um **Enclave**)

Open Enclave SDK

Observações:

- Permite um meio agnóstico de criar programas que utilizem enclaves.
  - Contudo, nesse momento só suporta Intel SGX e OP-TEE
- Framework criado pelo *Confidential Computing Consortium* , ou seja, é open source

Demonstração Open Enclave SDK

Demonstração Open Enclave SDK

- O Open Enclave SDK é uma abstração acima dos SDKs dos próprios fabricantes
- Permite escrever somente um código que poderá ser compilado para mais de um enclave sem alterações
- Para a demonstração, será usado um servidor com Intel SGX da Azure

Demonstração Open Enclave SDK

- **Passos da demonstração:**
- Download/execução de uma imagem Docker com os drivers SGX e SDK SGX
- Instalação de bibliotecas adicionais necessárias ao OESDK
- Clonagem local do repositório do Open Enclave no github
- Preparação, compilação e execução do exemplo “*hello world*” do repositório
- Verificação de quais módulos são executados fora e dentro do enclave
- Modificação do exemplo

```
azureuser@oe-sbseg:~$ sudo docker run -it --network host \
    --device /dev/sgx_enclave:/dev/sgx/enclave \
    --device /dev/sgx_provision:/dev/sgx/provision \
    --name oe -v \
    /home/azureuser:/home/larc occlum/occlum:latest-ubuntu20.04
Unable to find image 'occlum/occlum:latest-ubuntu20.04' locally
latest-ubuntu20.04: Pulling from occlum/occlum
17d0386c2fff: Pulling fs layer
c8c033a00815: Extracting [==>                                                ]  25.62MB/547.9MB
0effd0901446: Pulling fs layer
c8c033a00815: Download complete
bfab21665394: Download complete
4f4fb700ef54: Download complete
10c1d55cbdc0: Download complete
095f0b966da7: Download complete
14e68c6c7a98: Download complete
19cce46616e6: Download complete
8aa872cf8a72: Downloading [================================================> ]  508.2MB/518.1MB
db319e6d3229: Download complete
a53899402962: Download complete
db2789b1ba5e: Download complete
effd8313d7e6: Download complete
10d8064a2179: Download complete
2b7d9cf9970a: Download complete
8cdf5ebf7658: Download complete
30aa4058bb9c: Download complete
d33bd1cb5bb5: Download complete
f2640ded3361: Download complete
40780ca0298d: Downloading [=======================>                           ]  220.2MB/474.7MB
be7d615f0df6: Download complete
4247bb73954d: Download complete
4cf6806b371a: Downloading [=====================================>             ]  100.5MB/132MB
63bef61f6779: Waiting
4cf6806b371a: Pull complete
63bef61f6779: Pull complete
Digest: sha256:68c366e30800f2f3ff5a971d760b1aae574edc0823a0ddaba742e1ecf44ebaf6
Status: Downloaded newer image for occlum/occlum:latest-ubuntu20.04
root@oe-sbseg:~# aesm_service[11]: The server sock is 0x6244c3761810

root@oe-sbseg:~#
```

```
root@oe-sbseg:~# wget -qO - https://apt.llvm.org/llvm-snapshot.gpg.key | sudo apt-key add -
OK
root@oe-sbseg:~# echo "deb [arch=amd64] https://packages.microsoft.com/ubuntu/20.04/prod focal main" | sudo tee /etc/apt/sources.lis
t.d/msprod.list
deb [arch=amd64] https://packages.microsoft.com/ubuntu/20.04/prod focal main
root@oe-sbseg:~# wget -qO - https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add -
OK
root@oe-sbseg:~# sudo apt update
Get:1 https://apt.llvm.org/focal llvm-toolchain-focal-11 InRelease [6818 B]
Get:2 https://packages.microsoft.com/ubuntu/20.04/prod focal InRelease [3632 B]
Get:3 https://apt.llvm.org/focal llvm-toolchain-focal-11/main amd64 Packages [9012 B]
Get:4 http://security.ubuntu.com/ubuntu focal-security InRelease [128 kB]
Get:5 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 Packages [314 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 Packages [30.9 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [4016 kB]
Get:9 https://packages.microsoft.com/ubuntu/20.04/prod focal/main all Packages [2942 B]
Get:10 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [4008 kB]
Get:11 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [1273 kB]
Get:12 https://download.01.org/intel-sgx/sgx_repo/ubuntu focal InRelease [1239 B]
Get:13 http://archive.ubuntu.com/ubuntu focal-updates InRelease [128 kB]
Get:14 https://download.01.org/intel-sgx/sgx_repo/ubuntu focal/main amd64 Packages [144 kB]
Get:15 http://archive.ubuntu.com/ubuntu focal-backports InRelease [128 kB]
Get:16 http://archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [177 kB]
Get:17 http://archive.ubuntu.com/ubuntu focal/restricted amd64 Packages [33.4 kB]
Get:18 http://archive.ubuntu.com/ubuntu focal/main amd64 Packages [1275 kB]
Get:19 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [11.3 MB]
Get:20 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [4178 kB]
Get:21 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [4478 kB]
Get:22 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [1559 kB]
Get:23 http://archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 Packages [33.5 kB]
Get:24 http://archive.ubuntu.com/ubuntu focal-backports/main amd64 Packages [55.2 kB]
Get:25 http://archive.ubuntu.com/ubuntu focal-backports/universe amd64 Packages [28.6 kB]
Fetched 33.6 MB in 2s (15.0 MB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
82 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@oe-sbseg:~#
```

```
root@oe-sbseg:~# sudo apt -y install clang-11 libssl-dev gdb libsgx-enclave-common libsgx-quote-ex libprotobuf17 libsgx-dcap-ql libs
gx-dcap-ql-dev az-dcap-client open-enclave
Reading package lists... Done
Building dependency tree
Reading state information... Done
libprotobuf17 is already the newest version (3.6.1.3-2ubuntu5.2).
libprotobuf17 set to manually installed.
The following additional packages will be installed:
  binfmt-support libclang-common-11-dev libclang-cpp11 libclang1-11 libffi-dev libllvm11 libpfm4 libsgx-ae-epid
  libsgx-ae-id-enclave libsgx-ae-le libsgx-ae-pce libsgx-ae-qe3 libsgx-ae-qve libsgx-aesm-ecdsa-plugin libsgx-aesm-epid-plugin
  libsgx-aesm-launch-plugin libsgx-aesm-pce-plugin libsgx-aesm-quote-ex-plugin libsgx-dcap-quote-verify
  libsgx-dcap-quote-verify-dev libsgx-headers libsgx-launch libsgx-launch-dev libsgx-pce-logic libsgx-qe3-logic
  libsgx-quote-ex-dev libsgx-urts libssl1.1 libyaml-0-2 libz3-4 libz3-dev llvm-11 llvm-11-dev llvm-11-linker-tools llvm-11-runtime
  llvm-11-tools python3-pygments python3-yaml sgx-aesm-service
Suggested packages:
  clang-11-doc gdb-doc libssl-doc llvm-11-doc python-pygments-doc ttf-bitstream-vera
Recommended packages:
  libc-dbg gdbserver
The following NEW packages will be installed:
  az-dcap-client binfmt-support clang-11 libclang-common-11-dev libclang-cpp11 libclang1-11 libffi-dev libllvm11 libpfm4
  libyaml-0-2 libz3-4 libz3-dev llvm-11 llvm-11-dev llvm-11-linker-tools llvm-11-runtime llvm-11-tools open-enclave
  python3-pygments python3-yaml
The following packages will be upgraded:
  gdb libsgx-ae-epid libsgx-ae-id-enclave libsgx-ae-le libsgx-ae-pce libsgx-ae-qe3 libsgx-ae-qve libsgx-aesm-ecdsa-plugin
  libsgx-aesm-epid-plugin libsgx-aesm-launch-plugin libsgx-aesm-pce-plugin libsgx-aesm-quote-ex-plugin libsgx-dcap-ql
  libsgx-dcap-ql-dev libsgx-dcap-quote-verify libsgx-dcap-quote-verify-dev libsgx-enclave-common libsgx-headers libsgx-launch
  libsgx-launch-dev libsgx-pce-logic libsgx-qe3-logic libsgx-quote-ex libsgx-quote-ex-dev libsgx-urts libssl-dev libssl1.1
  sgx-aesm-service
28 upgraded, 20 newly installed, 0 to remove and 54 not upgraded.
Need to get 154 MB of archives.
After this operation, 725 MB of additional disk space will be used.
Get:1 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 az-dcap-client amd64 1.12.3 [164 kB]
Get:2 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 open-enclave amd64 0.19.4 [54.5 MB]
Get:3 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libssl-dev amd64 1.1.1f-1ubuntu2.23 [1586 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libssl1.1 amd64 1.1.1f-1ubuntu2.23 [1323 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal/main amd64 libyaml-0-2 amd64 0.2.2-1 [48.9 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-yaml amd64 5.3.1-1ubuntu0.1 [136 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/universe amd64 binfmt-support amd64 2.2.0-2 [58.2 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 gdb amd64 9.2-0ubuntu1~20.04.2 [3221 kB]
```

```
root@oe-sbseg:~# sudo apt -y install ninja-build
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  ninja-build
0 upgraded, 1 newly installed, 0 to remove and 54 not upgraded.
Need to get 107 kB of archives.
After this operation, 338 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/universe amd64 ninja-build amd64 1.10.0-1build1 [107 kB]
Fetched 107 kB in 0s (265 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package ninja-build.
(Reading database ... 58638 files and directories currently installed.)
Preparing to unpack .../ninja-build_1.10.0-1build1_amd64.deb ...
Unpacking ninja-build (1.10.0-1build1) ...
Setting up ninja-build (1.10.0-1build1) ...
Processing triggers for man-db (2.9.1-1) ...
root@oe-sbseg:~#
```

```
root@oe-sbseg:~# sudo apt-get -y install python3-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3-pip is already the newest version (20.0.2-5ubuntu1.10).
0 upgraded, 0 newly installed, 0 to remove and 54 not upgraded.
root@oe-sbseg:~#
```

```
root@oe-sbseg:~# sudo pip3 install cmake
Collecting cmake
  Downloading cmake-3.30.3-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (26.9 MB)
     |████████████████████████████████| 26.9 MB 42.9 MB/s
Installing collected packages: cmake
Successfully installed cmake-3.30.3
root@oe-sbseg:~#
```

```
root@oe-sbseg:/home/larc# git clone https://github.com/openenclave/openenclave
Cloning into 'openenclave'...
remote: Enumerating objects: 120914, done.
remote: Counting objects: 100% (905/905), done.
remote: Compressing objects: 100% (449/449), done.
remote: Total 120914 (delta 481), reused 715 (delta 404), pack-reused 120009 (from 1)
Receiving objects: 100% (120914/120914), 337.71 MiB | 88.72 MiB/s, done.
Resolving deltas: 100% (79043/79043), done.
root@oe-sbseg:/home/larc#
```

```
root@oe-sbseg:/home/larc# cd openenclave/samples/helloworld
root@oe-sbseg:/home/larc/openenclave/samples/helloworld# mkdir build && cd build
root@oe-sbseg:/home/larc/openenclave/samples/helloworld/build# cmake ..
-- The C compiler identification is Clang 11.1.0
-- The CXX compiler identification is Clang 11.1.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/clang-11 - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/clang++-11 - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Failed
-- Looking for pthread_create in pthreads
-- Looking for pthread_create in pthreads - not found
-- Looking for pthread_create in pthread
-- Looking for pthread_create in pthread - found
-- Found Threads: TRUE
-- Looking for crypto library - found
-- Looking for dl library - found
-- Performing Test OE_SPECTRE_MITIGATION_C_FLAGS_SUPPORTED
-- Performing Test OE_SPECTRE_MITIGATION_C_FLAGS_SUPPORTED - Success
-- Performing Test OE_SPECTRE_MITIGATION_CXX_FLAGS_SUPPORTED
-- Performing Test OE_SPECTRE_MITIGATION_CXX_FLAGS_SUPPORTED - Success
-- Configuring done (1.9s)
-- Generating done (0.0s)
-- Build files have been written to: /home/larc/openenclave/samples/helloworld/build
root@oe-sbseg:/home/larc/openenclave/samples/helloworld/build#
```

```
[ 40%] Built target helloworld_host
[ 50%] Generating helloworld_t.h, helloworld_t.c, helloworld_args.h
Generating edge routine, for the Open Enclave SDK.
Processing /home/larc/openenclave/samples/helloworld/helloworld.edl.
Processing /opt/openenclave/include/openenclave/edl/syscall.edl.
Processing /opt/openenclave/include/openenclave/edl/epoll.edl.
Processing /opt/openenclave/include/openenclave/edl/fcntl.edl.
Processing /opt/openenclave/include/openenclave/edl/ioctl.edl.
Processing /opt/openenclave/include/openenclave/edl/poll.edl.
Processing /opt/openenclave/include/openenclave/edl/signal.edl.
Processing /opt/openenclave/include/openenclave/edl/socket.edl.
Processing /opt/openenclave/include/openenclave/edl/time.edl.
Processing /opt/openenclave/include/openenclave/edl/unistd.edl.
Processing /opt/openenclave/include/openenclave/edl/utsname.edl.
Processing /opt/openenclave/include/openenclave/edl/sgx/platform.edl.
Processing /opt/openenclave/include/openenclave/edl/sgx/attestation.edl.
Processing /opt/openenclave/include/openenclave/edl/sgx/cpu.edl.
Processing /opt/openenclave/include/openenclave/edl/sgx/debug.edl.
Processing /opt/openenclave/include/openenclave/edl/sgx/thread.edl.
Processing /opt/openenclave/include/openenclave/edl/sgx/switchless.edl.
Processing /opt/openenclave/include/openenclave/edl/sgx/tdx_verification.edl.
Success.
[ 60%] Building C object enclave/CMakeFiles/enclave.dir/enc.c.o
[ 70%] Building C object enclave/CMakeFiles/enclave.dir/helloworld_t.c.o
[ 80%] Linking C executable enclave
[ 80%] Built target enclave
[ 90%] Generating private.pem, public.pem
Generating RSA private key, 3072 bit long modulus (2 primes)
.....++++
.....................................................................++++
e is 3 (0x03)
writing RSA key
[100%] Generating enclave/enclave.signed
Created /home/larc/openenclave/samples/helloworld/build/enclave/enclave.signed
[100%] Built target sign
Hello world from the enclave
Enclave called into host to print: Hello World!
[100%] Built target run
root@oe-sbseg:/home/larc/openenclave/samples/helloworld/build#
```

```
root@oe-sbseg:/home/larc/openenclave/samples/helloworld# vi host/host.c
root@oe-sbseg:/home/larc/openenclave/samples/helloworld# vi enclave/enc.c
```

```
// Copyright (c) Open Enclave SDK contributors.
// Licensed under the MIT License.

#include <openenclave/host.h>
#include <stdio.h>

// Include the untrusted helloworld header that is generated
// during the build. This file is generated by calling the
// sdk tool oeedger8r against the helloworld.edl file.
#include "helloworld_u.h"

bool check_simulate_opt(int* argc, const char* argv[])
{
    for (int i = 0; i < *argc; i++)
    {
        if (strcmp(argv[i], "--simulate") == 0)
        {
            fprintf(stdout, "Running in simulation mode\n");
            memmove(&argv[i], &argv[i + 1], (*argc - i) * sizeof(char*));
            (*argc)--;
            return true;
        }
    }
    return false;
}

// This is the function that the enclave will call back into to
// print a message.
void host_helloworld()
{
    fprintf(stdout, "Enclave called into host to print: Hello World from LARC!\n");
}

int main(int argc, const char* argv[])
{
    oe_result_t result;
    int ret = 1;
    oe_enclave_t* enclave = NULL;
-- INSERT --
```

```
// Copyright (c) Open Enclave SDK contributors.
// Licensed under the MIT License.

#include <stdio.h>

// Include the trusted helloworld header that is generated
// during the build. This file is generated by calling the
// sdk tool oeedger8r against the helloworld.edl file.
#include "helloworld_t.h"

// This is the function that the host calls. It prints
// a message in the enclave before calling back out to
// the host to print a message from there too.
void enclave_helloworld()
{
    // Print a message from the enclave. Note that this
    // does not directly call fprintf, but calls into the
    // host and calls fprintf from there. This is because
    // the fprintf function is not part of the enclave
    // as it requires support from the kernel.
    fprintf(stdout, "Hello world from the enclave by LARC\n");

    // Call back into the host
    oe_result_t result = host_helloworld();
    if (result != OE_OK)
    {
        fprintf(
            stderr,
            "Call to host_helloworld failed: result=%u (%s)\n",
            result,
            oe_result_str(result));
    }
}
~
~
~
~
~
:wq
```

```
root@oe-sbseg:/home/larc/openenclave/samples/helloworld# vi host/host.c
root@oe-sbseg:/home/larc/openenclave/samples/helloworld# vi enclave/enc.c
root@oe-sbseg:/home/larc/openenclave/samples/helloworld# cd build/
root@oe-sbseg:/home/larc/openenclave/samples/helloworld/build#
```

```
root@oe-sbseg:/home/larc/openenclave/samples/helloworld# vi host/host.c
root@oe-sbseg:/home/larc/openenclave/samples/helloworld# vi enclave/enc.c
root@oe-sbseg:/home/larc/openenclave/samples/helloworld# cd build/
root@oe-sbseg:/home/larc/openenclave/samples/helloworld/build# make run
[ 10%] Building C object host/CMakeFiles/helloworld_host.dir/host.c.o
[ 20%] Linking C executable helloworld_host
[ 40%] Built target helloworld_host
[ 50%] Building C object enclave/CMakeFiles/enclave.dir/enc.c.o
[ 60%] Linking C executable enclave
[ 80%] Built target enclave
[ 90%] Generating enclave/enclave.signed
Created /home/larc/openenclave/samples/helloworld/build/enclave/enclave.signed
[100%] Built target sign
Hello world from the enclave by LARC
Enclave called into host to print: Hello World from LARC!
[100%] Built target run
root@oe-sbseg:/home/larc/openenclave/samples/helloworld/build#
```

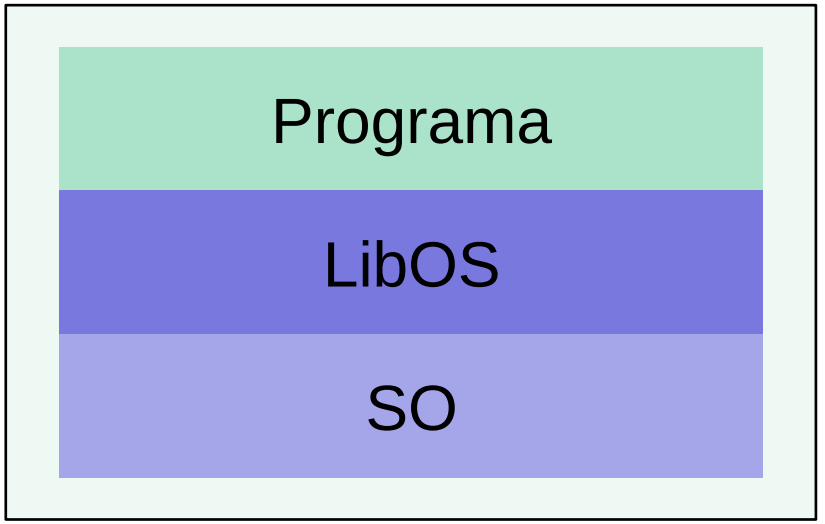
Frameworks e Toolchains
Occlum

Occlum

- O Occlum é um projeto do *Confidential Computing Consortium* que visa criar um ambiente seguro para a execução de aplicativos.
- Ele oferece isolamento de memória e processos, permitindo que aplicativos sejam executados em um ambiente confidencial.
- O Occlum é especialmente útil para proteger dados sensíveis, como informações pessoais ou segredos comerciais, em sistemas de computação em nuvem ou dispositivos edge.
- Ele é uma ferramenta disponibilizada para garantir a privacidade e a segurança dos dados em ambientes de computação confidencial (TEEs)
- Funciona como uma LibOS, ou seja, como uma biblioteca que “se passa” por um SO

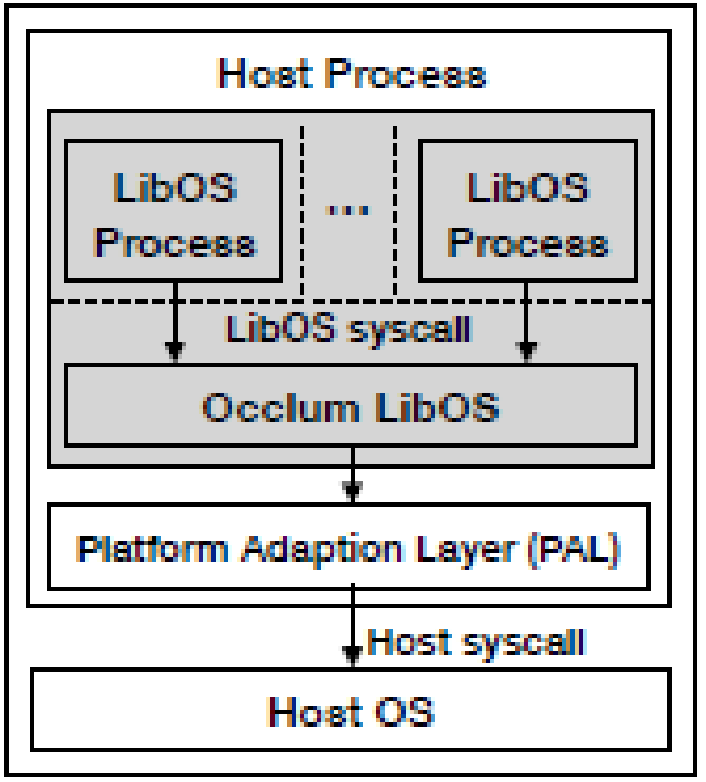
Occlum

- **O que é uma LibOS?**
  - Intermedia as *system calls* e interrupções entre:
    - Software ←→ SO e hardware (quando aplicável)
  - O software que utiliza a LibOS “pensa” que está usando o SO
  - Outras soluções que utilizam LibOS:
    - Haven [21]
    - Scone [19]
    - Panoply [52]
    - Graphene-SGX [55].



Occlum

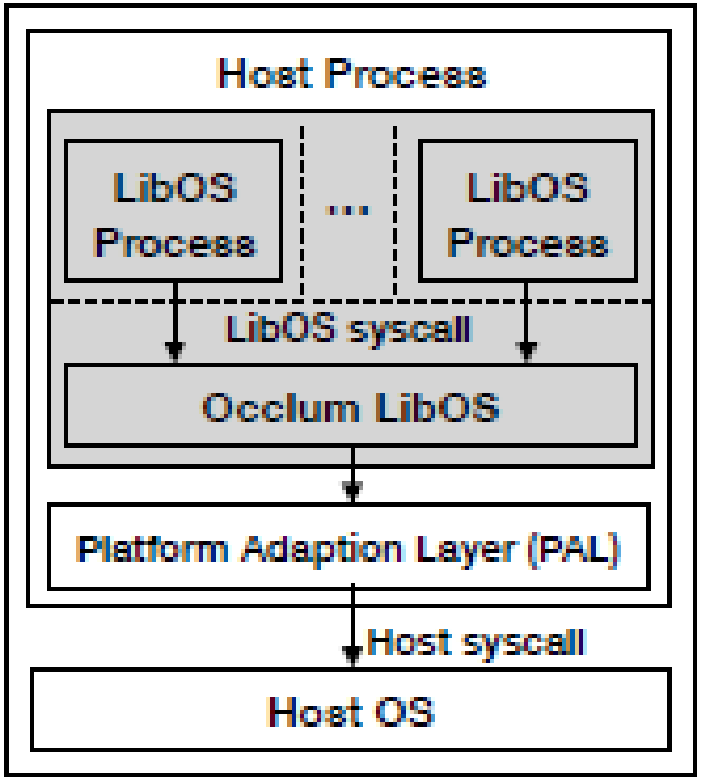
- **Motivação:**
  - Muito “trabalho” na conversão de um código para funcionar em SGX, pois é necessário que existam duas partes do programa: *enclave (trusted)* e *non-enclave (untrusted)*.
  - Portanto, oferecem o Occlum (Occlum LibOS) como uma solução para códigos legados



(a) Occlum LibOS

Occlum

- **Carregamento de um programa binário no Occlum:**
  - **Disassembly**: o programa é binário desconstruído e suas instruções alcançáveis são mapeadas
  - **Verificação do conjunto de instruções**: realização de busca pelas instruções que são privilegias, que deveriam ser somente executadas pela LibOS, como instruções SGX entre outras
  - **Verificação de transferência de controle**: Avaliação dos momentos nos quais há transferência de controle - destinos de operações “jumps”



(a) Occlum LibOS

Occlum

Observações:

- Permite executar programas dentro de um enclave Intel SGX, mesmo que o programa não tenha sido criado para isso. Ou seja, um programa que não tenha partes *Trusted* e *Untrusted* distintas
  - Há exceções! O MySQL, por exemplo, necessita ser alterado e recompilado
- Muita facilidade no uso: o fabricante disponibiliza imagens Docker, qual já estão presentes:
  - Software da Intel (SGX SDK)
  - Software Occlum pronto para criar ambientes e executar
  - Demonstrações

Demonstração Occlum

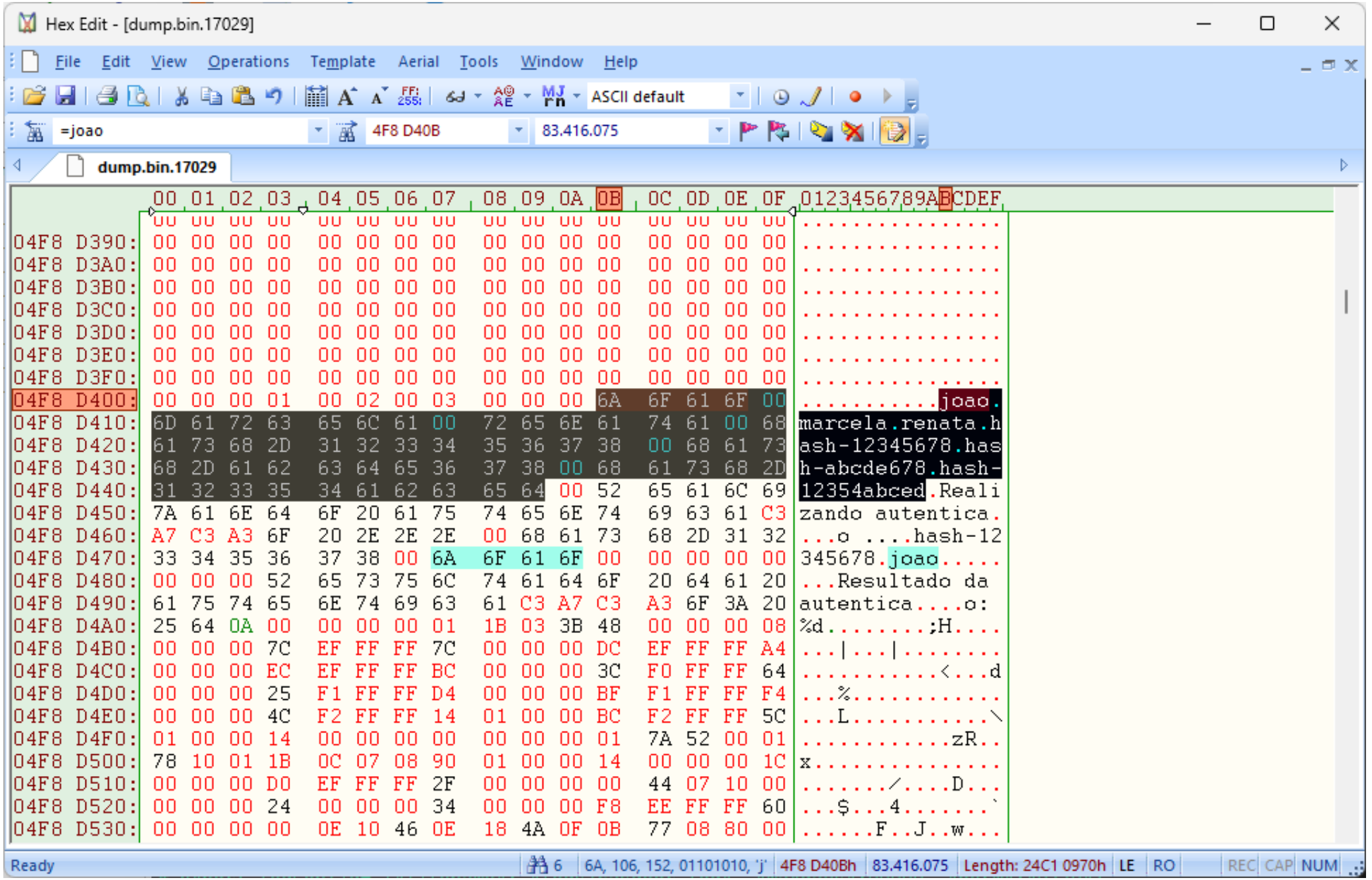
Demonstração Occlum

Geração de Occlum (prática):

- Utilização de imagem Docker oferecida pelo fabricante:
  - Facilita a o uso de Intel SGX SDK, pois já está presente na imagem
  - Contém as ferramentas de geração para abrigar um programa no Occlum:
    - `occlum new:` Criação de workspace do occlum
    - `copy_bom:` Cópia de arquivos para imagem, de acordo com a configuração de perfil
    - `occlum build:` Compila e gera o ambiente de execução
    - `occlum run:` Executa o programa indicado
- Realização de testes com o autenticador

Demonstração Occlum

Realização da demonstração

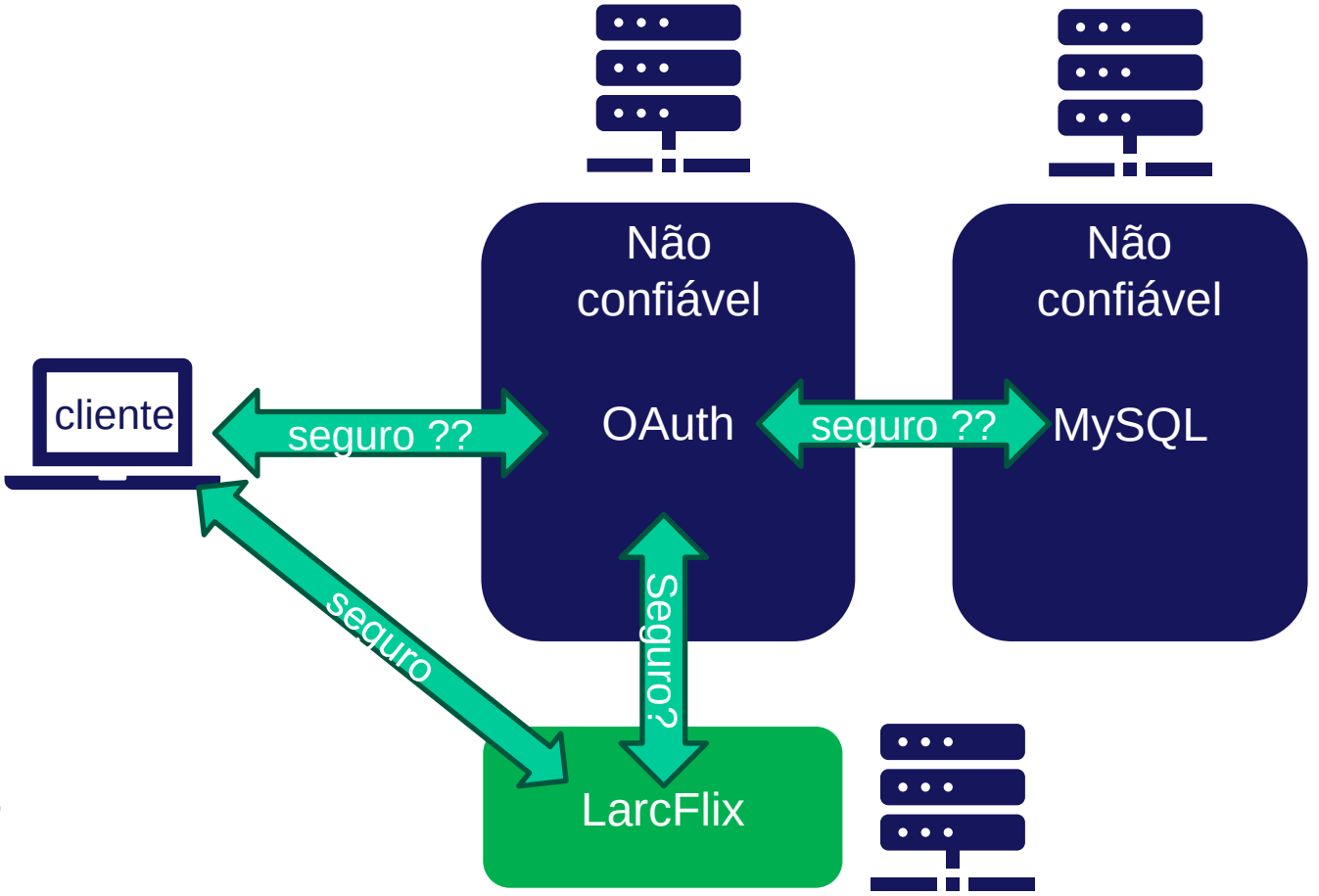

Hex Edit - [dump.bin.17029]
File Edit View Operations Template Aerial Tools Window Help
ASCII default
=joao
4F8 D40B
83.416.075
dump.bin.17029
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 0123456789ABCDEF
04F8 D390: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
04F8 D3A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
04F8 D3B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
04F8 D3C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
04F8 D3D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
04F8 D3E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
04F8 D3F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
04F8 D400: 00 00 00 01 00 02 00 03 00 00 00 6A 6F 61 6F 00 ....joao.
04F8 D410: 6D 61 72 63 65 6C 61 00 72 65 6E 61 74 61 00 68 marcela.renata.h
04F8 D420: 61 73 68 2D 31 32 33 34 35 36 37 38 00 68 61 73 ash-12345678.has
04F8 D430: 68 2D 61 62 63 64 65 36 37 38 00 68 61 73 68 2D h-abcde678.hash-
04F8 D440: 31 32 33 35 34 61 62 63 65 64 00 52 65 61 6C 69 12354abced.Reali
04F8 D450: 7A 61 6E 64 6F 20 61 75 74 65 6E 74 69 63 61 C3 zando autentica.
04F8 D460: A7 C3 A3 6F 20 2E 2E 2E 00 68 61 73 68 2D 31 32 ...o hash-12
04F8 D470: 33 34 35 36 37 38 00 6A 6F 61 6F 00 00 00 00 00 345678.joao.....
04F8 D480: 00 00 00 52 65 73 75 6C 74 61 64 6F 20 64 61 20 ...Resultado da
04F8 D490: 61 75 74 65 6E 74 69 63 61 C3 A7 C3 A3 6F 3A 20 autentica....o:
04F8 D4A0: 25 64 0A 00 00 00 00 01 1B 03 3B 48 00 00 00 08 %d........;H....
04F8 D4B0: 00 00 00 7C EF FF FF 7C 00 00 00 DC EF FF FF A4 ...|...|........
04F8 D4C0: 00 00 00 EC EF FF FF BC 00 00 00 3C F0 FF FF 64 ....<...d
04F8 D4D0: 00 00 00 25 F1 FF FF D4 00 00 00 BF F1 FF FF F4 ...%............
04F8 D4E0: 00 00 00 4C F2 FF FF 14 01 00 00 BC F2 FF FF 5C ...L...........\
04F8 D4F0: 01 00 00 14 00 00 00 00 00 00 00 01 7A 52 00 01 ....zR..
04F8 D500: 78 10 01 1B 0C 07 08 90 01 00 00 14 00 00 00 1C x...............
04F8 D510: 00 00 00 D0 EF FF FF 2F 00 00 00 00 44 07 10 00 ..../....D...
04F8 D520: 00 00 00 24 00 00 00 34 00 00 00 F8 EE FF FF 60 ...$...4.......`
04F8 D530: 00 00 00 00 0E 10 46 0E 18 4A 0F 0B 77 08 80 00 ......F..J..w...
Ready
6
6A, 106, 152, 01101010, 'j'
4F8 D40Bh
83.416.075
Length: 24C1 0970h
LE
RO
REC CAP NUM

Exemplo de Autenticador na Nuvem: Case OAuth e MySQL

Case OAuth e MySQL

Desafio:

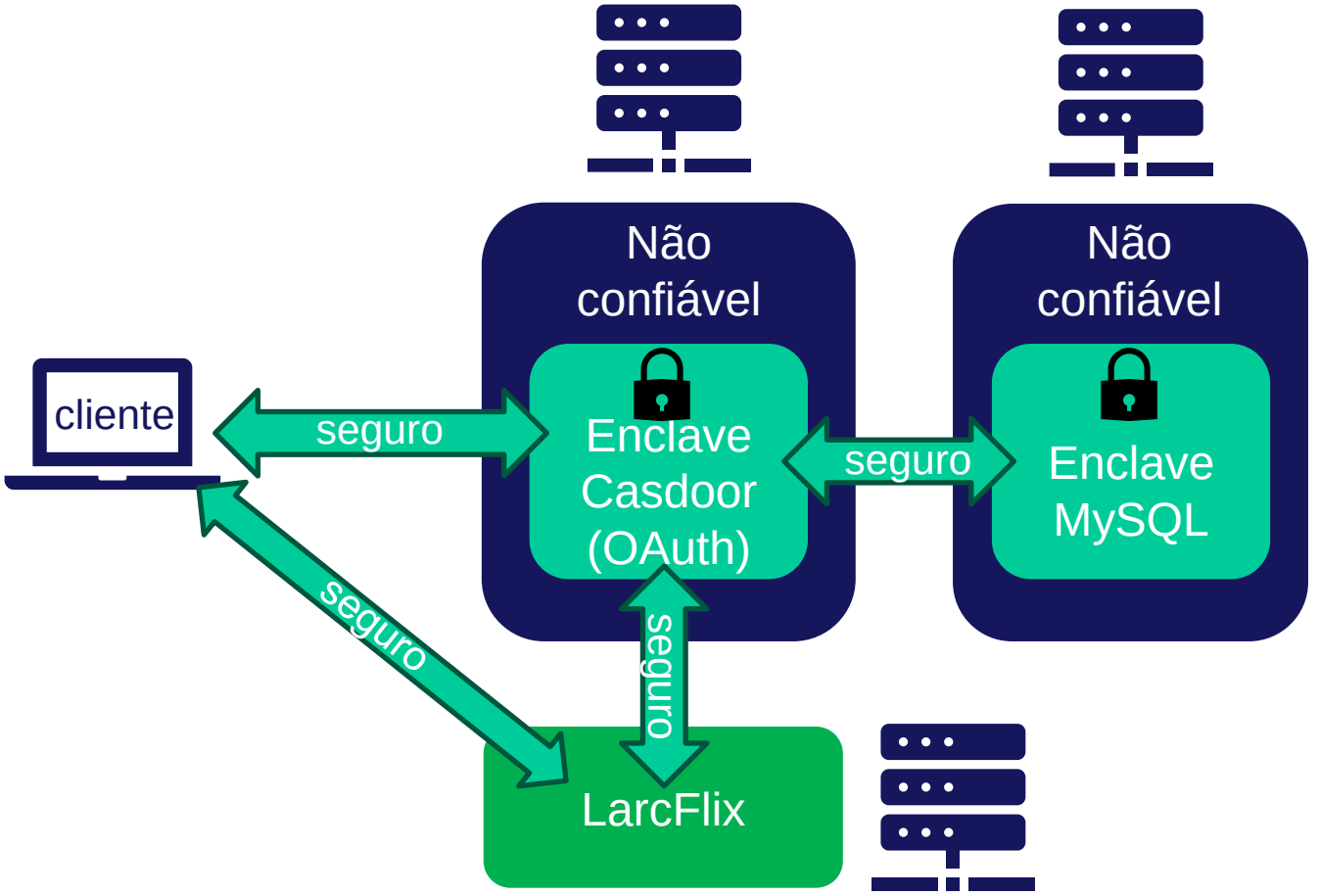
- Montar um autenticador OAuth utilizando computadores (ou VMs) não confiáveis.
- Como realizar a autenticação de um usuário em computadores não confiáveis?
- Como consultar o SGBD em um computador não confiável?



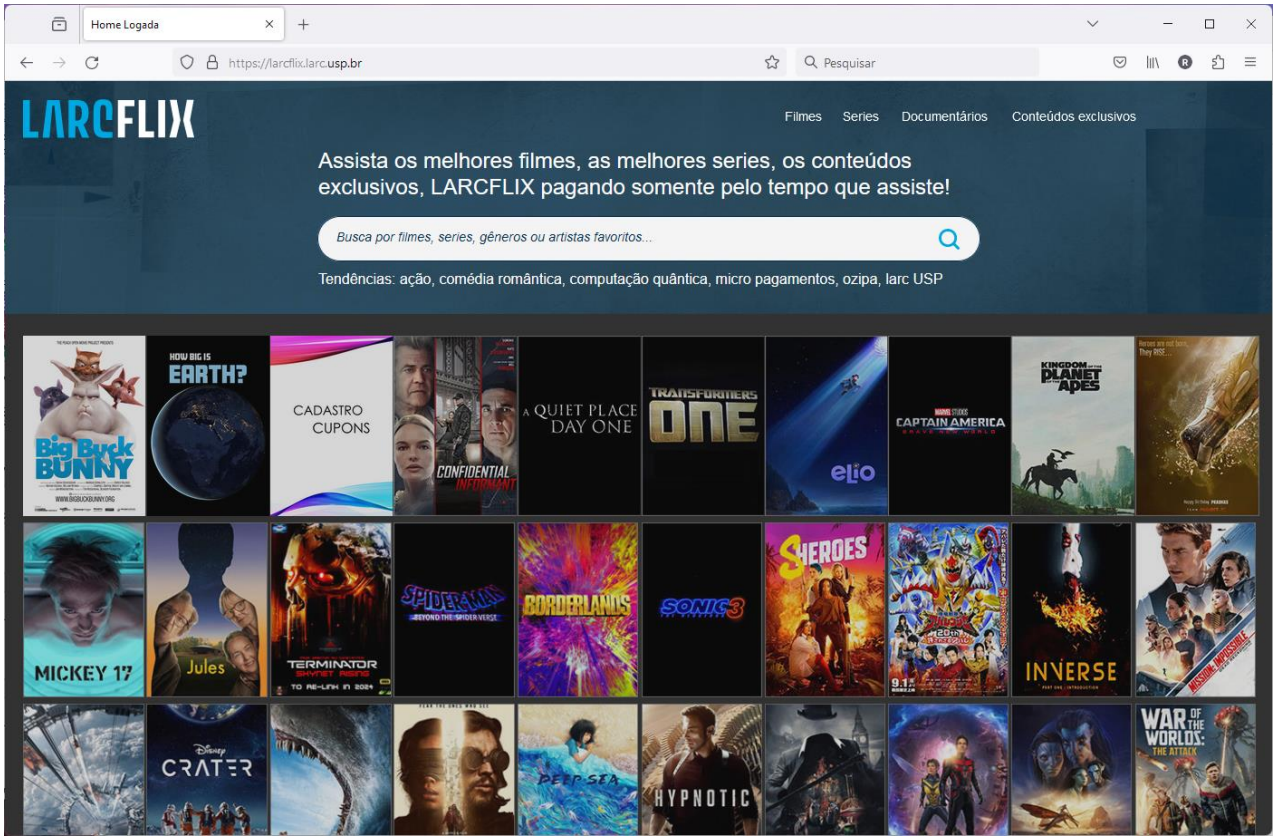
Case OAuth e MySQL

Proposta:

- Utilização de enclaves
- Tecnologia SGX - Intel
- Framework Occlum
  - Servidor Oauth: software Casdoor
  - SGBD: software MySQL

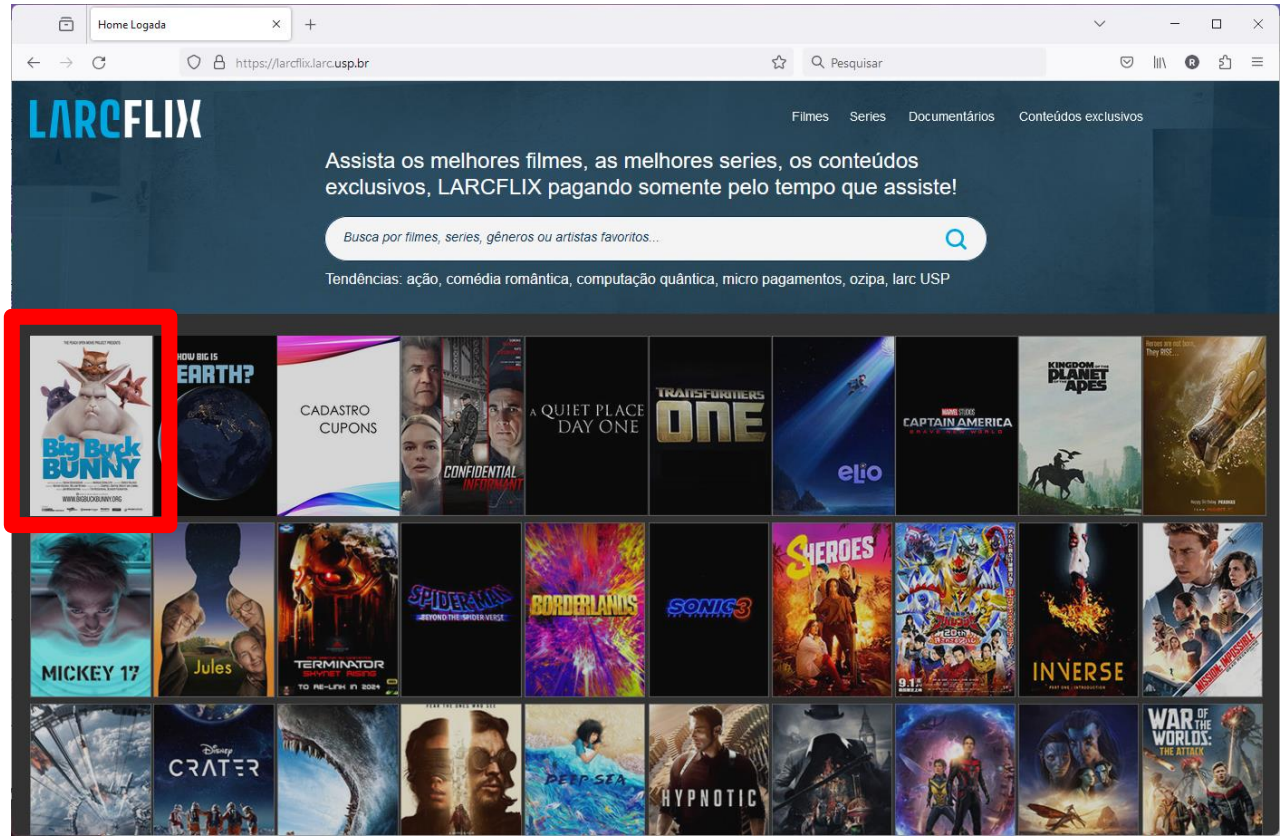


Fluxo

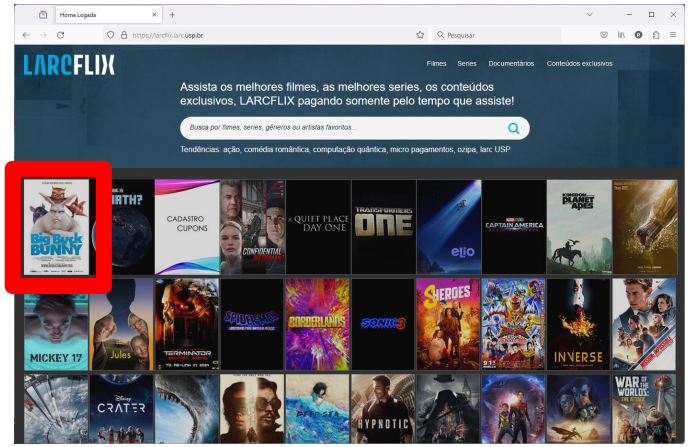


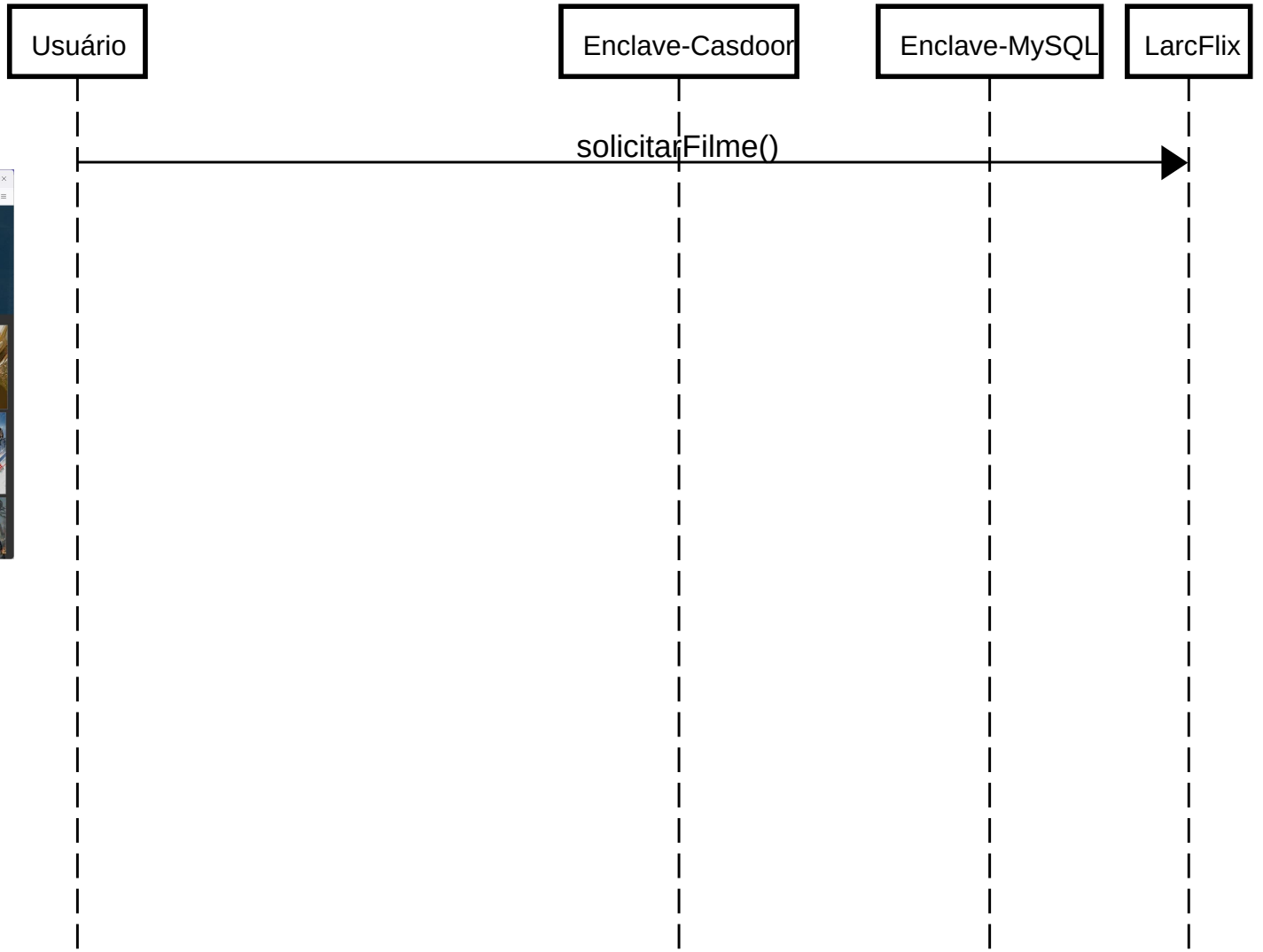
- Usuário acessa um site “client” do OAuth, no caso ao lado, o “Larcflix”

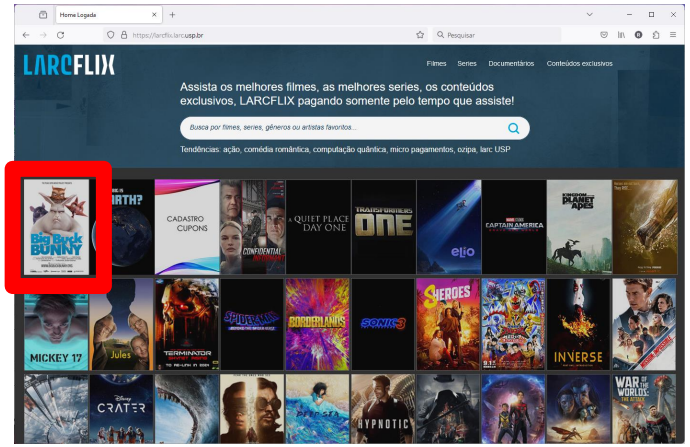
Fluxo

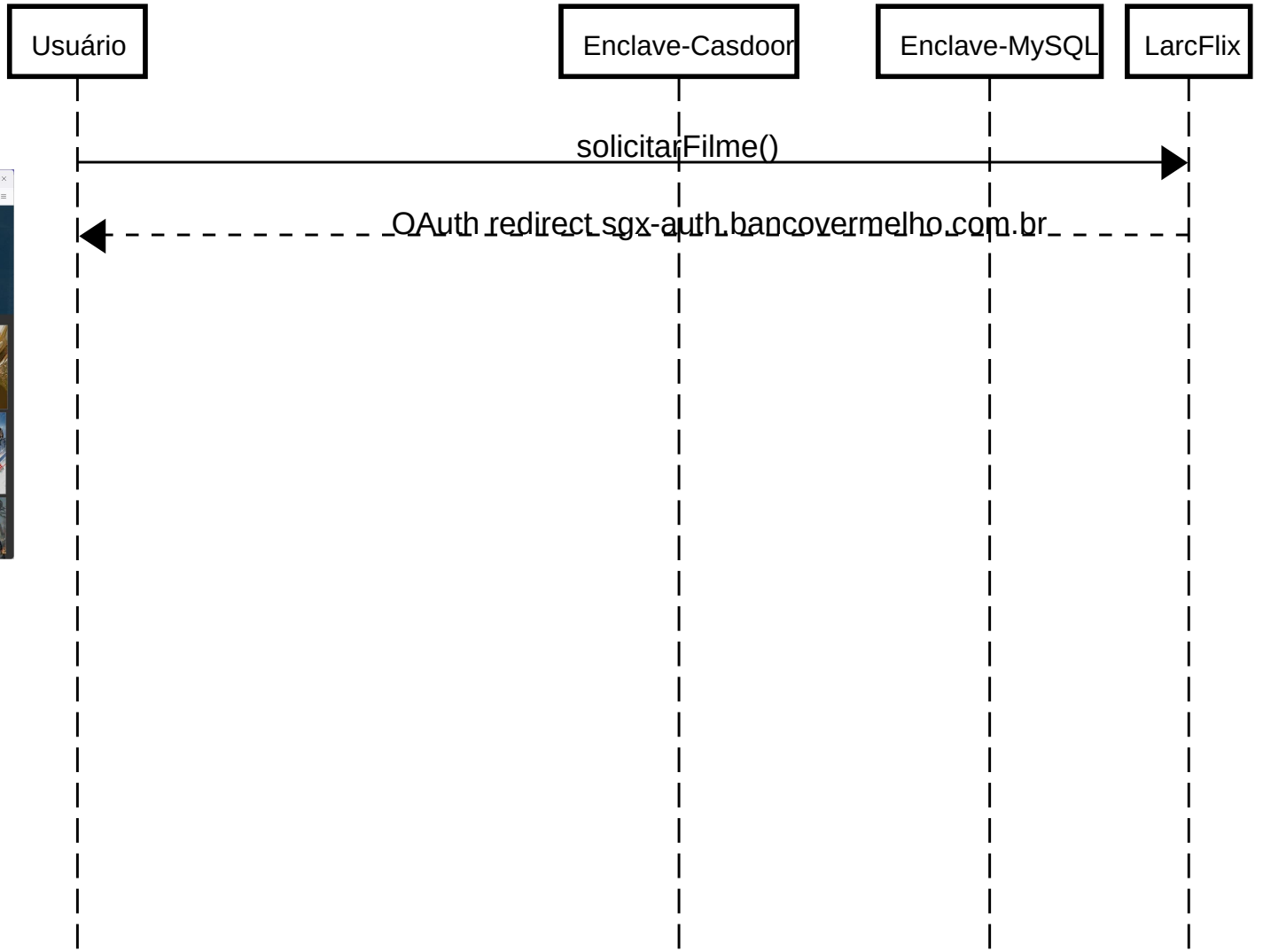


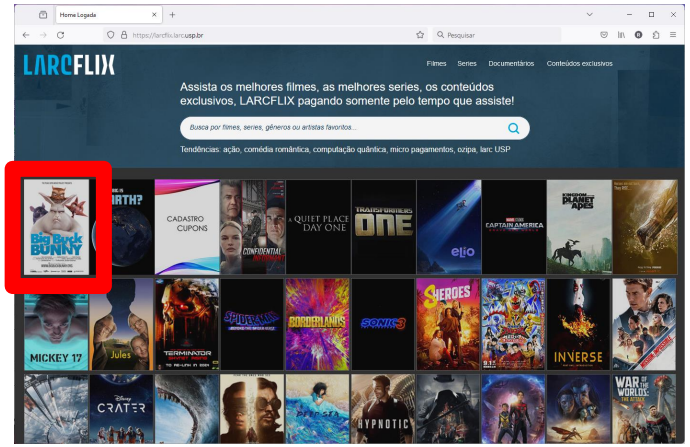
- Clica no filme que deseja assistir

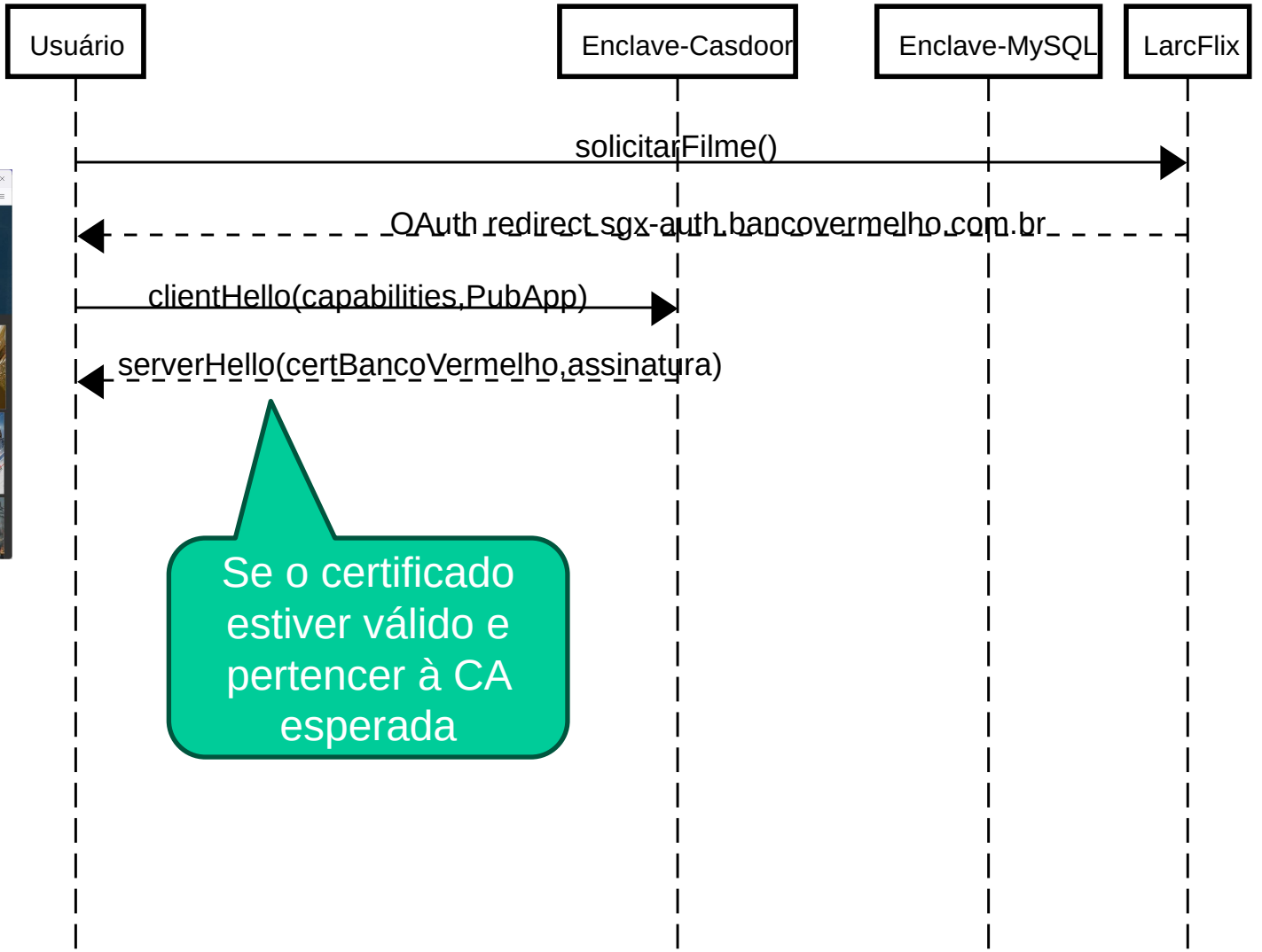

LARCFLIX
Filmes
Series
Documentários
Conteúdos exclusivos
Assista os melhores filmes, as melhores series, os conteúdos exclusivos, LARCFLIX pagando somente pelo tempo que assiste!
Busca por filmes, series, gêneros ou artistas favoritos...
Tendências: ação, comédia romântica, computação quântica, micro pagamentos, ozipa, larc USP
Big Buck Bunny
CADASTRO CUPONS
QUIET PLACE DAY ONE
ONE
elio
CAPTAIN AMERICA
PLANET OF THE APES
MICKEY 17
Jules
TERMINATOR
BORDERLANDS
SONIC 3
HEROES
INVERSE
CRATER
HYPNOTIC
WAR OF THE WORLDS

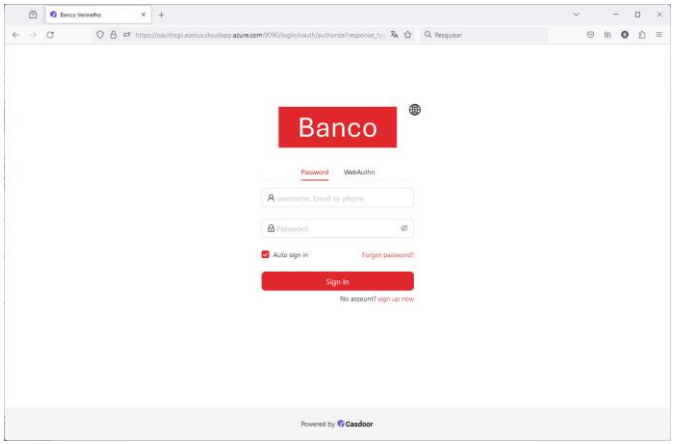

Usuário
Enclave-Casdoor
Enclave-MySQL
LarcFlix
solicitarFilme()

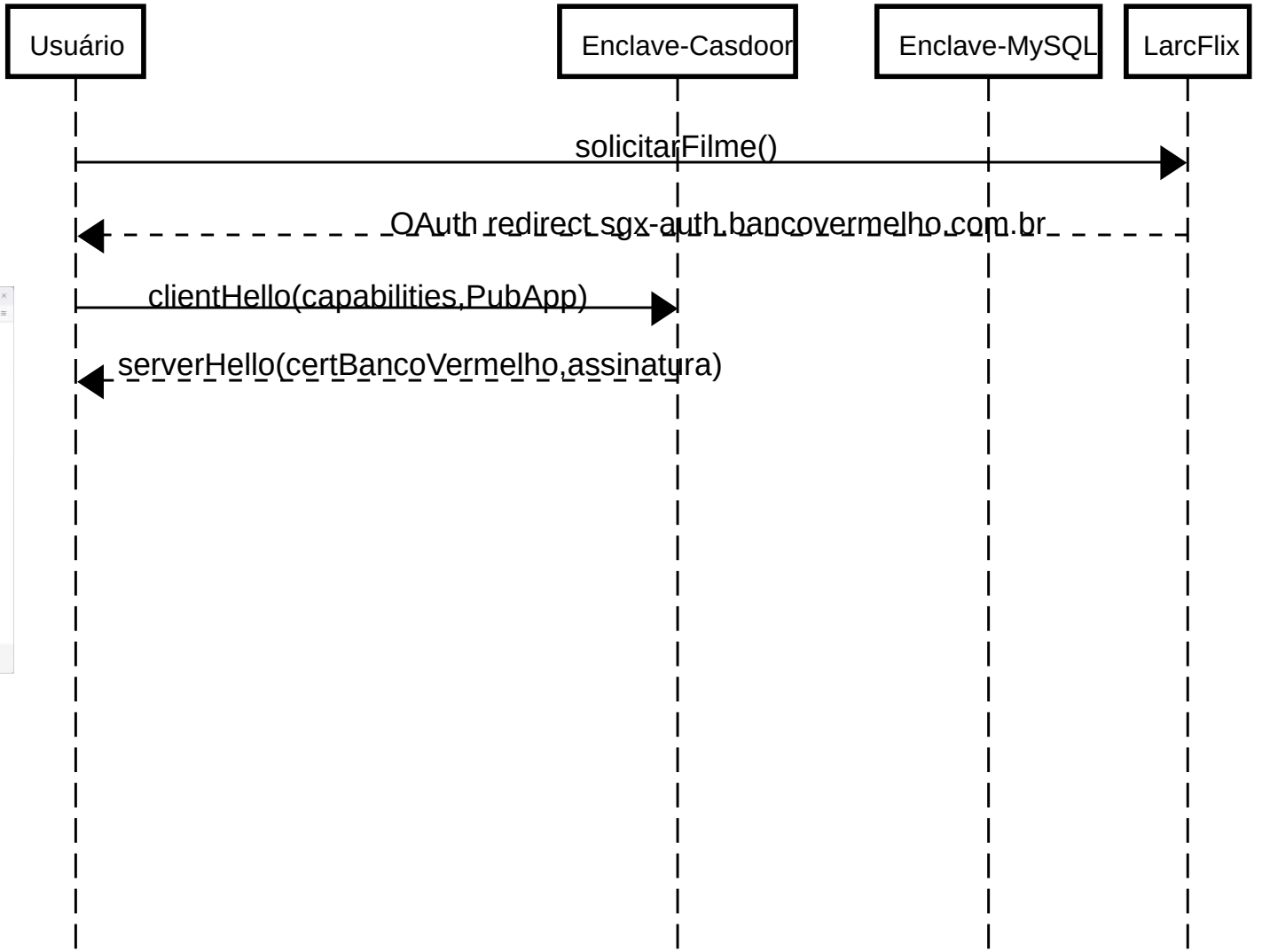

LARCFLIX
Assista os melhores filmes, as melhores series, os conteúdos exclusivos, LARCFLIX pagando somente pelo tempo que assiste!
Busca por filmes, series, gêneros ou artistas favoritos...
Tendências: ação, comédia romântica, computação quântica, micro pagamentos, ozipa, larc USP
Filmes
Series
Documentários
Conteúdos exclusivos
Big Buck Bunny
CADASTRO CUPONS
QUIET PLACE DAY ONE
ONE
elio
MICKEY 17
Jules
TERMINATOR
BORDERLANDS
SONIC 3
HEROES
INVERSE
CRATER
HYPNOTIC
WAR OF THE WORLDS

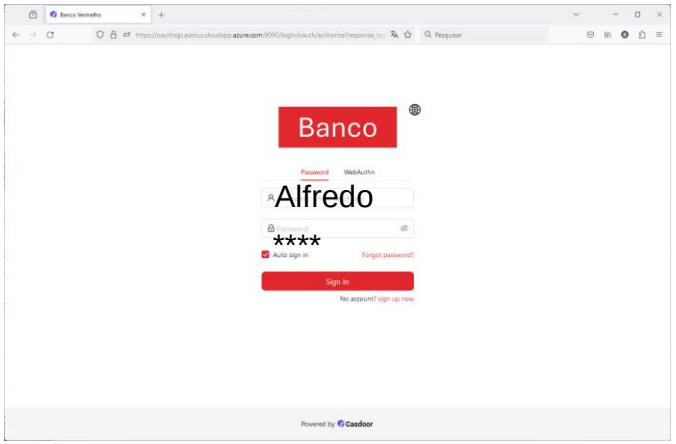

Usuário
Enclave-Casdoor
Enclave-MySQL
LarcFlix
solicitarFilme()
OAuth redirect sgx-auth.bancovermelho.com.br

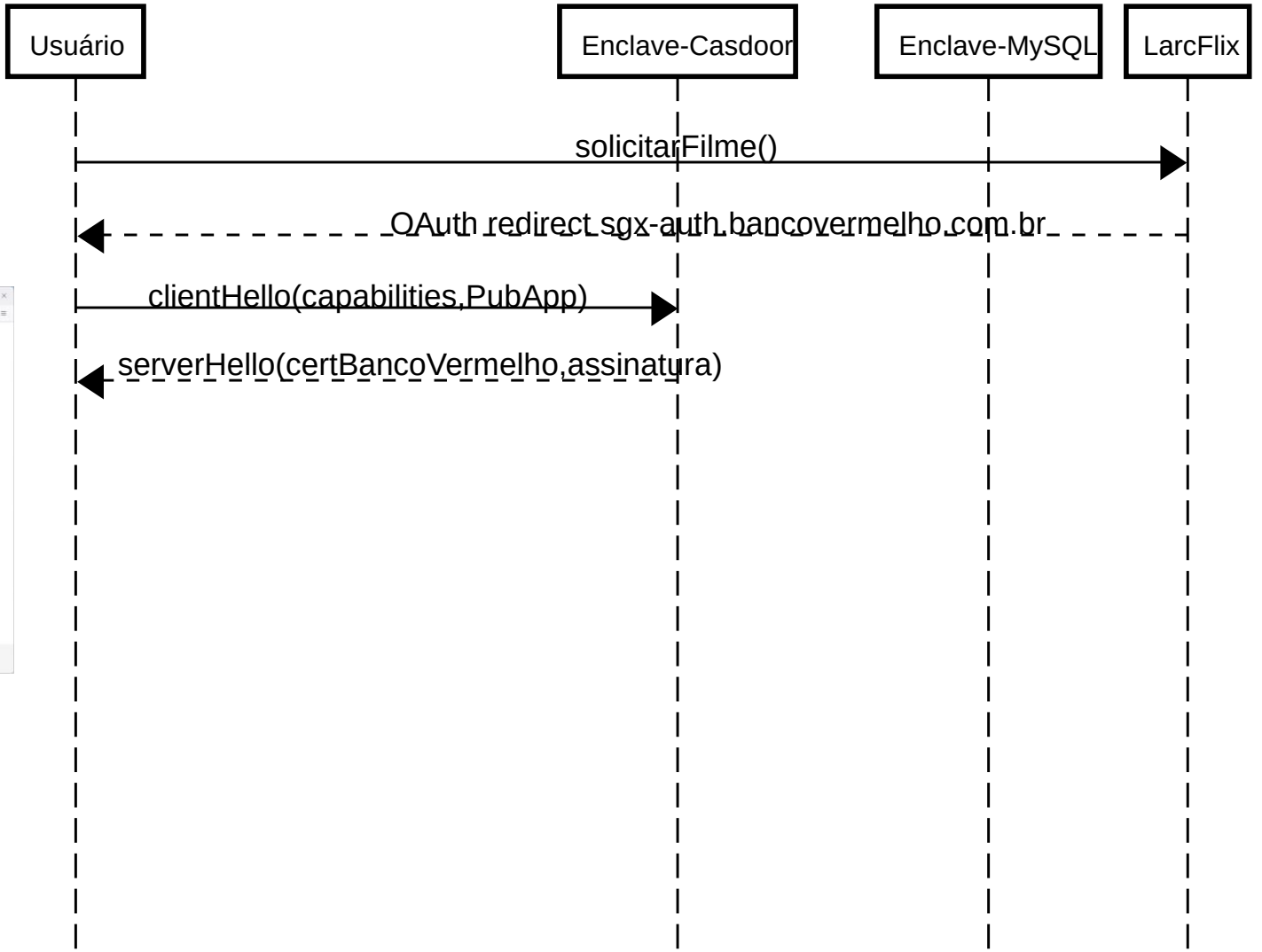

LARCFLIX
Assista os melhores filmes, as melhores series, os conteúdos exclusivos, LARCFLIX pagando somente pelo tempo que assiste!

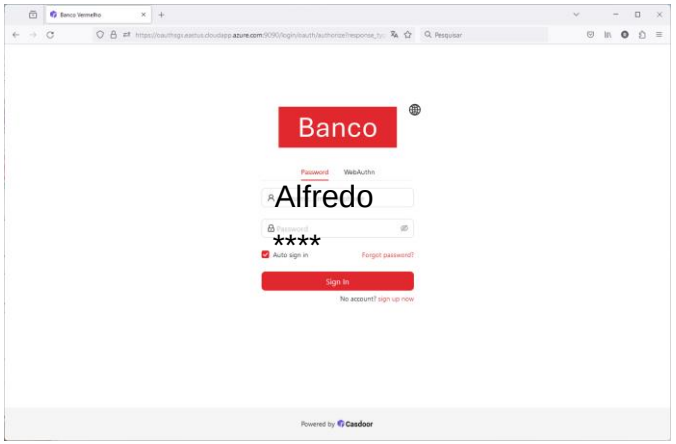

Usuário
Enclave-Casdoor
Enclave-MySQL
LarcFlix
solicitarFilme()
OAuth redirect sgx-auth.bancovermelho.com.br
clientHello(capabilities,PubApp)
serverHello(certBancoVermelho,assinatura)
Se o certificado estiver válido e pertencer à CA esperada

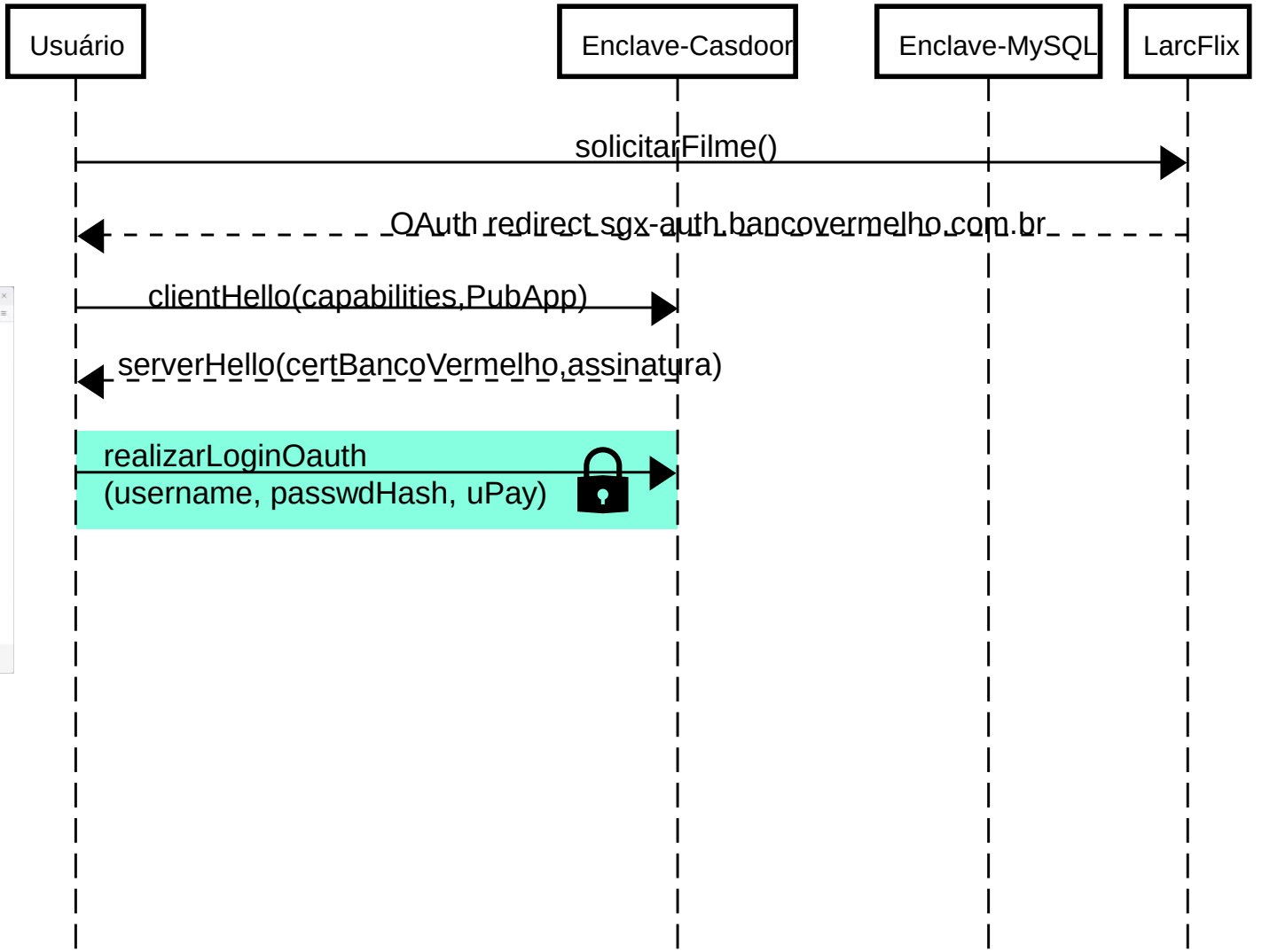

Banco
Password
WebAuthn
Auto sign in
Forgot password?
Sign In
No account? sign up now
Powered by Casdoor

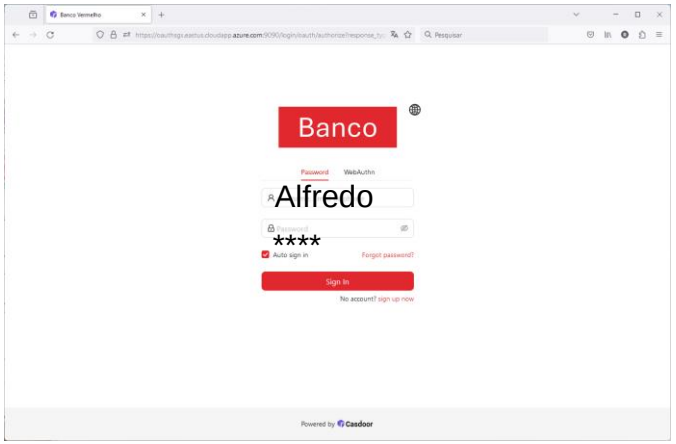

Usuário
Enclave-Casdoor
Enclave-MySQL
LarcFlix
solicitarFilme()
OAuth redirect sgx-auth.bancovermelho.com.br
clientHello(capabilities,PubApp)
serverHello(certBancoVermelho,assinatura)

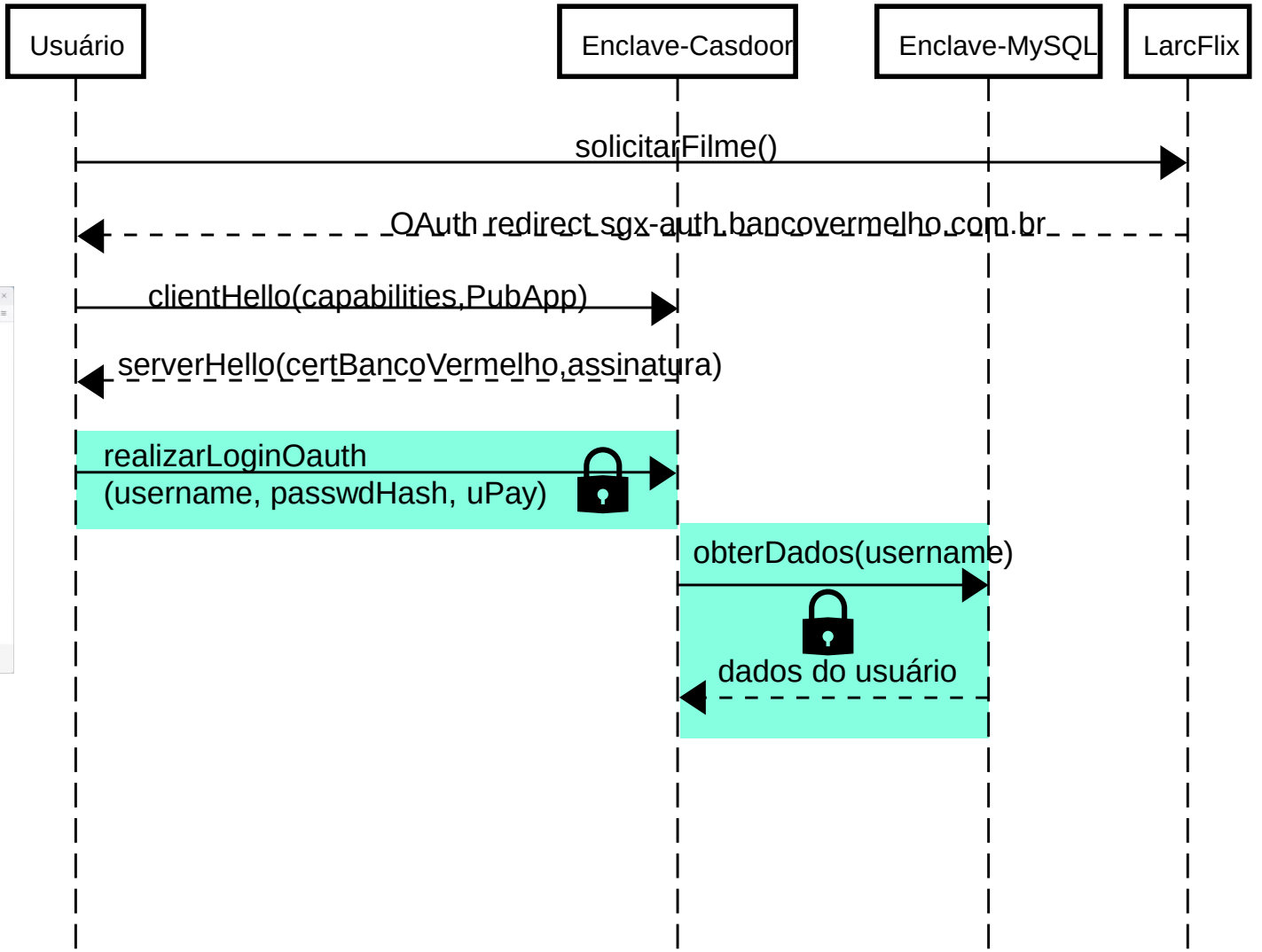

Banco
Alfredo
****
Sign In

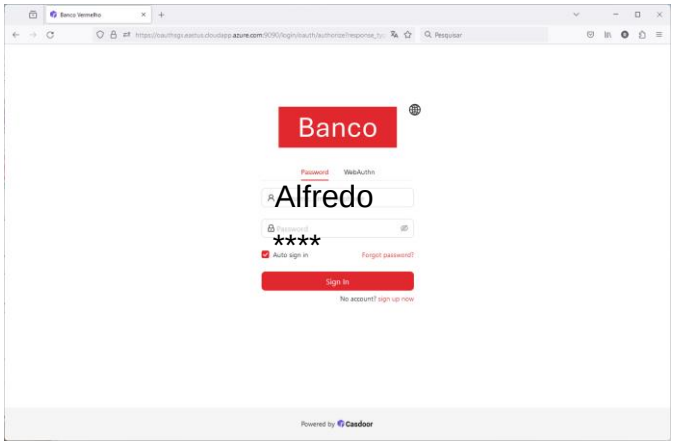

Usuário
Enclave-Casdoor
Enclave-MySQL
LarcFlix
solicitarFilme()
OAuth redirect sgx-auth.bancovermelho.com.br
clientHello(capabilities,PubApp)
serverHello(certBancoVermelho,assinatura)

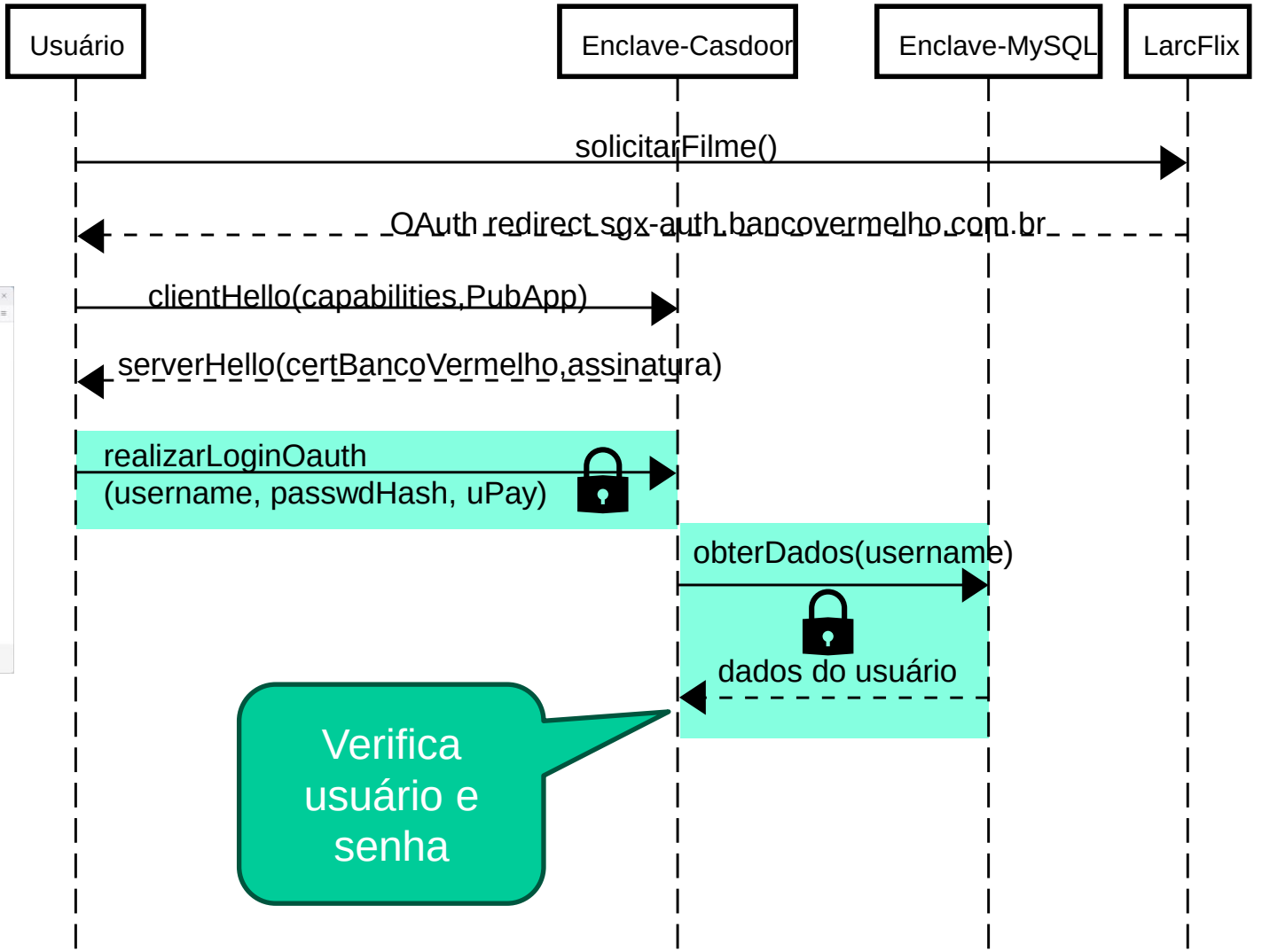

Banco
Alfredo
****
Sign In

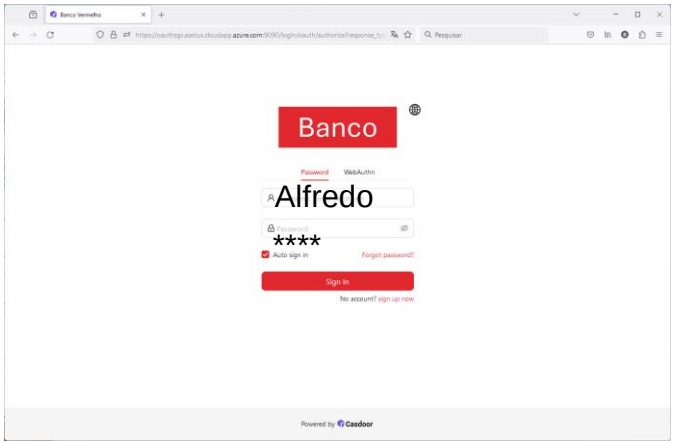

Usuário
Enclave-Casdoor
Enclave-MySQL
LarcFlix
solicitarFilme()
OAuth redirect sgx-auth.bancovermelho.com.br
clientHello(capabilities,PubApp)
serverHello(certBancoVermelho,assinatura)
realizarLoginOauth
(username, passwdHash, uPay)

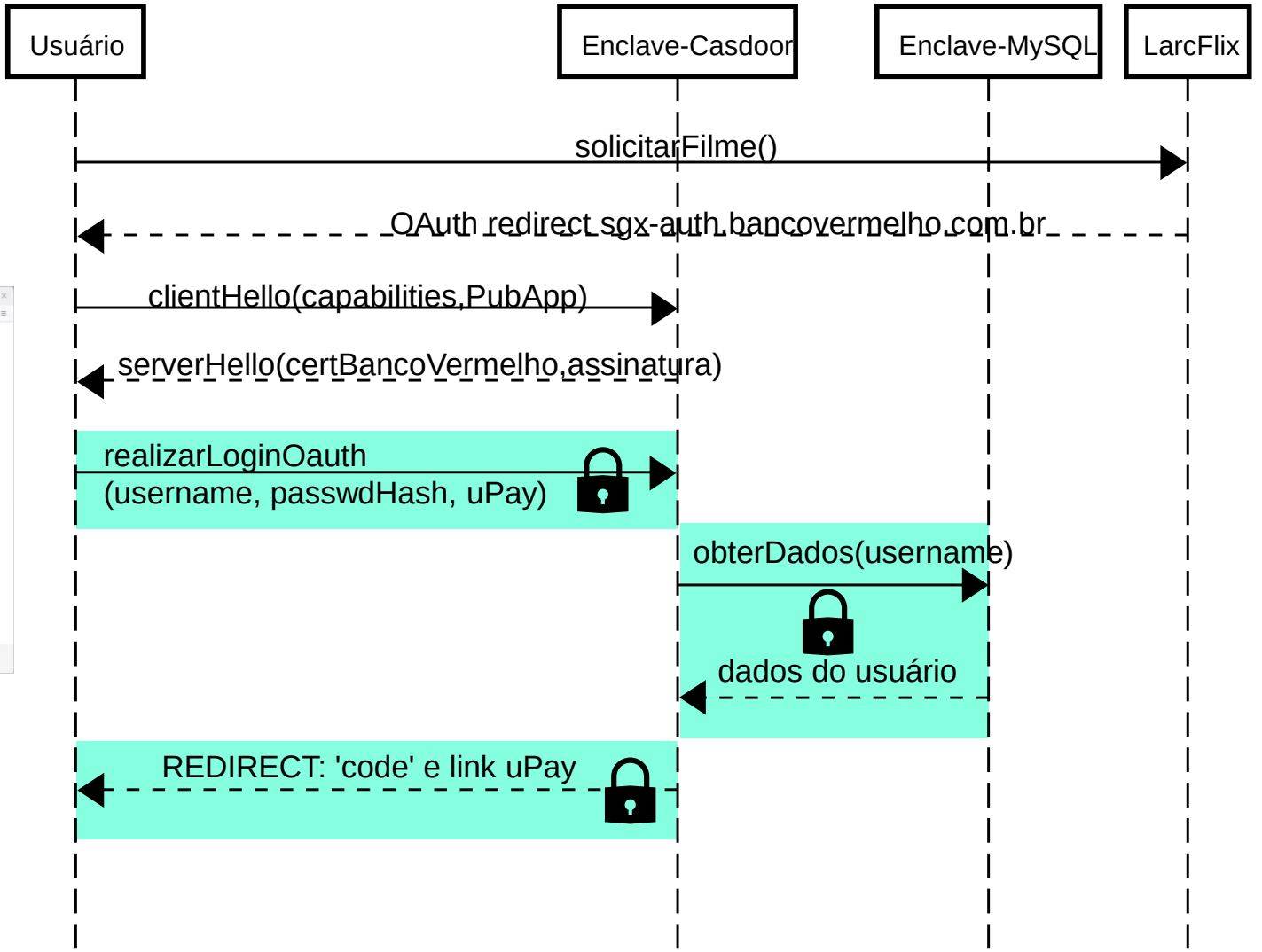

Banco
Alfredo
****
Sign In

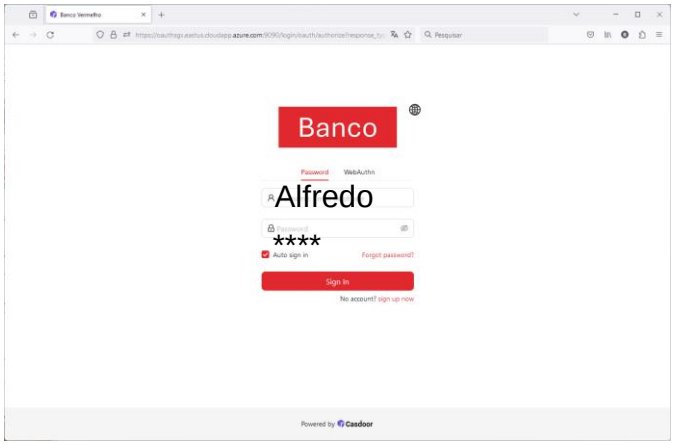

Usuário
Enclave-Casdoor
Enclave-MySQL
LarcFlix
solicitarFilme()
OAuth redirect sgx-auth.bancovermelho.com.br
clientHello(capabilities,PubApp)
serverHello(certBancoVermelho,assinatura)
realizarLoginOauth
(username, passwdHash, uPay)
obterDados(username)
dados do usuário

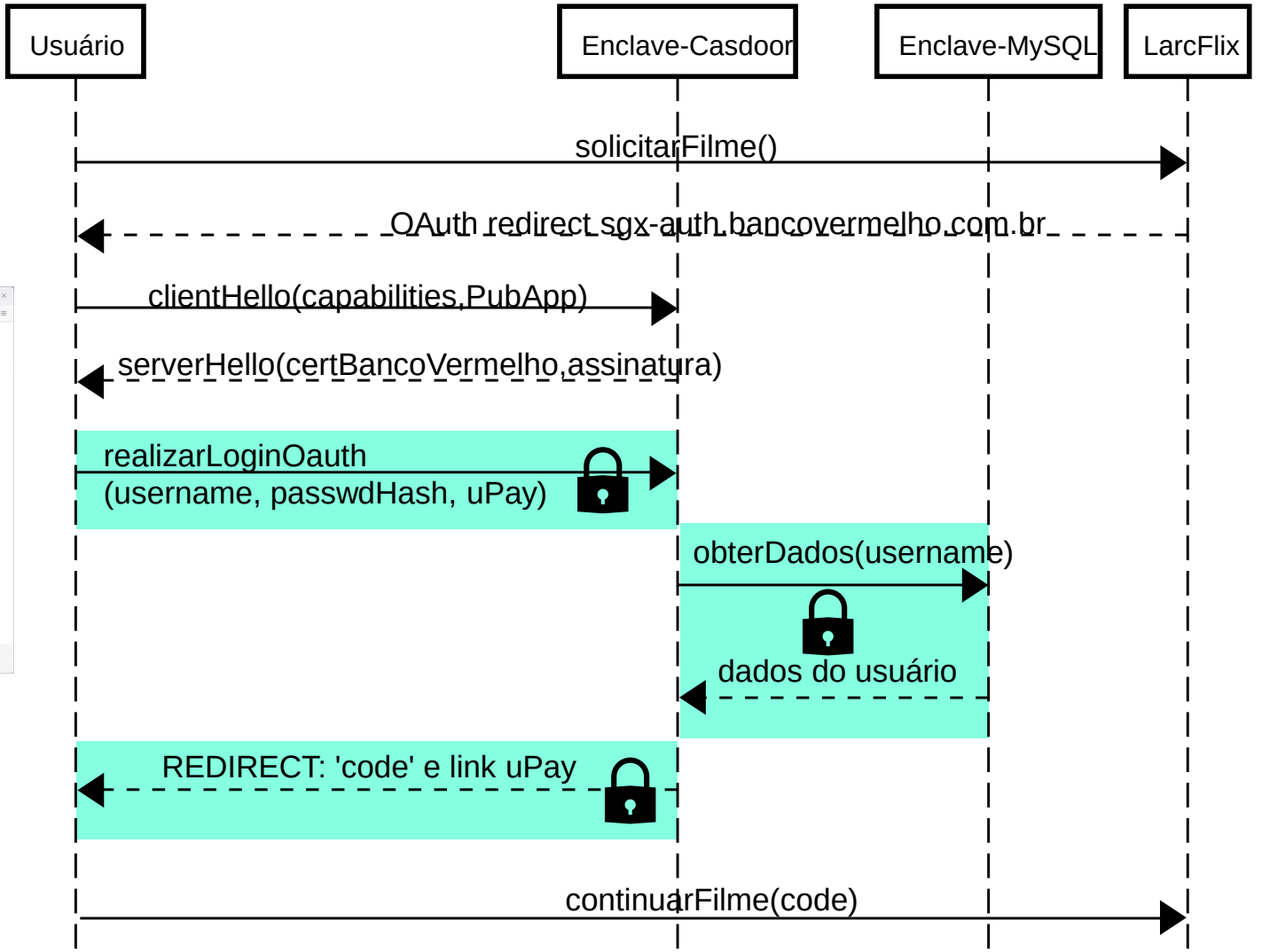

Banco
Alfredo
****
Sign In

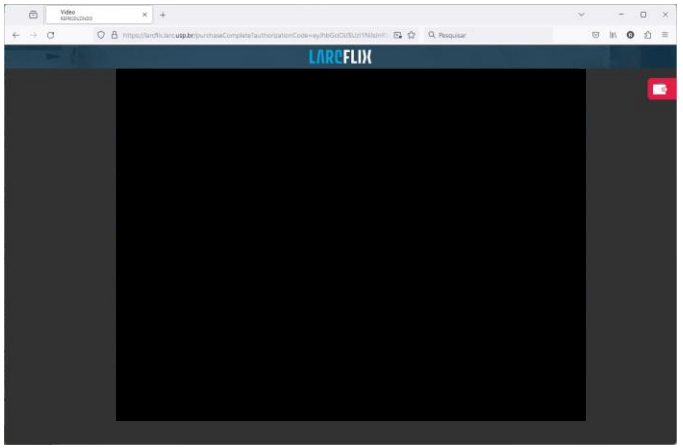

Usuário
Enclave-Casdoor
Enclave-MySQL
LarcFlix
solicitarFilme()
OAuth redirect sgx-auth.bancovermelho.com.br
clientHello(capabilities,PubApp)
serverHello(certBancoVermelho,assinatura)
realizarLoginOauth
(username, passwdHash, uPay)
obterDados(username)
dados do usuário
Verifica usuário e senha

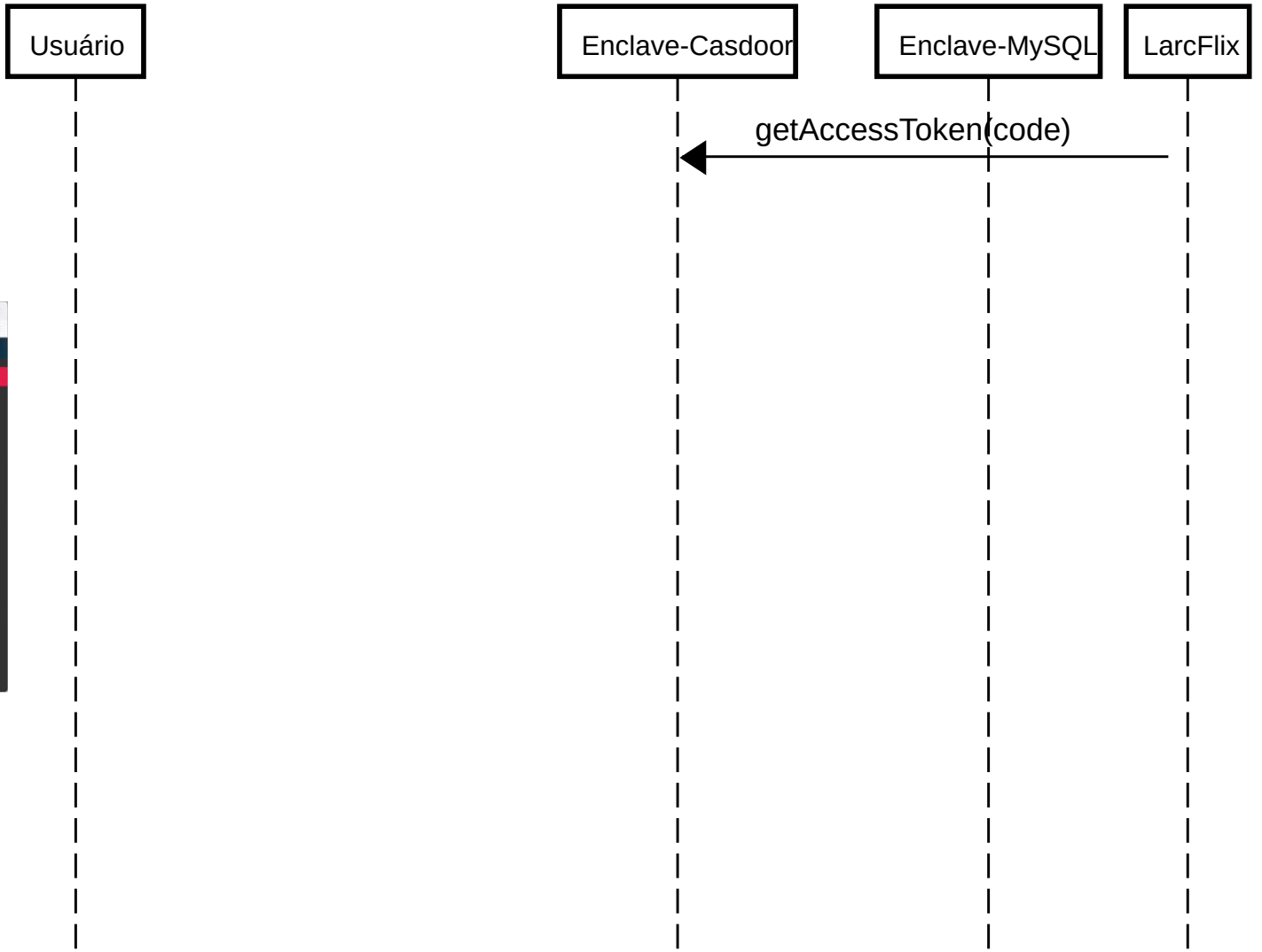

Banco
Alfredo
****
Sign In

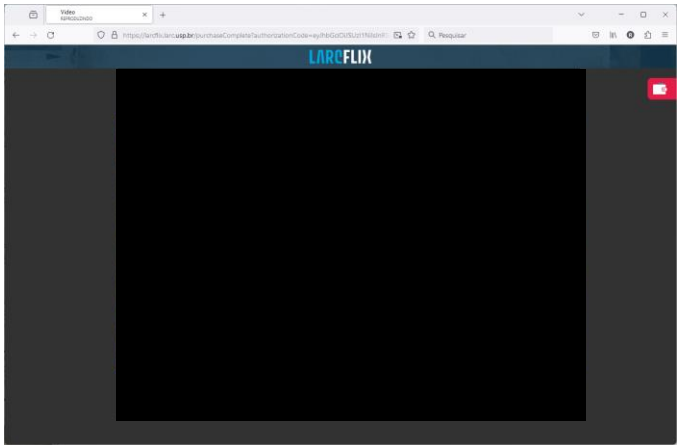

Usuário
Enclave-Casdoor
Enclave-MySQL
LarcFlix
solicitarFilme()
OAuth redirect sgx-auth.bancovermelho.com.br
clientHello(capabilities,PubApp)
serverHello(certBancoVermelho,assinatura)
realizarLoginOauth
(username, passwdHash, uPay)
obterDados(username)
dados do usuário
REDIRECT: 'code' e link uPay

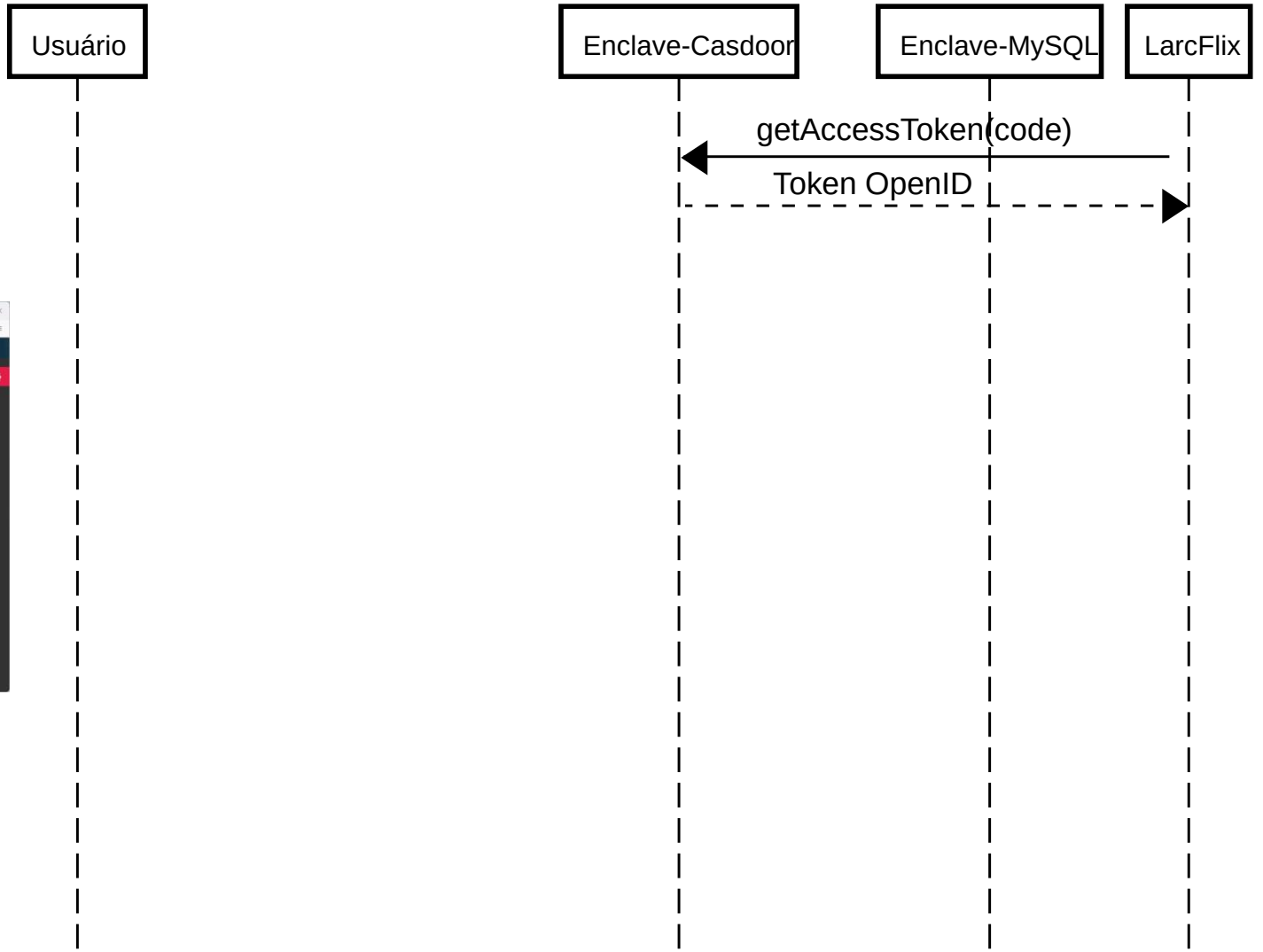

Banco
Alfredo
****
Sign In

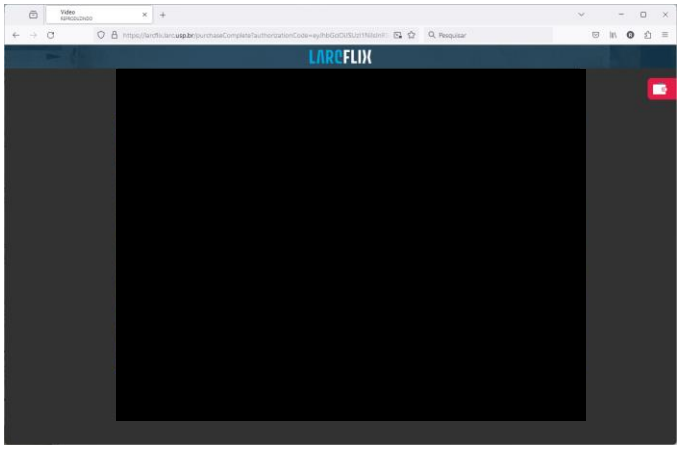

Usuário
Enclave-Casdoor
Enclave-MySQL
LarcFlix
solicitarFilme()
OAuth redirect sgx-auth.bancovermelho.com.br
clientHello(capabilities,PubApp)
serverHello(certBancoVermelho,assinatura)
realizarLoginOauth
(username, passwdHash, uPay)
obterDados(username)
dados do usuário
REDIRECT: 'code' e link uPay
continuarFilme(code)

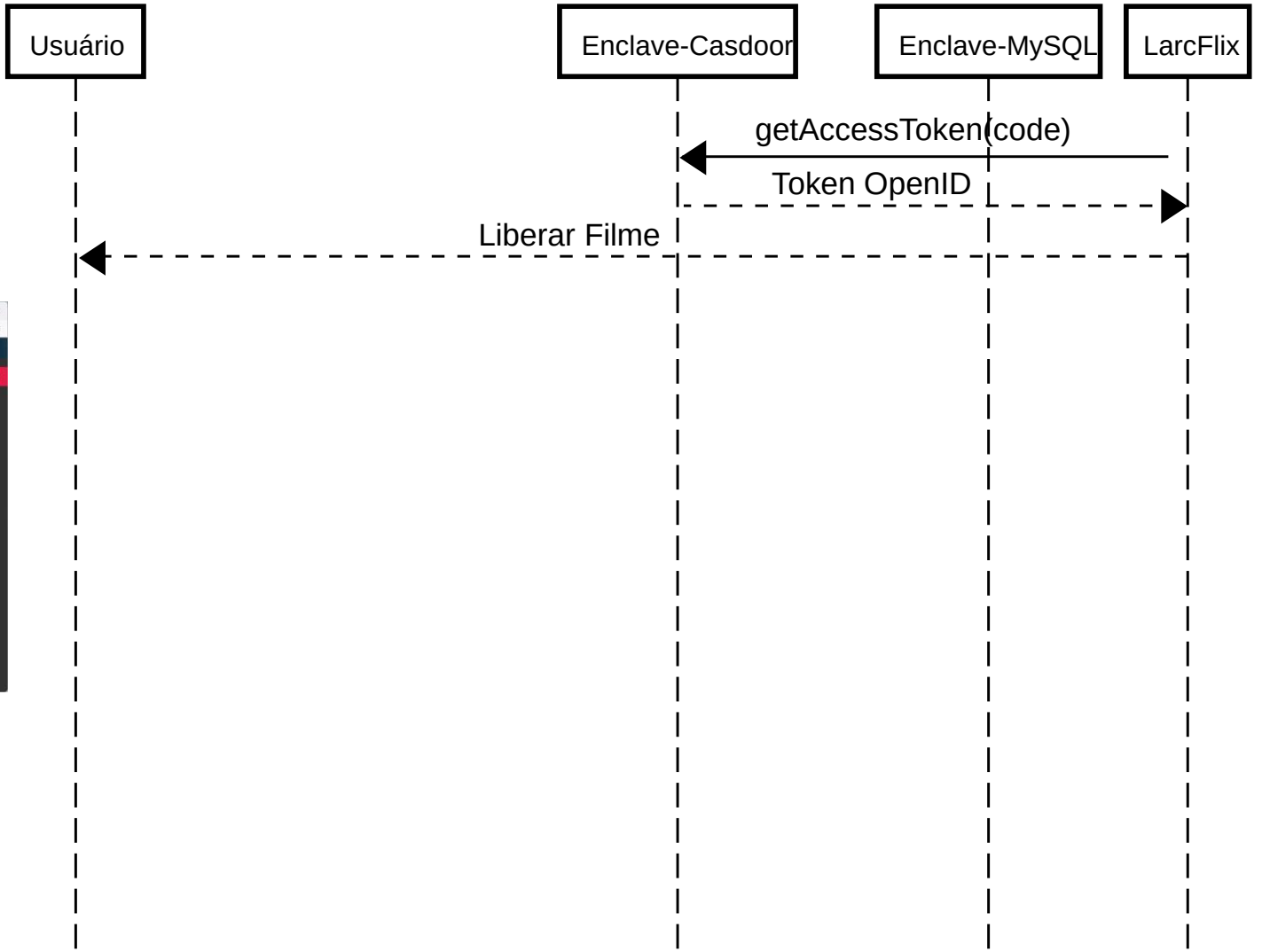

LARCFLIX

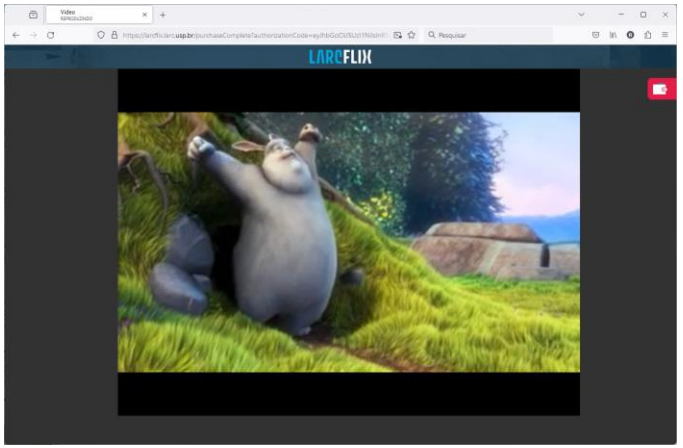

Usuário
Enclave-Casdoor
Enclave-MySQL
LarcFlix
getAccessToken(code)

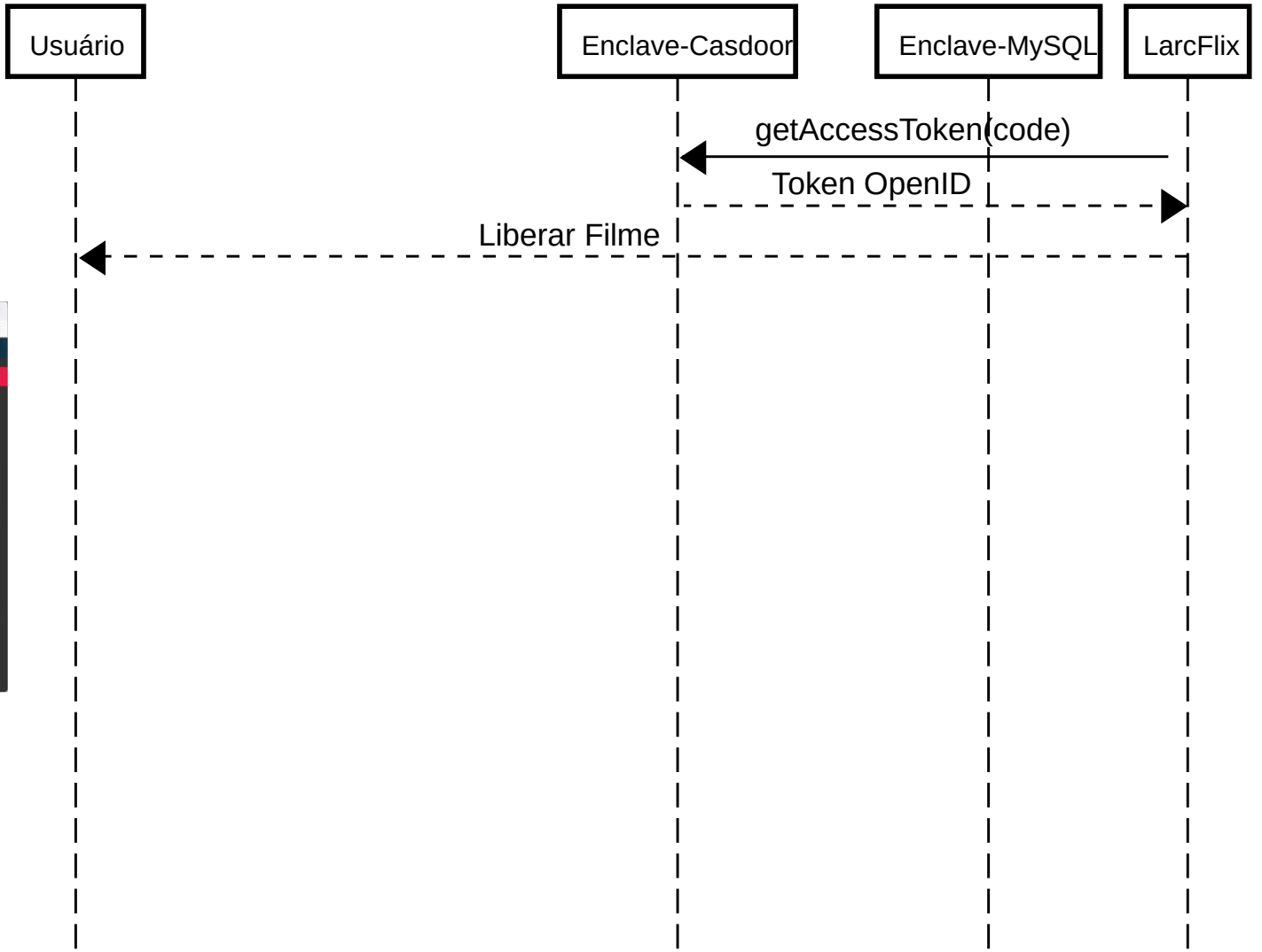

LARCFLIX


Usuário
Enclave-Casdoor
Enclave-MySQL
LarcFlix
getAccessToken(code)
Token OpenID


LARCFLIX


Usuário
Enclave-Casdoor
Enclave-MySQL
LarcFlix
getAccessToken(code)
Token OpenID
Liberar Filme


LARCFLIX


Usuário
Enclave-Casdoor
Enclave-MySQL
LarcFlix
getAccessToken(code)
Token OpenID
Liberar Filme

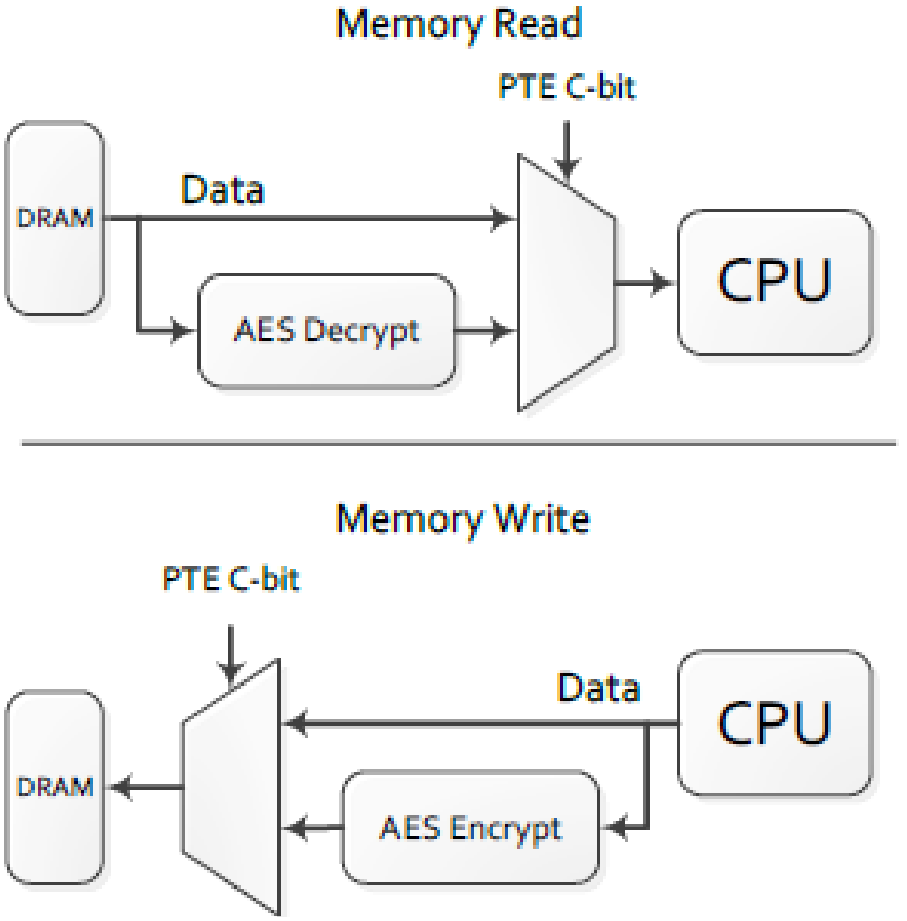
AMD SEV/SNP

- **Objetivo:**
  - Desenvolvimento de novas tecnologias de encriptação de memória para fins de TEE.
- **SME - Secure Memory Encryption**
  - Memória é cifrada pelo processador utilizando uma chave simétrica gerada no momento do boot. Precisa ser habilitada na BIOS
- **SEV - Secure Encrypted Virtualization**
  - Mecanismo de execução de VM seguras. Elas ficam isoladas entre si e do hypervisor. Precisa ser habilitada no hypervisor e na VM guest
  - Aplicável em um cenário de nuvem: muitas VMs em execução no mesmo HW

AMD SEV/SNP

SME – Secure Memory Encryption

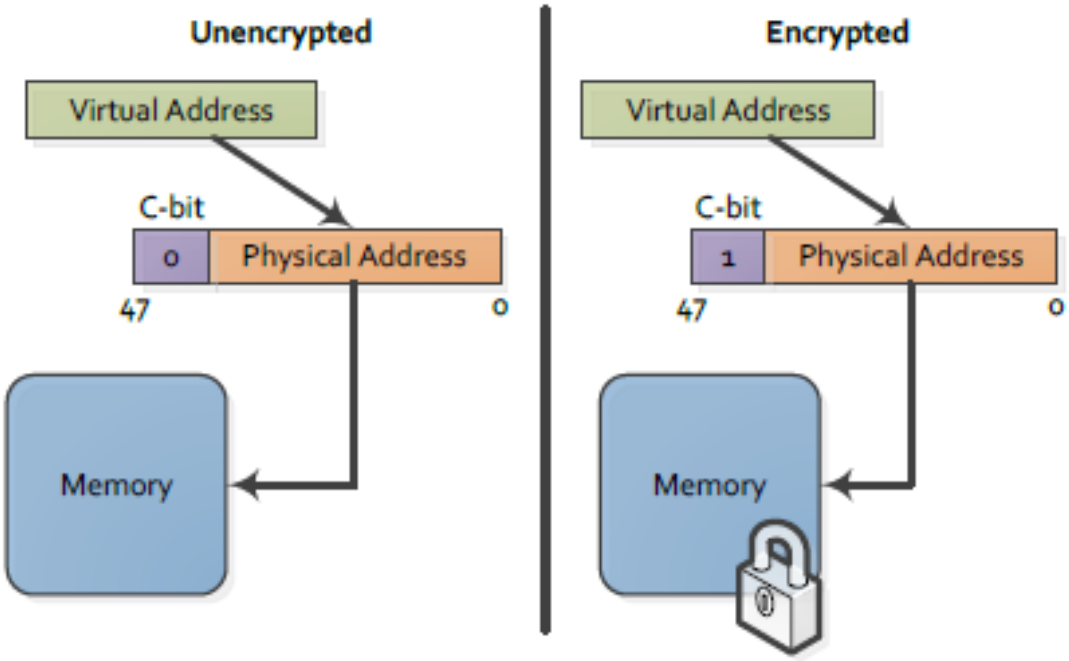
- **Memória RAM cifrada**
  - Mecanismo de cifrar / decifrar funciona no controlador de memória no chip
  - Algoritmo AES com chave de 128 bits
  - AMD Secure Processor (AMD-SP):
    - Microcontrolador ARM® Cortex®-A5 (32-bit) integrado no processador
    - Responsável por gerenciar as chaves



AMD SEV/SNP

SME – Secure Memory Encryption

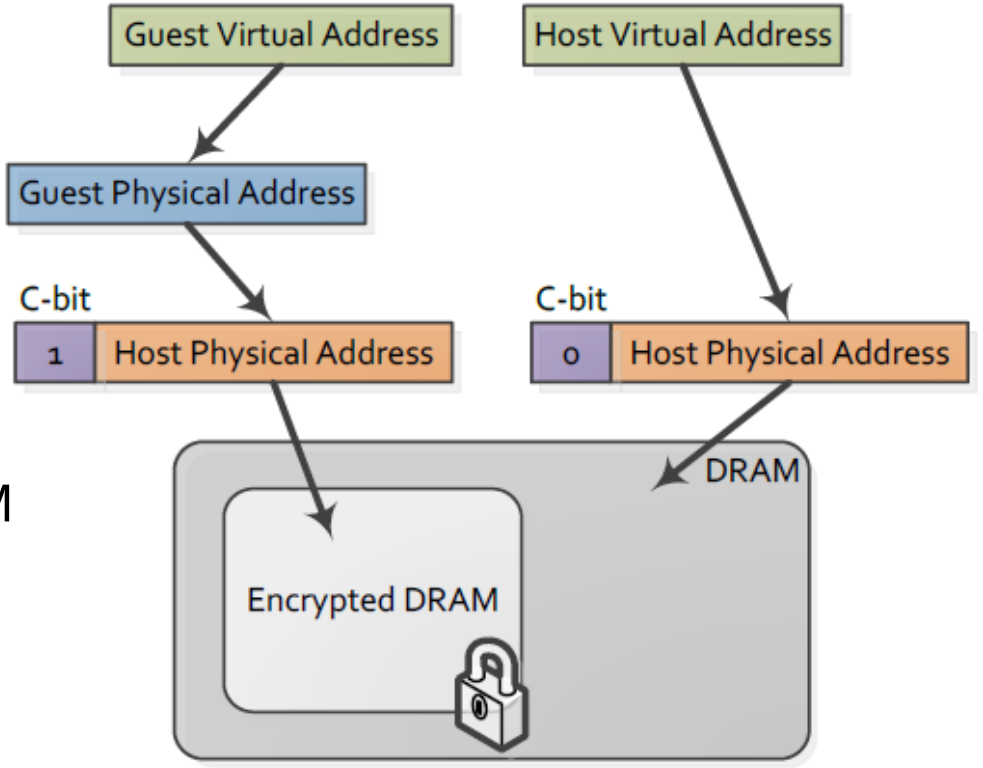
- **Memória RAM cifrada**
  - A escolha/seleção de quais páginas de memórias são cifradas é feita pelo SO ou pelo hypervisor
    - Bit 47 de uma página de memória indica se ela é cifrada (C bit “enCrypted”)
  - No processo de cifração/decifração há uma pequena latência dependendo do tipo de processamento



AMD SEV/SNP

SME – Secure Memory Encryption

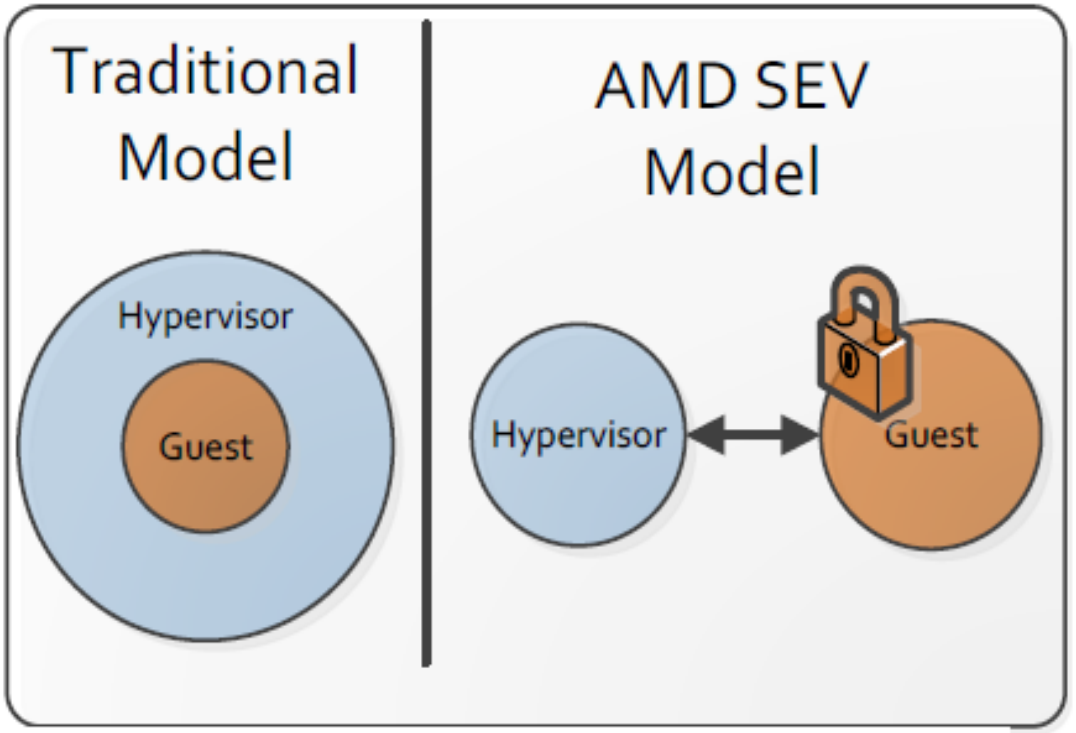
- **Modos de cifração**:
  - **Total**: criptografia total da memória.
    - Configura o bit 47 em todas páginas físicas de memória.
    - Previne contra ataques a NVDIMM (non-volatile DIMM);
  - **Parcial**: somente área utilizada por uma VM guest – Figura.
- **Obs.**: com a criptografia acionada, não é possível escanear a memória, mesmo com ferramentas especiais.
  - **Transparent SME**: ativado na BIOS.
  - Não precisa de suporte do SO ou do hypervisor.



AMD SEV/SNP

SEV - Secure Encrypted Virtualization

- **Principal função de SEV**
  - Isolar uma VM guest, até mesmo de execuções privilegiadas como o próprio hypervisor.

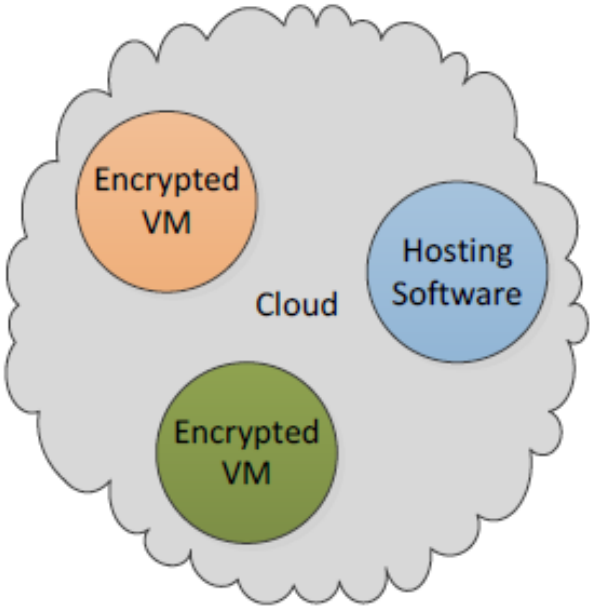


AMD SEV/SNP

SEV - Secure Encrypted Virtualization

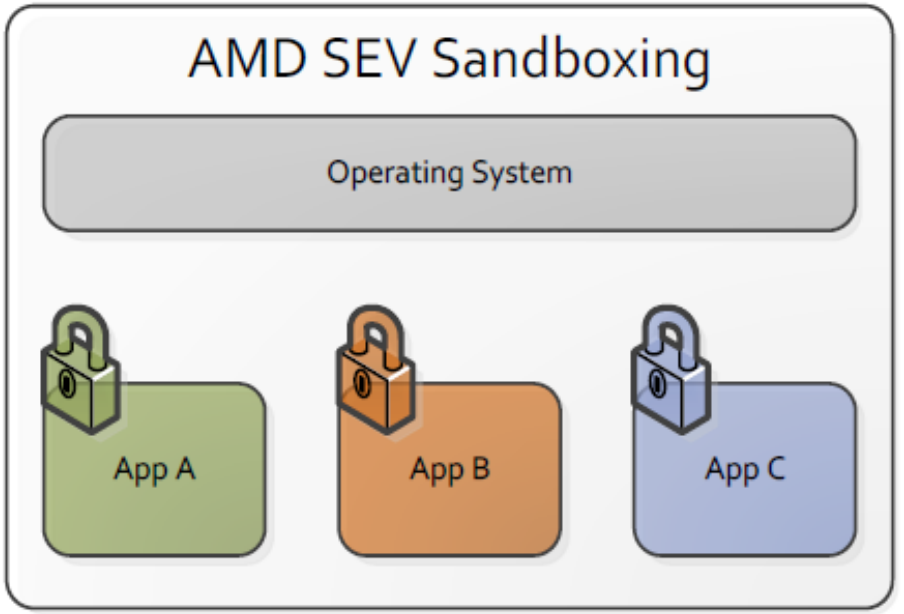
- **Casos de Uso**

Cloud



Sandboxing

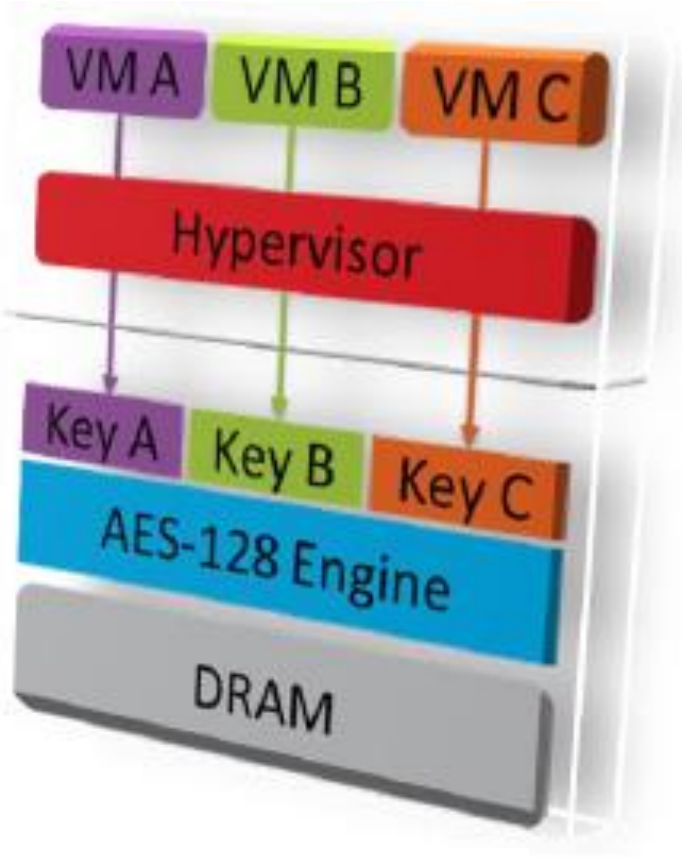
Ex.: Container Docker



AMD SEV/SNP

SEV - Secure Encrypted Virtualization

- **Arquitetura**
  - SEV é uma extensão do AMD-V (análogo ao intel VT-x)
  - Quando acionado, o SEV cria uma marcação (TAG) em todo código e dados com o ASID (*Address Space ID*) da VM.
  - Essa TAG é armazenada dentro do SoC.
  - Quando a memória RAM é utilizada a partir da VM, é cifrada e decifrada automaticamente dentro do SoC, usando uma chave associada com a TAG.
  - Até mesmo o software de hypervisor também é marcado com uma TAG

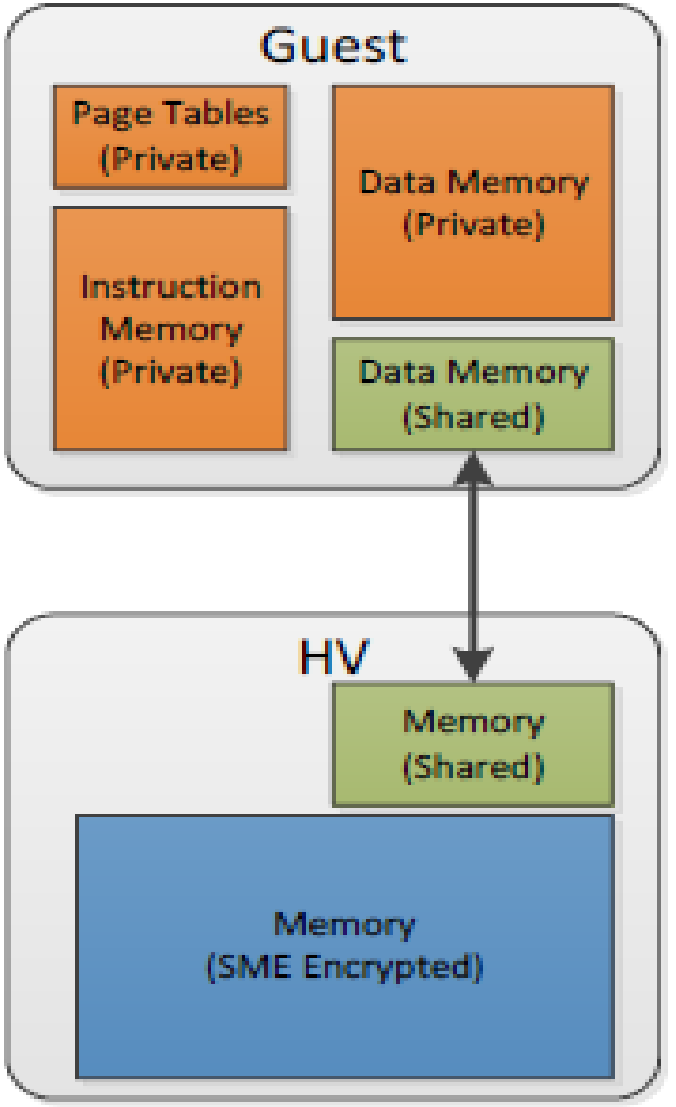


Arquitetura

AMD SEV/SNP

SEV - Secure Encrypted Virtualization

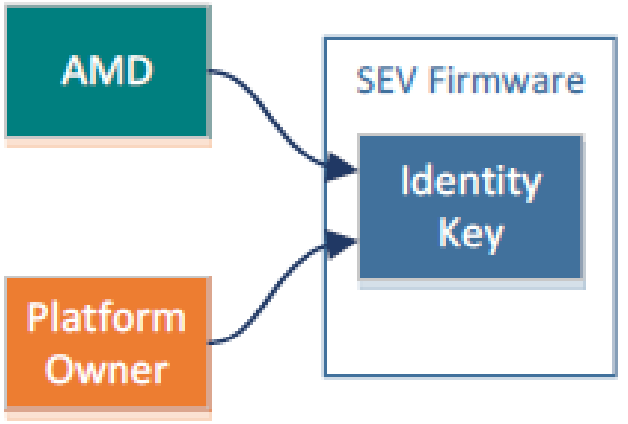
- Nem toda memória de uma VM precisa ser cifrada → a VM pode selecionar quais páginas podem ser ou não cifradas.
- Além disso, a memória compartilhada entre o hypervisor e a VM pode ser cifrada com a chave do hypervisor ou, simplesmente, não ser cifrada.
- No exemplo ao lado, toda memória da VM é cifrada e a memória compartilhada não é cifrada.



AMD SEV/SNP

SEV - Secure Encrypted Virtualization

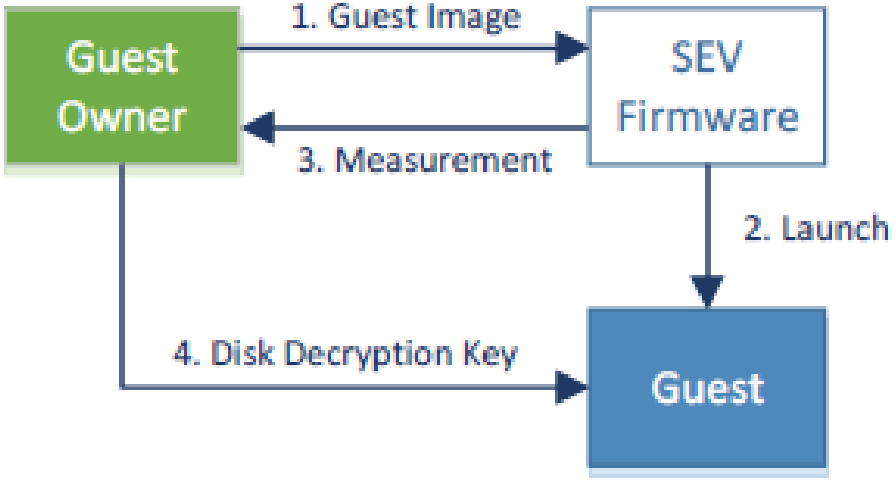
- **Gestão de Chaves**
  - O firmware executado no AMD-SP realiza a gestão de chaves para garantir a segurança. Embora o hypervisor faça a gestão dos recursos das VMs, as chaves sempre ficam em posse do AMD-SP
  - O firmware auxilia as VMs na imposição das propriedades de segurança:
    - Autenticidade da plataforma: verifica se é genuinamente AMD
    - Confidencialidade dos dados (página ou VM *guest*)
    - Atestação da VM *guest*: vários componentes do estado da VM, inclusive do conteúdo inicial da memória (próximo slide)



AMD SEV/SNP

SEV - Secure Encrypted Virtualization

- **Atestação**
  1. Guest owner envia/transmite a imagem da VM a ser executada.
  2. SEV inicia a VM e tira a medida (*measurement*) da VM
  3. SEV envia a *measurement* ao Guest owner
  4. Se o Guest Owner estiver de acordo, isto é, a *measurement* está correta, pode enviar dados adicionas a VM como uma chave de decifração de disco, por exemplo



Obs.: Há uma situação que a VM pode ser “replicada” remotamente: a VM é transmitida usando a chave que, por sua vez, também é enviada a outra máquina com SEV.

Para que isso ocorra, há uma autenticação da plataforma.

AMD SEV/SNP

SEV - Secure Encrypted Virtualization

- **Observações Gerais**
  - **Hypervisor:**
    - Requer comunicação com o AMD-SP (microcontrolador) para solicitações de chaves (mesmo que não tenha acesso)
    - Para tal, precisa ser compatível com SEV e, também, ter o driver para comunicação
    - A comunicação visa: carregar e/ou gerar a chave correta para a VM guest; estabelecer um mecanismo seguro para atestação, ...
    - Exemplos de *hypervisors* compatíveis: KVM/QEMU e VMware vSphere
    - Ambiente seguro, mas com observações acerca dos dispositivos virtuais, visto que podem ser um ponto de falha

AMD SEV/SNP

SEV - Secure Encrypted Virtualization

- **Observações Gerais**
  - **VM guest:**
    - Pode usar memória cifrada somente em algumas páginas
    - Se precisar usar DMA para comunicação com HW, deve usar memória não cifrada ou, alternativamente utilizar *bounce buffers*
    - Pode utilizar quantos cores (lógicos ou reais) quiser sem penalidade de eficiência, pois o *hypervisor* indicará qual chave utilizar (ASID)

AMD SEV/SNP

Atualizações na Tecnologia

- **Histórico**
  - SEV - 2016
  - SEV-ES (*Encrypted State*) – 2017
    - Os registradores de estado atuais de uma VM *guest* são cifrados usando uma chave própria para ela. A plataforma proporciona também a integridade desses estados
      - Somente a VM *guest* pode modificar esses registradores
    - A VM guest deve explicitamente compartilhar esses registradores com o *hypevisor*. Ou seja, o hypervisor não consegue acessar esses registradores.
      - Proteção adicional contra ataques relacionados: *exfiltration, control flow, rollback*

AMD SEV/SNP

Atualizações na Tecnologia

- **Histórico**
  - SEV-SNP (*Secure Nested Paging*) – 2020
    - Proteção adicional contra ataques do hypervisor
    - Melhorias de segurança opcionais para as VM
    - Proteção para tratamento de interrupções
    - Proteção contra novos ataques do tipo side-channel

Tecnologia – ARM CCA (Confidential Compute Architecture)

ARM CCA

Objetivo:

- Prover ambiente de computação segura e confidencial, protegendo a privacidade e confidencialidade
- Predecessor de ARM CCA: **TrustZone**

ARM CCA

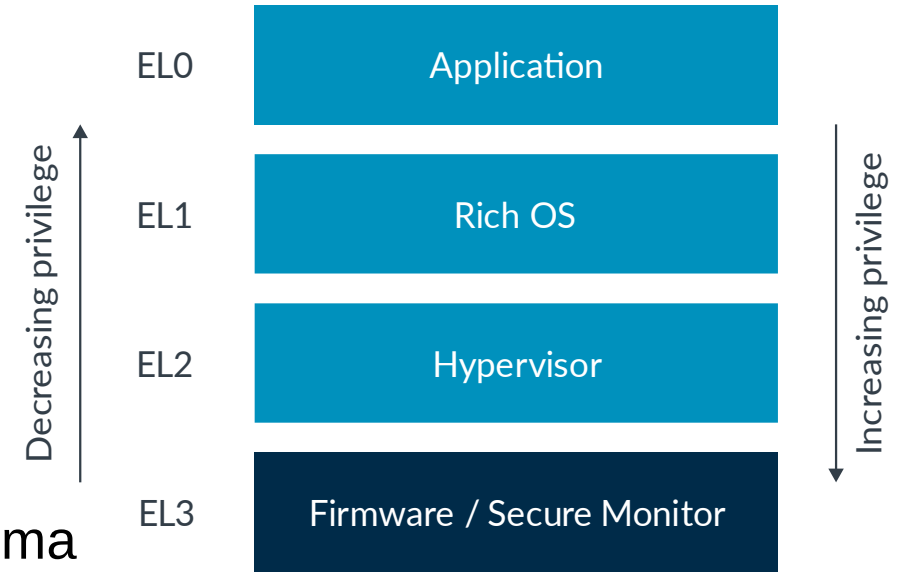
TrustZone:

- **Objetivo:** implementar uma separação por hardware (CPU) capaz de abranger a memória, software, interrupções e transações de barramento (*bus*)
- Idealizado para ser uma região para funções de segurança do computador (ou dispositivo) abrigando um único TEE, inicialmente disponível para o firmware do dispositivo
- O TrustZone separa os softwares em dois tipos:
  - **Normal World:** não podem acessar recursos diretamente
    - Destinado a softwares de grande porte e grandes pilhas de execução como sistemas operacionais. Geralmente mais vulneráveis a ataques.
  - **Trusted World:** acesso livre aos recursos
    - Destinado a softwares “pequenos”, com uma superfície de ataque proporcionalmente pequena como: algoritmos de cifração/decifração, processamento de chaves entre outros.

ARM CCA

TrustZone – *Exception Level* (EL):

- Os *Exception Levels* tem como finalidade representar o grau de privilégio de execução (privilégio no acesso à memória e aos recursos do processador)
- Numerados de 0 a 3. Quanto mais alto, maior o grau de privilégio.
- A arquitetura ARM não fixa qual software deve ser utilizado em qual exception level. Mas há uma recomendação:
  - EL0 – Alguma aplicação em geral
  - EL1 – Sistema Operacional
  - EL2 – Hypervisor
  - EL3 – Firmware e programas de segurança
- Um EL só pode mudar quando: capturar uma exceção, retornar de uma exceção, reset, no modo *debug* e ao sair do modo debug



ARM CCA

TrustZone – *Exception Level* (EL):

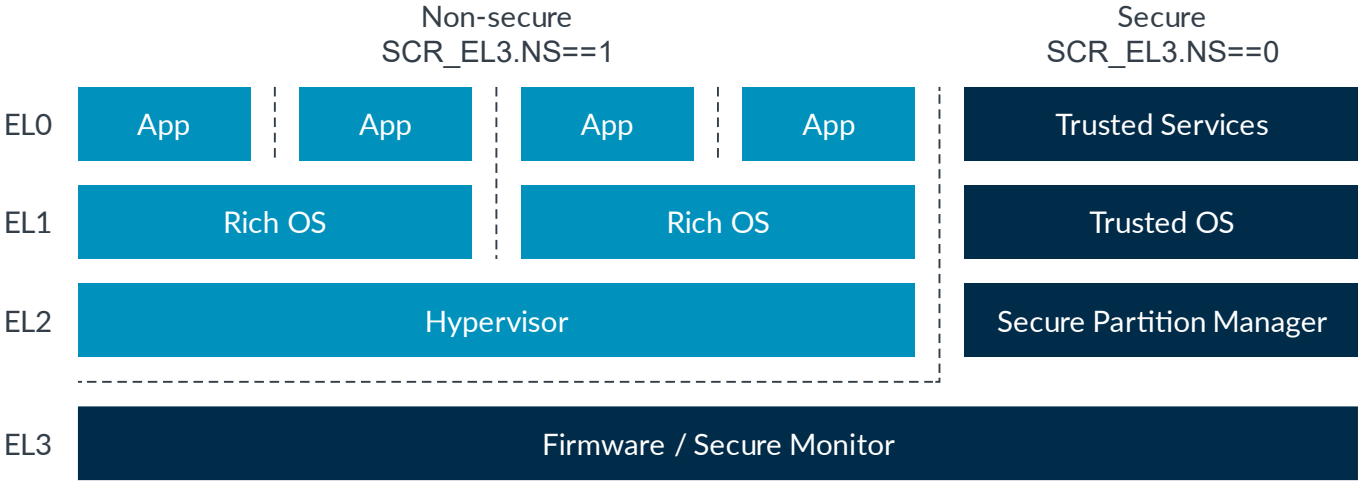
- Os privilégios de ELs são usados no acesso:
  - À Memória: regiões de memória podem ser configuradas na *Memory Management Unit* (MMU) para leitura/escrita (R/W) de determinados ELs.
    - A própria tabela da MMU pode ter seu acesso R/W configurado para determinados ELs
  - Aos recursos do processador como registradores:
    - Registradores com sufixos **"_ELx"** indicam que o grau mínimo de privilégio para acessar tal registrador é o ELx

Register	Name	Description
Exception Link Register	ELR_ELx	Holds the address of the instruction which caused the exception
Exception Syndrome Register	ESR_ELx	Includes information about the reasons for the exception
Fault Address Register	FAR_ELx	Holds the virtual faulting address

ARM CCA

TrustZone – *Security States*:

- Representa os “worlds” no processador, sendo:
  - Secure State (Trusted/Secure World) e Non-secure State (Normal World)
- **Security States e ELs**
  - O grau de privilégio EL3 só é possível no *Secure State*
  - Os demais (EL0, EL1 e EL2), em qualquer State
- Com relação ao acesso da memória: Non-secure State só pode acessar memória Non-secure. Já o Secure State pode acessar tudo.



ARM CCA

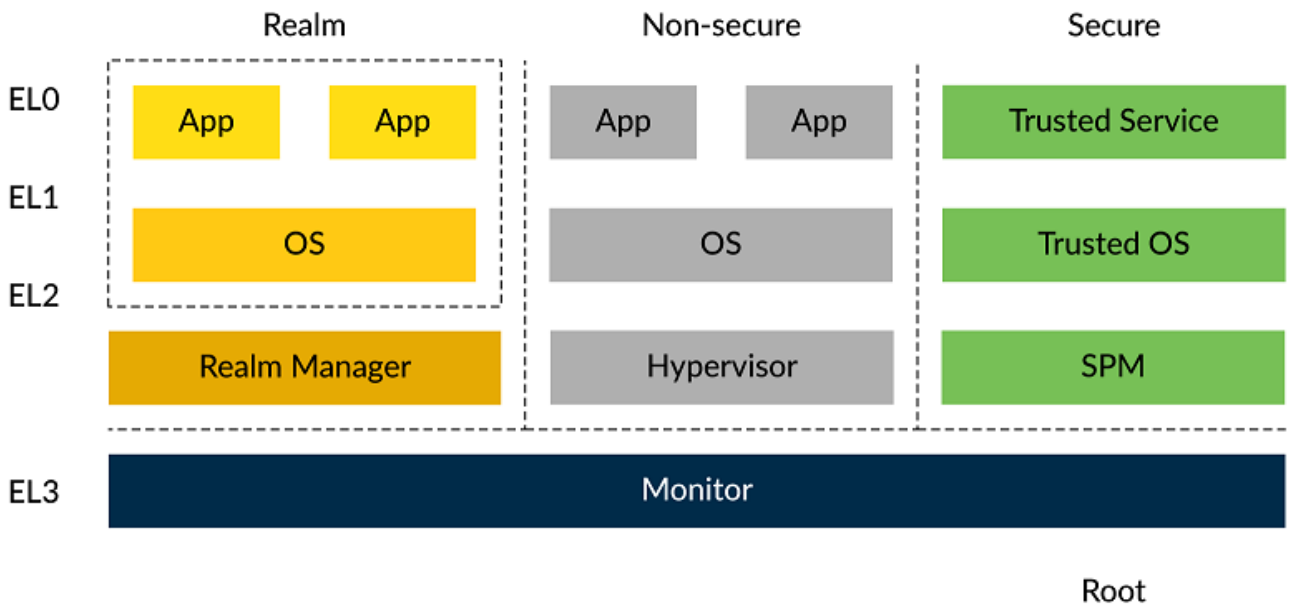
ARM CCA – *Confidential Compute Architecture*

- Novo componente na arquitetura: ***Realm Management Extension*** **(RME)** implementado em hardware e software a partir do Armv9
- Herda todas as definições do **TrustZone**, mantendo compatibilidade
- Conceito de **Realm**: ambiente de execução protegida, agora disponível também para aplicações que precisem executar em TEEs.
  - Um Realm pode ser criado a partir de um *Non-secure State*.
  - Por ex.: um *hypevisor* pode instanciar um VM em um Realm e não ter acesso aos dados dela.
  - Capacidade de dinamicamente transferir memória e recursos para um endereçamento de memória que softwares com alto privilégio ou o TrustZone não são capazes de acessar
  - Permite que softwares de baixo privilégio de execução como aplicações e VMs consigam proteger seus conteúdos

ARM CCA

ARM CCA – *RME Security States*

- Expande os Security States de 2 originalmente (TrustZone) para 4:
  - Secure state (TrustZone)
  - Non-secure state (TrustZone)
  - Realm state (novo) – ambiente de computação protegida
  - Root state (novo)



Exception Level EL3 Reservado para o Root state (software/firmware inicial)

ARM CCA

ARM CCA – *Endereçamento de memória física*

- Existem 4 tipos de espaços de endereçamentos de memória:
  - Secure (original do TrustZone)
  - Non-secure (original TrustZone)
  - Realm
  - Root
- A tabela abaixo indica a possibilidade de acesso de acordo com o Security State corrente:

Tipo de endereçamento	Secure State	Non-secure State	Realm State	Root State
Secure	Sim	Não	Não	Sim
Non-secure	Sim	Sim	Sim	Sim
Realm	Não	Não	Sim	Sim
Root	Não	Não	Não	Sim

ARM CCA

ARM CCA – *Endereçamento de memória física*

- Existem 4 tipos de espaços de endereçamentos de memória:
  - Secure (original do TrustZone)
  - stZone)

O tipo de endereçamento **Non-secure** pode ser acessado por todos Security States

O **Security State Root** pode acessar todos os tipos de endereçamentos

- A abaixo indica a possibilidade de acesso de acordo co Security S orrente:

ipo de e ereçamento	Secure State	Non-secure State	Realm State	Root State
S cure	Sim	Não	Não	Sim
Non-secure	Sim	Sim	Sim	Sim
Realm	Não	Não	Sim	Sim
Root	Não	Não	Não	Sim

ARM CCA

ARM CCA – *Verificação do Acesso à Memória*

- Quando um endereço de memória virtual é convertido em física, há uma tabela para a conversão e validação do acesso.
- As variáveis consideradas são:
  - *Security State* corrente
  - EL corrente
  - Tabelas de tradução (endereço virtual → físico)
  - Registradores do sistema (configurações de acesso)

ARM CCA

Observações finais:

- Nem todas arquiteturas/especificações precisam implementar TrustZone ou o CCA
- Dependendo da série ARM Cortex-M ou Cortex-A a implementação pode ser diferente:
  - Cortex-M: microcontroladores ou processadores simplificados (voltados principalmente para IoT).
    - O TrustZone para esse caso funciona de modo diferente do Cortex-A.
    - Não existe a camada de *Secure Monitor*, isto é, o processador opera o TrustZone de modo diferente, mas com o mesmo resultado
  - Cortex-A: processadores voltados para computação mais intensa (desktops e servidores)

Ataque CacheZoom

CacheZoom

Definição

- CacheZoom é uma ferramenta desenvolvida por pesquisadores para realizar a demonstração de um ataque (furto de chave criptográfica AES) de um programa sendo executado dentro de um enclave implementado por SGX

CacheZoom

Principais Caraterísticas

- Realiza ataque do tipo *side-channel*
- Ataca o cache do processador, em especial o L1D (L1 de dados)
- Se apoia no fato de que, mesmo o enclave estando presente na memória RAM cifrada, os dados presentes no cache não estão cifrados
- O ataque especificamente é o *Prime+Probe*

CacheZoom

Principais Cara

- Realiza ataq
- Ataca o cach
- Se apoia no
  RAM cifrada,
- O ataque esp

Prime + Probe (extraído do artigo):

- **Prime**: o atacante preenche os *sets* de memória cache com dados que ele conhece.
- **Acesso da vítima**: o atacante aguarda o acesso da vítima aos *sets* de memória cache, desejando que a vítima faça uso de alguma chave criptográfica.
- **Probe**: o atacante realiza acessos aos dados que ele gravou anteriormente na memória. Se a leitura for obtida muito rapidamente, significa que os dados ainda estão na memória cache. Se foram obtidos com maior tempo de resposta, significa que não estão na memória cache, ou seja, os endereços de memória relacionados à memória cache foram utilizados pela vítima. Deste modo o atacante consegue obter os endereços de memória que a vítima usou e, por consequência, possíveis informações da chave criptográfica utilizada.

CacheZoom

Preparação e Condições do Ambiente

- Atacante tem acesso total aos recursos do SO - permissão para instalar módulos kernel (*root*) e alterar propriedades do *boot*
- Linux OS (Ubuntu 14.04.5 LTS) modificado para que o enclave e o programa do atacante sejam executados no mesmo *core*
- Restante de serviços e processos do SO serão executados nos demais cores (facilitar a eficácia do CacheZoom)
- *Hyper-threading* (*thread* adicional por core) é desativado para que somente uma linha de execução por core seja permitida

CacheZoom

Preparação e Condições do Ambiente (cont.)

- Acesso ao binário do enclave → algoritmo AES utilizando chave de 128 bits
- Offset das tabelas de substituição
- Conhecimento dos dados (*payload*) a serem cifrados
- HW: Intel(R) Skylake Core(TM) i7-6500U i7 (com 2 cores) - L1D com 32kB e L1I com 32kB
- *Pointer chasing eviction set technique* → para encontrar os ponteiros associados com o cache
  - "In the specific case of our L1D cache, the access time for chasing 8 pointers associated to a specific set is about 40 cycles on average."

CacheZoom

Execução

- O CacheZoom, na preparação proposta, faz interrupções com frequência em curtos períodos ao programa concorrente (enclave) para obter dados.
- Tais interrupções são muito rápidas, de modo que o perfil de uso do cache L1 pode ser obtido.
- Os autores indicam que a resolução espacial e temporal são muito altas.

CacheZoom

Eficiência

- Dependendo do algoritmo AES implementado, a descoberta da chave pode ser mais ou menos eficiente: T-Table, S-Box

Table 2. Vulnerable implementations in popular current cryptographic libraries. These implementations can be configured through compile/runtime settings.

Library	Vulnerable implementations
OpenSSL 1.1.0f	`aes_core.c` T-table, `aes_x86core.c` Large T-table, S-box and prefetching configurable through AES_COMPACT_IN_INNER_ROUNDS, AES_COMPACT_IN_OUTER_ROUNDS
WolfCrypt 3.11.0	`aes.c` T-Table with prefetching before the first round
Mozilla NSS 3.30.2	`rijndael.c` T-Table and S-box configurable through RIJNDAEL_GENERATE VALUES_MACRO
Nettle 3.3	`aes-encrypt-internal.asm` T-table
Libtomcrypt 1.17	`aes.c` T-table
Libgcrypt 1.7.7	`rijndael.c` T-table, S-box for the last round with prefetching
MbedTLS 2.4.2	`aes.c` T-table, S-box for the last round

CacheZoom

Como obter a chave (baseado no artigo referenciado pelo CacheZoom *1)

- Foco em obter chave AES de outro processo na máquina
- 2 cenários: furto da chave ao cifrar; furto da chave ao decifrar
- Atacante utiliza a mesma biblioteca (OpenSSL) que a vítima
- Demonstra “como” obter a chave AES:
  - Preenche memória cache (usando todas linhas). Nota: a linha é parte de um endereço de memória
  - Lê a memória cache e descobre as “linhas” sobrescritas a partir do tempo de resposta (alto nesse caso)
  - Visto que a implementação do AES utiliza parte da chave no endereço de memória (questão de implementação), ao descobrir as linhas sobrescritas, parte da chave pode ser descoberta, principalmente no caso de se conhecer o “texto claro”

*1 Fonte: Ashokkumar, C., Giri, R.P., Menezes, B.: Highly efficient algorithms for AES key retrieval in cache access attacks. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P)

Ataque Crossline

CrossLine

CrossLine: *Breaking "Security-by-Crash" based Memory Isolation in AMD SEV*

- Desmitificam o uso da **ASID** (**Address Space ID** - identificação da VM):
  - Os autores indicam que o uso da tecnologia poderia ser melhor
- Propõe um ataque específico, intitulado ***CrossLine***, para personificar uma VM
- Vantagem do CrossLine: ele é indetectável, ao contrário dos outros ataques
- Premissa: o hypervisor é modificado pelo atacante
- Resultado: atacante consegue ler a memória

Crossline

- **Definições:**
  - **CR3**: registrador de controle onde se localiza uma Page Table (PT)
  - **gCR3**: endereço de memória onde se localiza a tabela gPT
    - Utilizada por uma VM
  - **nCR3**: endereço de memória onde se localiza a tabela nPT (*nested page table*)
    - Utilizada pelo Hypervisor
- **Conversão endereço virtual de uma guest VM em real:**
  - gVA → gPT → gPA → nPT → sPA
- **VMCB (Virtual Machine Control Block)**: Registradores e bits de controle de uma determinada VM
- **VMSA: VM Save Area** (existente a partir do SEV-ES)

Legendas:

- PA: Physical Address
- PT: Page Table
- gVA: guest Physical Address
- gPT: guest Page Table
- PTE: Page Table Entry
- CR3: endereço da PT (real)

Obs.: informações sobre páginas de memória e hypervisor: http://www.cs.cmu.edu/~410-f06/lectures/L31_Virtualization.pdf

CrossLine

CrossLine v1 – preparação

- O hypervisor e uma VM adicional são necessárias ao lado da VM vítima
- Deste modo:
  - O hypervisor é modificado e integra o ataque
  - VM vítima é iniciada (SEV)
  - VM atacante é iniciada como non-SEV (o hypervisor modificará o ASID em runtime)

CrossLine

CrossLine v1 – passo a passo

1. Limpar o bit de presença
   - Limpa o bit de presença de todos PTEs (Page Table Entries) da VM atacante
   - A ideia é futuramente provocar um *page fault*, isto é, o mapeamento de gPA em sPA vai falhar e será reconstruído
2. Remapear o gCR3 corrente da VM atacante:
   - Hypervisor modifica o registrador gCR3 da VM atacante e o aponta para uma página da VM vítima
3. Modificar o VMCB da VM atacante:
   - Hypervisor copia o ASID da VM vítima para a VM atacante
4. Especificar o *offset* da página alvo:
   - Hypervisor modifica o ponteiro de instrução (nRIP) e aponta ele para o endereço da página de memória da VM vítima que vai ser atacado

CrossLine

CrossLine v1 – passo a passo (cont.)

5. Extração de segredos a partir de falhas das páginas de memória *Nested*

   - VMRUN na VM atacante → o processador obtém a próxima instrução (nRIP), mas como a TLB está limpa (PTEs limpos), vai obtê-los a partir do gCR3 copiado.

   - A operação vai falhar, mas os dados vão ser extraídos e notificados para a VM atacante (campo EXITINFO2 da VMCB) contendo um bloco de 8 *bytes* de dados

Realização de *dump* das tabelas de páginas da VM vítima:

- Por meio da repetição dos passos, é possível alterar a página de memória a ser atacada, bem como o ponteiro de instrução para, aos poucos, obter blocos de 8 bytes de dados em cada ataque.

CrossLine

CrossLine v2 – preparação

- Idêntica a preparação do CrossLine v1
- Outro objetivo: executar código na VM Vítima
  - Instrução: “`MOVL $2020, %r15d`”
  - Ler dados
  - Executar instruções na memória da VM vítima

CrossLine

CrossLine v2 – passo a passo

1. Preparar nPT
   - Limpa o bit de presença de todos PTEs (Page Table Entries) da VM atacante
   - A instrução mencionada a ser executada encontra-se no endereço gVA0
   - O hypervisor prepara cinco mapeamentos gPA para sPA: para os 4 níveis de gPFNs e a página de instrução respectivamente
2. Configurar nRIP
   - O hypervisor configura o nRIP para o gVA0. Também limpa o flag de interrupção (registrador RFLAG.IF) no VMCB para que a instrução seja diretamente executada

CrossLine

CrossLine v2 – passo a passo (cont.)

3. Alterar ASID

   - O hypervisor substitui o ASID da VM atacante (pela da VM vítima), sinaliza o VMCB como *dirty* e executa a VM atacante (VMRUN).

   - Ao final, %r15 (registrador 15) estará com o valor $2020, indicando que o ataque foi bem sucedido.

CrossLine

Observações dos autores

- Conseguiram encontrar a página de memória que roda o *sshd* (SEV)

- Observações para SEV-ES:
  - CrossLine v1 modificado ainda obtém sucesso.
    - Contudo perde parte de sua “invisibilidade” (*stealthness*), pois precisa alterar um registrador de controle (CR2)
  - CrossLine v2 não é mais possível
- Observações para SEV-SNP:
  - CrossLine v1: não funcionará
  - CrossLine v2: visto que não mais no SEV-ES, não se aplica ao SEV-SNP
  - Autores indicam que ainda há um caminho, mas com baixa probabilidade de sucesso

CrossLine

Trabalhos Relacionados (1/2)

- Unencrypted VMCB
  - Ocorre antes do SEV: o conjunto de registradores VMCB ficava decifrado
  - Permite manipulação de registradores das VMs, permitindo ataques do tipo "*return oriented programming*" ROP
- Unauthenticated Encryption
  - Ocorre desde o SME
  - O hypervisor pode alterar a memória cifrada de uma VM, pois tem o acesso
  - Ataques nas versões iniciais conseguiam reduzir a segurança, permitindo o teste de força bruta para uma equivalência de 32 bits
- Unprotected nPT
  - Acesso indevido permite o mapeamento do perfil de uso de memória
  - Alterações podem ser realizadas para o furto (vazamento) de dados: modificação em uma página para que a VM vítima produza uma resposta a partir de uma página de memória forjada

Trabalhos Relacionados (2/2)

- Unprotected I/O
  - HWs que comunicam com a VM, utilizam IOMMU e DMA
  - Buffer de tradução: memória decifrada <-> memória cifrada
  - Hypervisor pode modificar e alterar esses dados
- Ciphertext accessibility
  - Primeira família de ataques ao SEV-SNP
  - Monitoram o VMSA (região do VMCB) que está cifrada desde a versão do SEV-ES
  - Atacantes monitoram o perfil dos registradores dentro do VMSA
  - O objetivo é furtar chaves RSA e o ECDSA nonce da OpenSSL
  - AMD criou patches (microcódigo) para mitigar esse problema

CrossLine

Tabela – Vulnerabilidades x Mitigação

Research Papers	Exploited Vulnerabilities	I/O Interaction	Breach Confidentiality	Breach Integrity	Stealthiness	Mitigated by
Du *et al.* [8]	Unauthenticated encryption	✓	✗	✓	✗	SEV-SNP
Buhren *et al.* [6]	Unauthenticated encryption	✓	✓	✗	✗	SEV-SNP
Wilke *et al.* [28]	Unauthenticated encryption	✓	✓	✓	✗	SEV-SNP
Werner *et al.* [27]	Unencrypted VMCB	✓	✓	✗	✗	SEV-ES
Hetzelt & Buhren [10]	Unencrypted VMCB Unprotected PT	✓	✓	✓	✗	SEV-SNP
Morbitzer *et al.* [20]	Unprotected PT	✓	✓	✗	✗	SEV-SNP
Morbitzer *et al.* [19]	Unprotected PT	✓	✓	✗	✗	SEV-SNP
Li *et al.* [17]	Unprotected I/O Unauthenticated encryption	✓	✓	✓	✗	SEV-SNP
Li *et al.* [16]	Ciphertext accessibility	✓	✓	✗	✓	Hardware Patch
CrossLine V1	Security-by-Crash	✗	✓	✗	✓	SEV-SNP
CrossLine V2	Security-by-Crash Unencrypted VMCB	✗	✓	✓	✓	SEV-ES

- **I/O Interaction**: Indica se o ataque necessita interagir com aplicações hospedadas dentro da VM por meio de operações E/S (disco, rede, etc.)
- **Breach Confidentiality**: Indica quebra de confidencialidade
- **Breach Integrity**: Indica quebra de integridade
- **Stealthiness**: Indica se o ataque/vulnerabilidade é detectável pela VM atacada

Tecnologia – RISC-V - Sanctum

RISC-V Sanctum

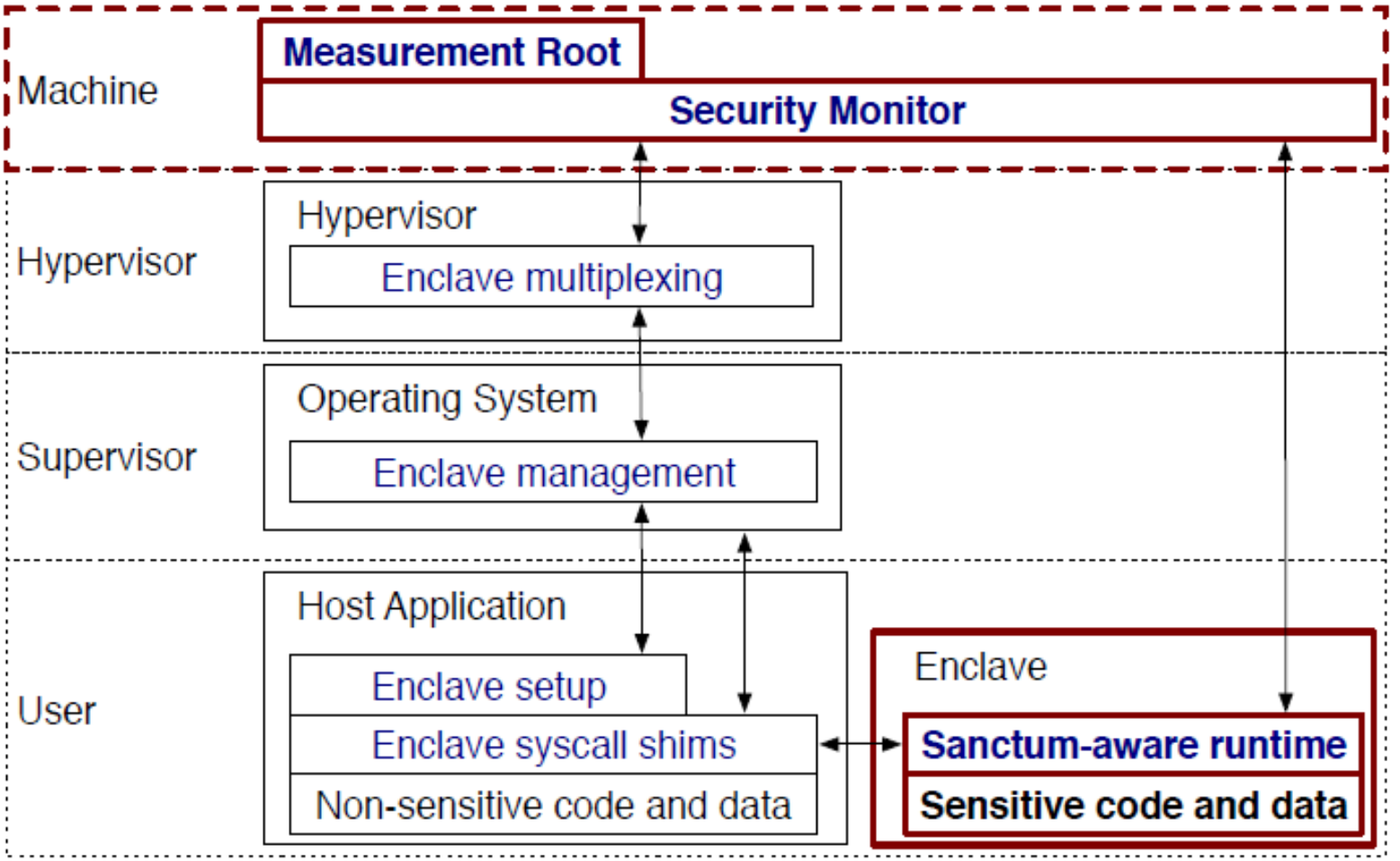
- Implementação aberta (https://github.com/ilebedev/sanctum)
- Baseado no Rocket RISC-V core
- Proposta de ser uma alternativa ao Intel SGX
- Muita lógica como software – de acordo com os autores isso melhora a transparência

RISC-V Sanctum

- **Observações dos autores sobre as outras tecnologias (2016):**
  - Muitas linhas de código
  - Muitas vulnerabilidades em SOs e hypervisors
- **Proposta:**
  - Sanctum cria um enclave (Security Monitor) por meio de software que limita o acesso aos endereços de memória aliado a uma mínima modificação de hardware
  - As alterações de HW se limitam à interfaces adicionadas entre blocos da arquitetura RISC-V, sem alterar as E/S dos blocos.
  - Programação análoga ao SGX
- **Não protege de:**
  - Escrita em memória *untrusted*
  - DoS
  - Outros componentes de HW: DRAM, ...
  - Outras situações: modificação da temperatura de operação, ...

RISC-V Sanctum

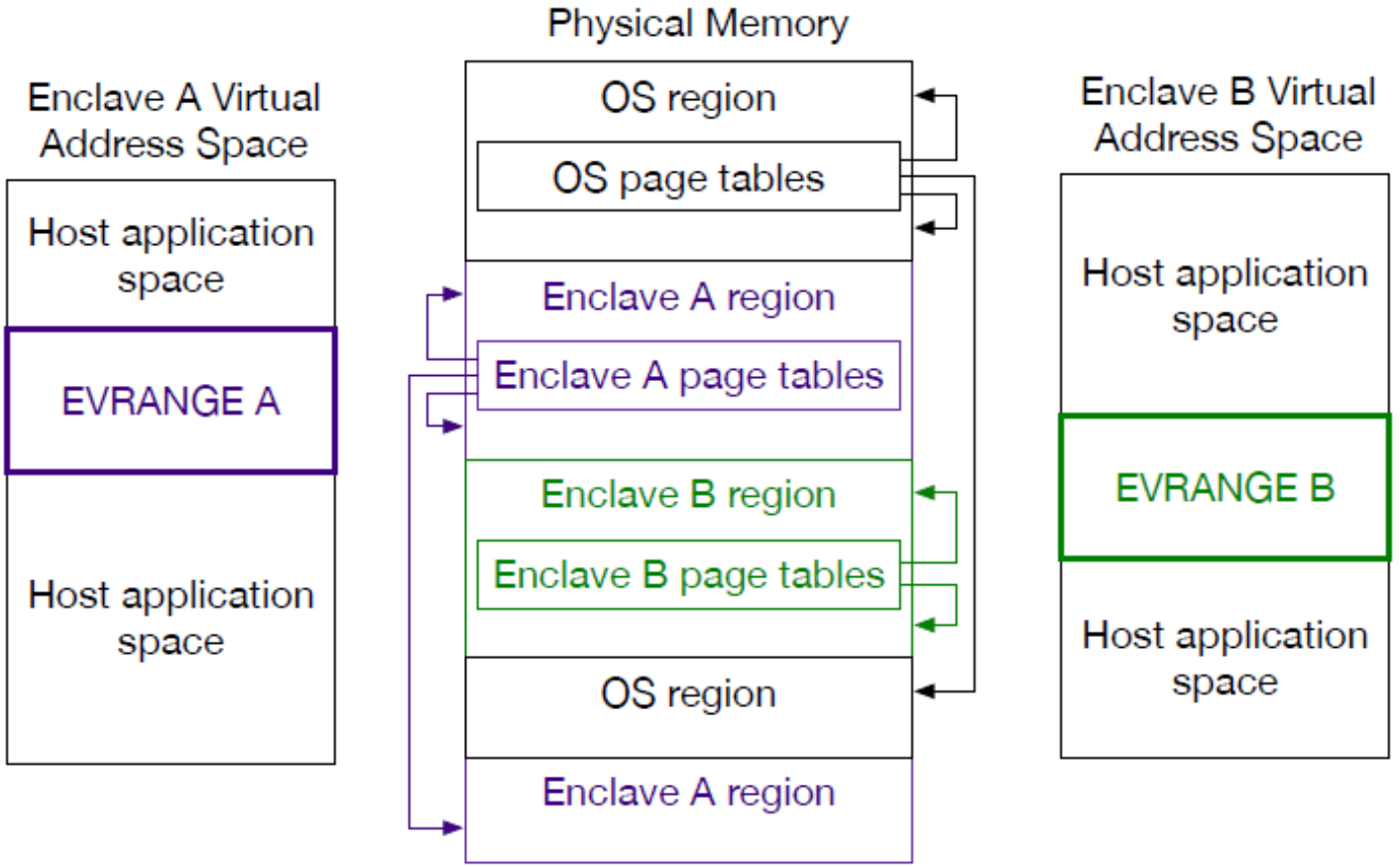
- **Pilha software do Sanctum**



- Texto na cor azul: Representa elementos adicionais necessários ao Sanctum
- Texto em negrito: Elementos no software da **TCB (***Trusted Computing Base*)

RISC-V Sanctum

- **Tabela de páginas de memória por enclave**



EVRANGE: faixa de endereçamento virtual de um enclave

RISC-V Sanctum

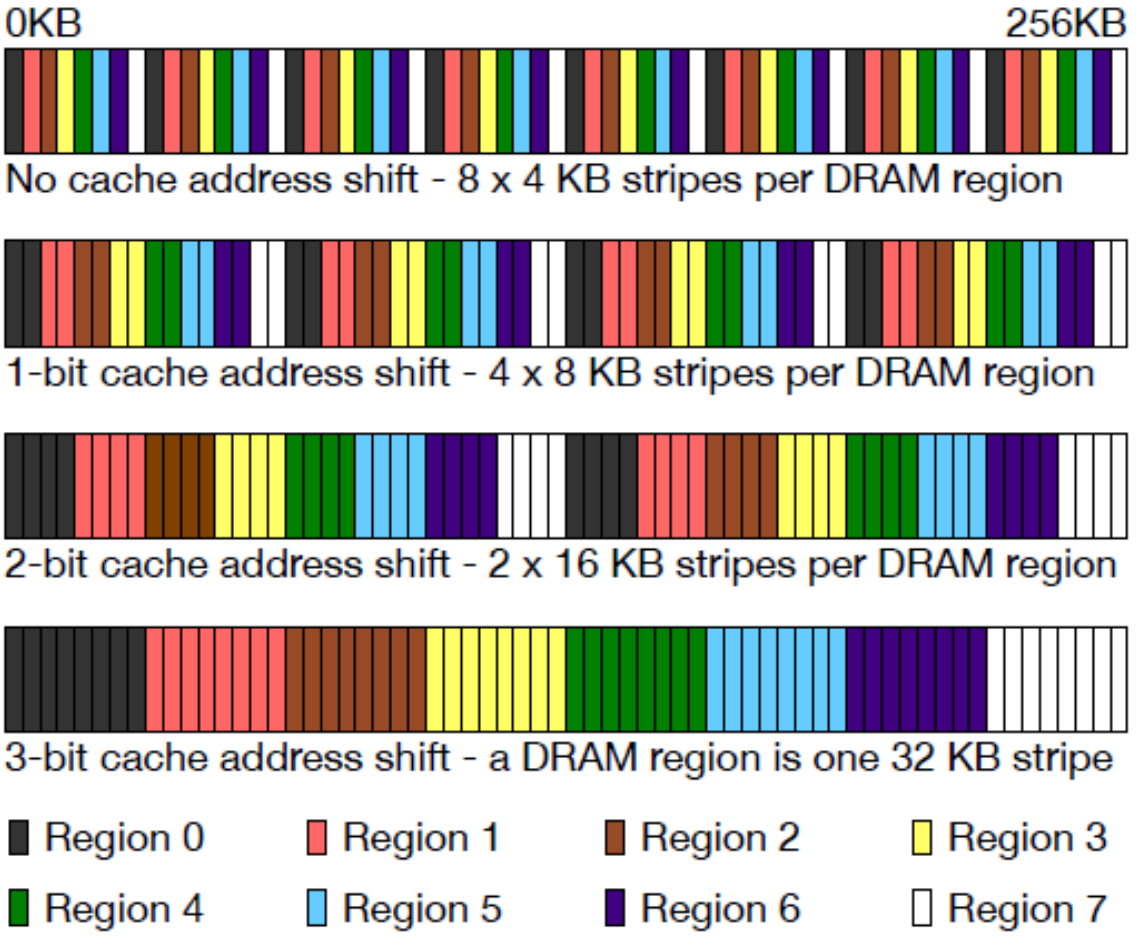
- **Modelo parecido com SGX:**
  - Enclaves não podem fazer operações I/O – nem *syscalls*
  - Ciclo de vida: um programa solicita ao SO a criação de um enclave
  - Quoting Enclave: enclave especial para atestação
- **Mas com funcionamento diferente**:
  - Enclaves só comunicam com sua parte *untrusted* via interrupção (assíncrono)
  - Solução não suporta *hyperthreading*
  - Importante ressaltar que o *Security Monitor* é o primeiro software a tratar uma interrupção (devido ao seu nível de privilégio)
  - O Security Monitor salva os registradores e depois zera eles

RISC-V Sanctum

- **Mas com funcionamento diferente (cont.)**:
  - Ao continuar a execução do enclave, os registradores são restaurados pelo *Security Monitor*.
  - Flush do cache L1 e TLB toda vez que há chaveamento entre programa no enclave e *untrusted*
  - O acesso ao TLB é feito diretamente pelo Security Monitor somente, ou seja, se o OS precisar, precisa usar o Security Monitor
  - Já o cache LLC é particionado utilizando extensões de HW

RISC-V Sanctum

- **Deslocamento de endereço por regiões contíguos de DRAM**

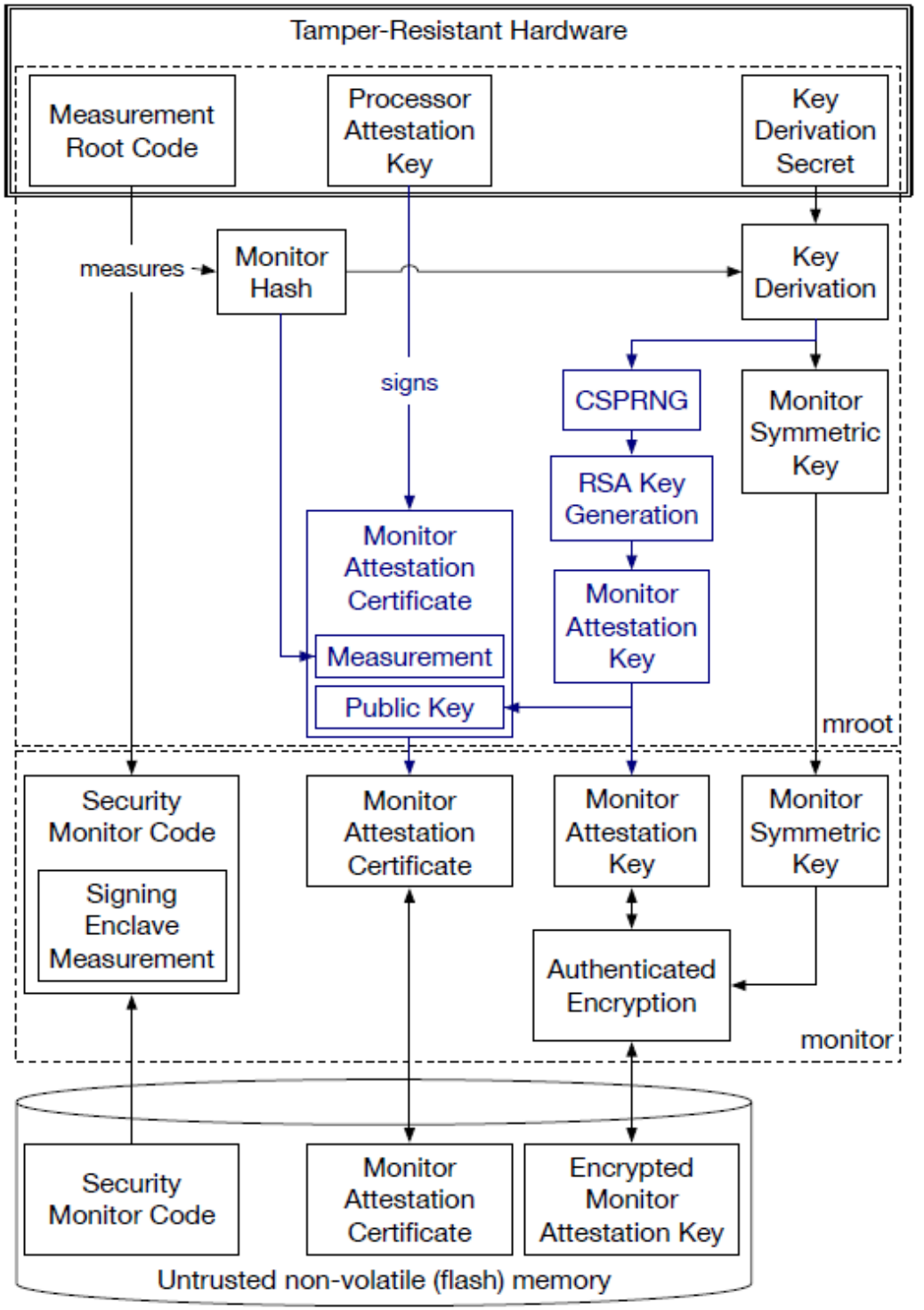


RISC-V Sanctum

- **Atestação**
  - Existem 3 softwares confiáveis:
    - **Measurement root**: gravado na ROM
    - **Security monitor**: gravado no firmware – flash memory
    - **Signing Enclave**: gravado em mídia untrusted

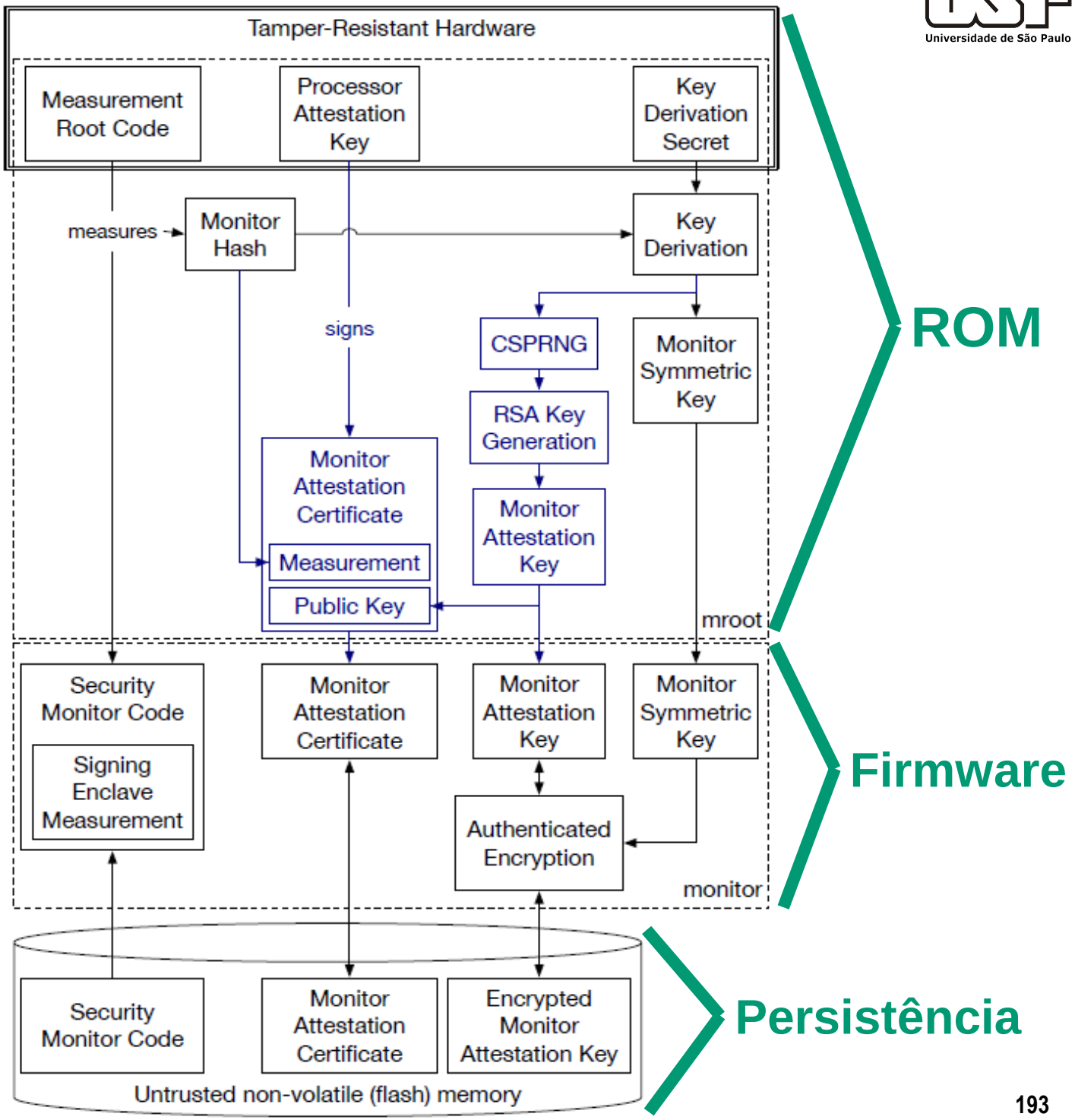
RISC-V Sanctum

Atestação - Boot



RISC-V Sanctum

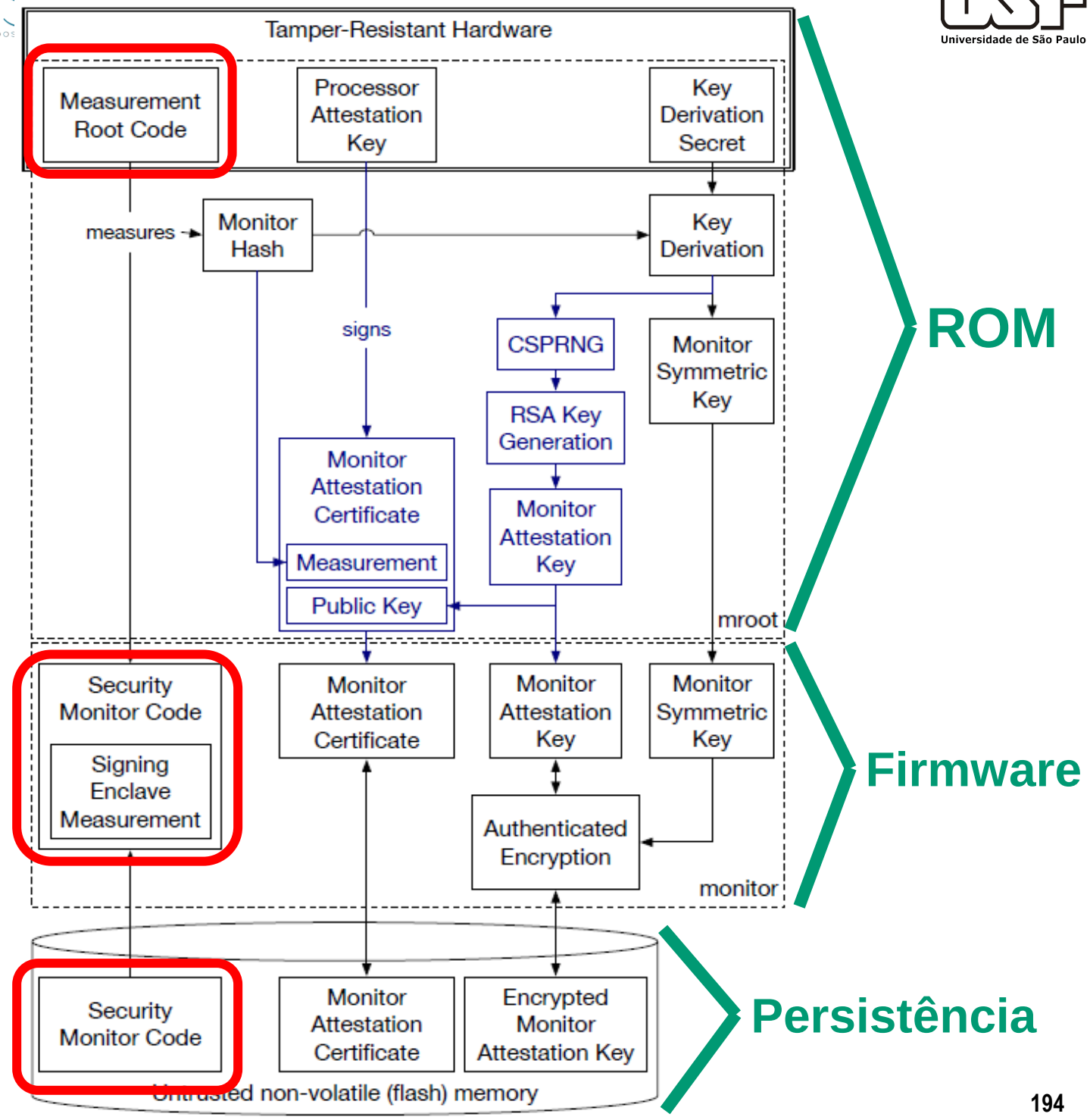
Atestação - Boot



RISC-V Sanctum

Atestação - Boot

Measurement Root calcula hash do *Security Monitor Code*

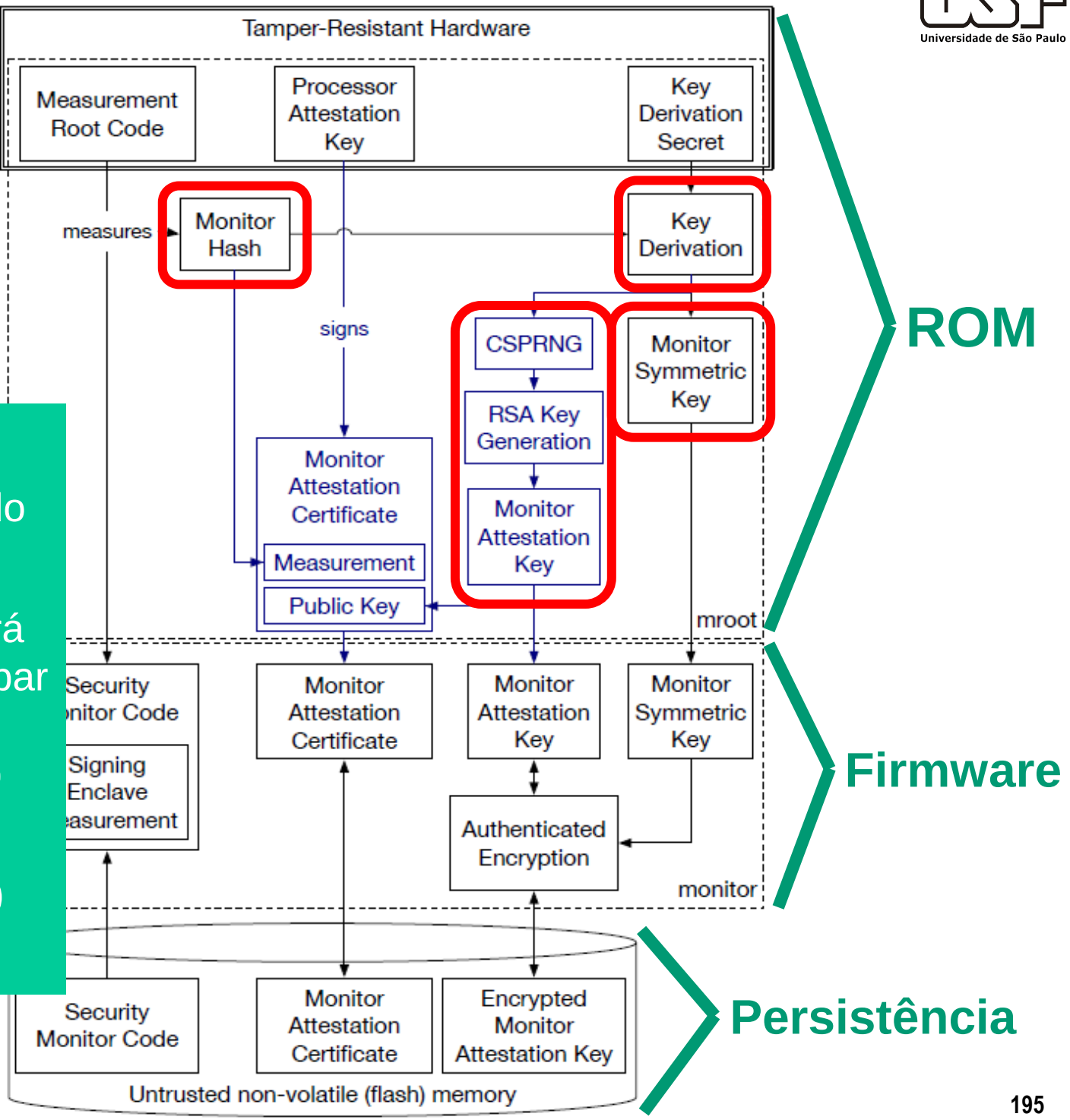


RISC-V Sanctum

Atestação - Boot

O *hash* produzido é utilizado para gerar:

- Uma chave RSA que será utilizada para compor o par de chaves de atestação (*Monitor Attestation Key*)
- Uma chave simétrica (*Monitor Symmetric Key*)



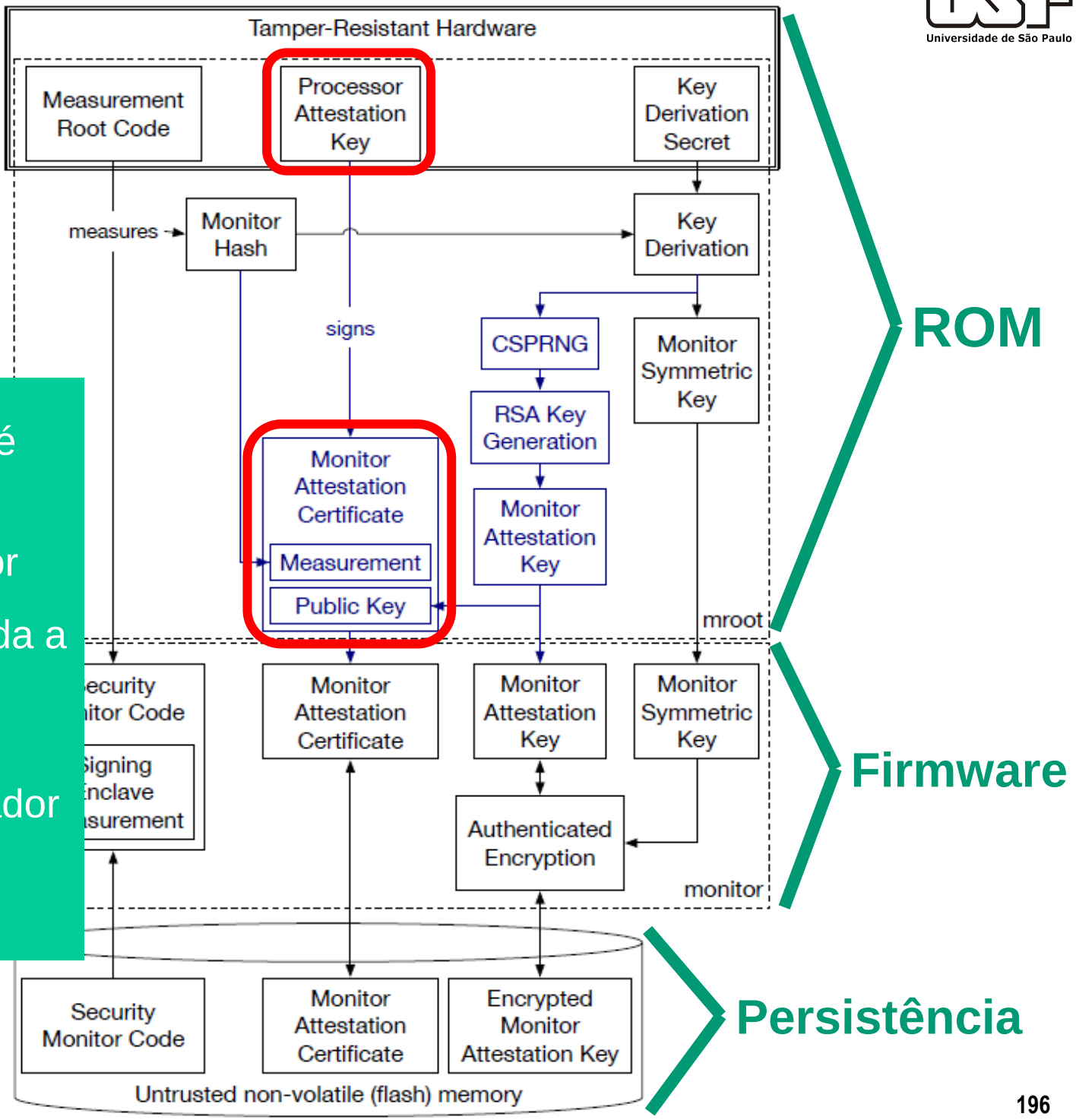
RISC-V Sanctum

Atestação - Boot

O certificado de atestação é gerado usando o:

- *Measurement* do Monitor
- Uma chave pública criada a partir da chave privada *Monitor Attestation Key*

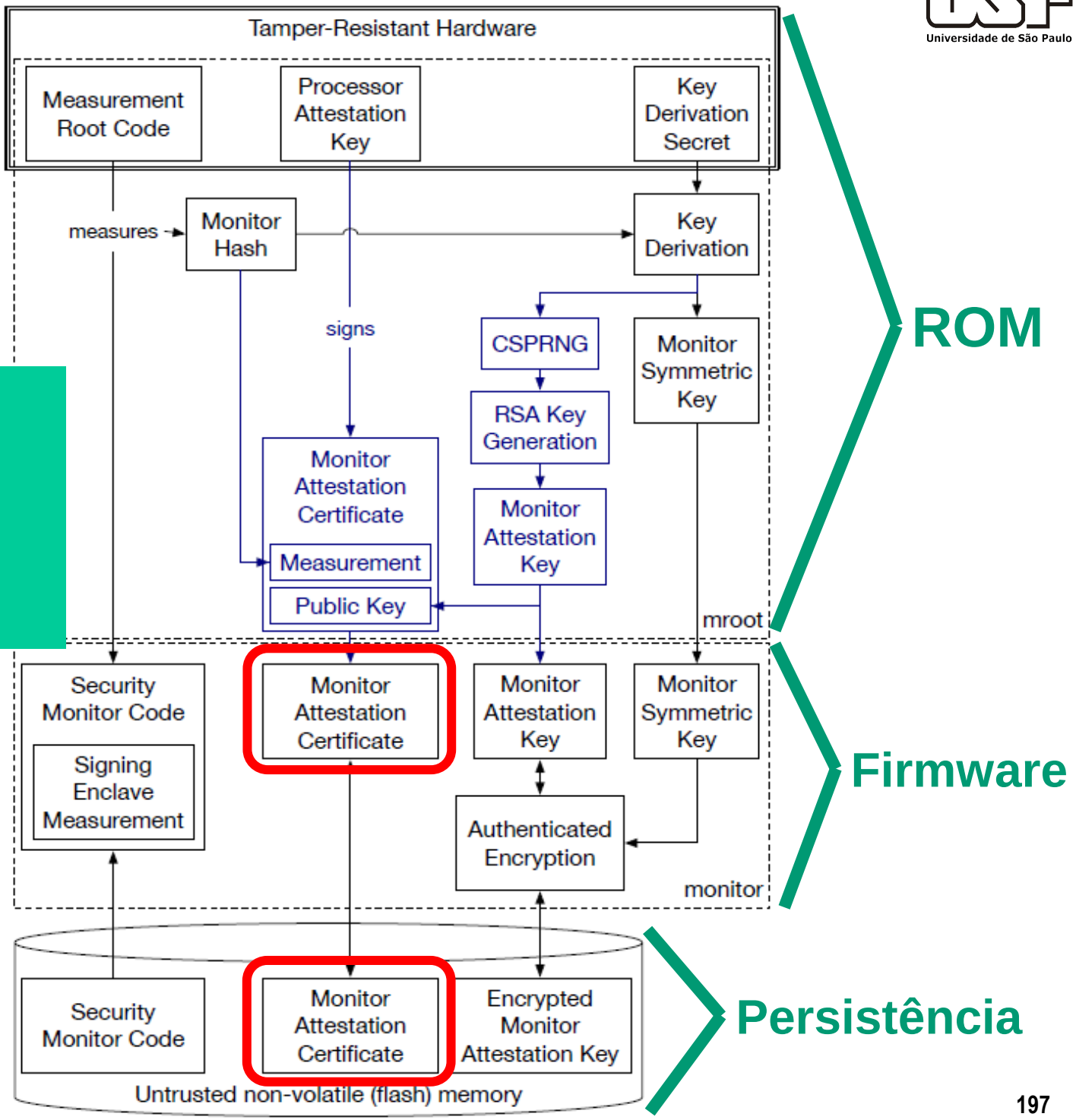
Após, a chave do processador é utilizada para assinar o certificado de atestação.



RISC-V Sanctum

Atestação - Boot

O certificado de atestação gerado pode ser persistido decifrado, pois se trata de um certificado público.

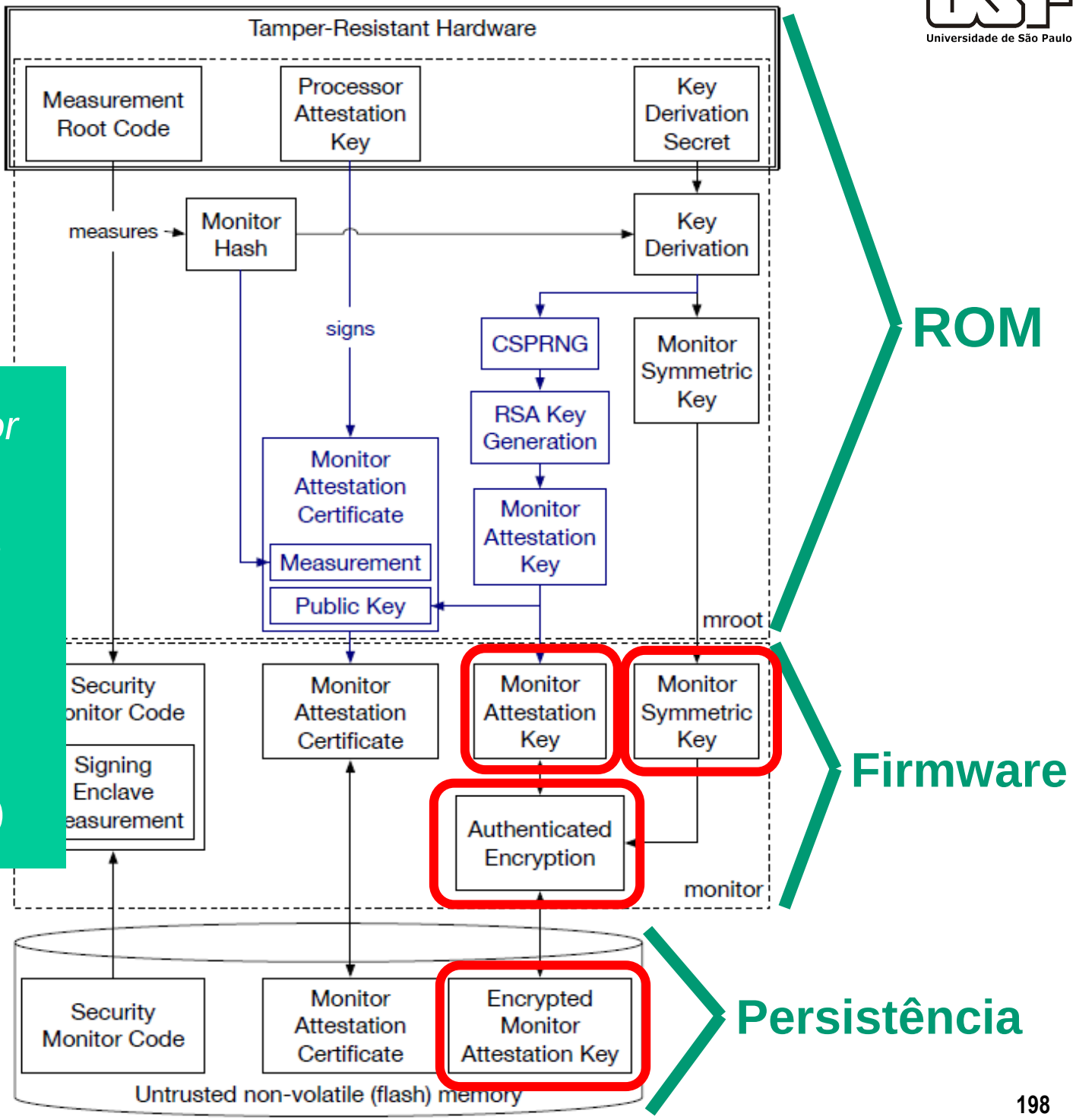


RISC-V Sanctum

Atestação - Boot

Já a chave privada (*Monitor Attestation Key*), é:

- Cifrada usando a chave simétrica (*Monitor Symmetric Key*)
- Depois de cifrada, é persistida (*Encrypted Monitor Attestation Key*)

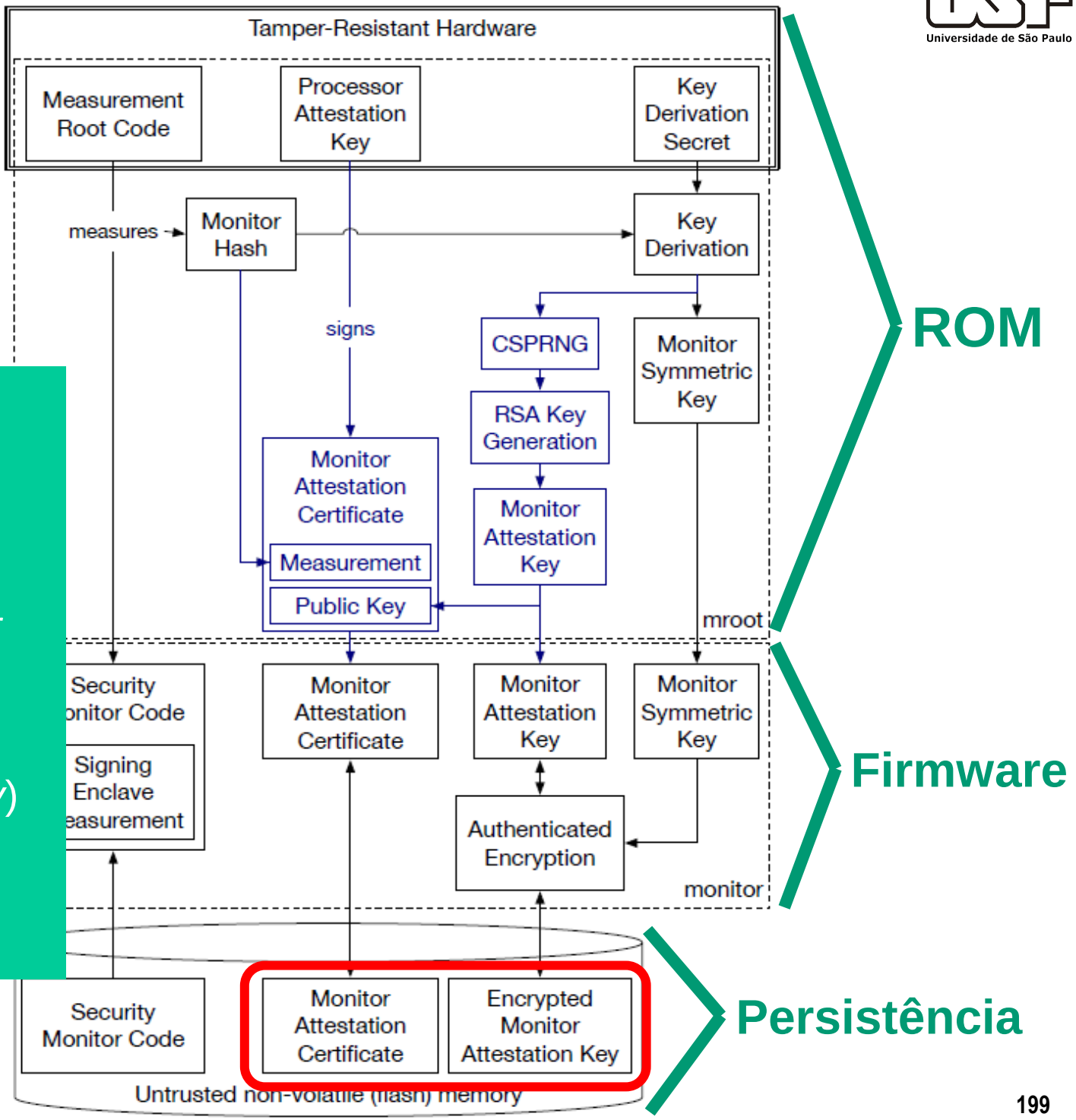


RISC-V Sanctum

Atestação - Boot

Por fim, após a primeira inicialização, serão produzidos:

- Um certificado público para atestação (*Monitor Attestation Certificate*)
- Uma chave privada (*Monitor Attestation Key*) cifrada na persistência (*Encrypted Monitor Attestation Key)*



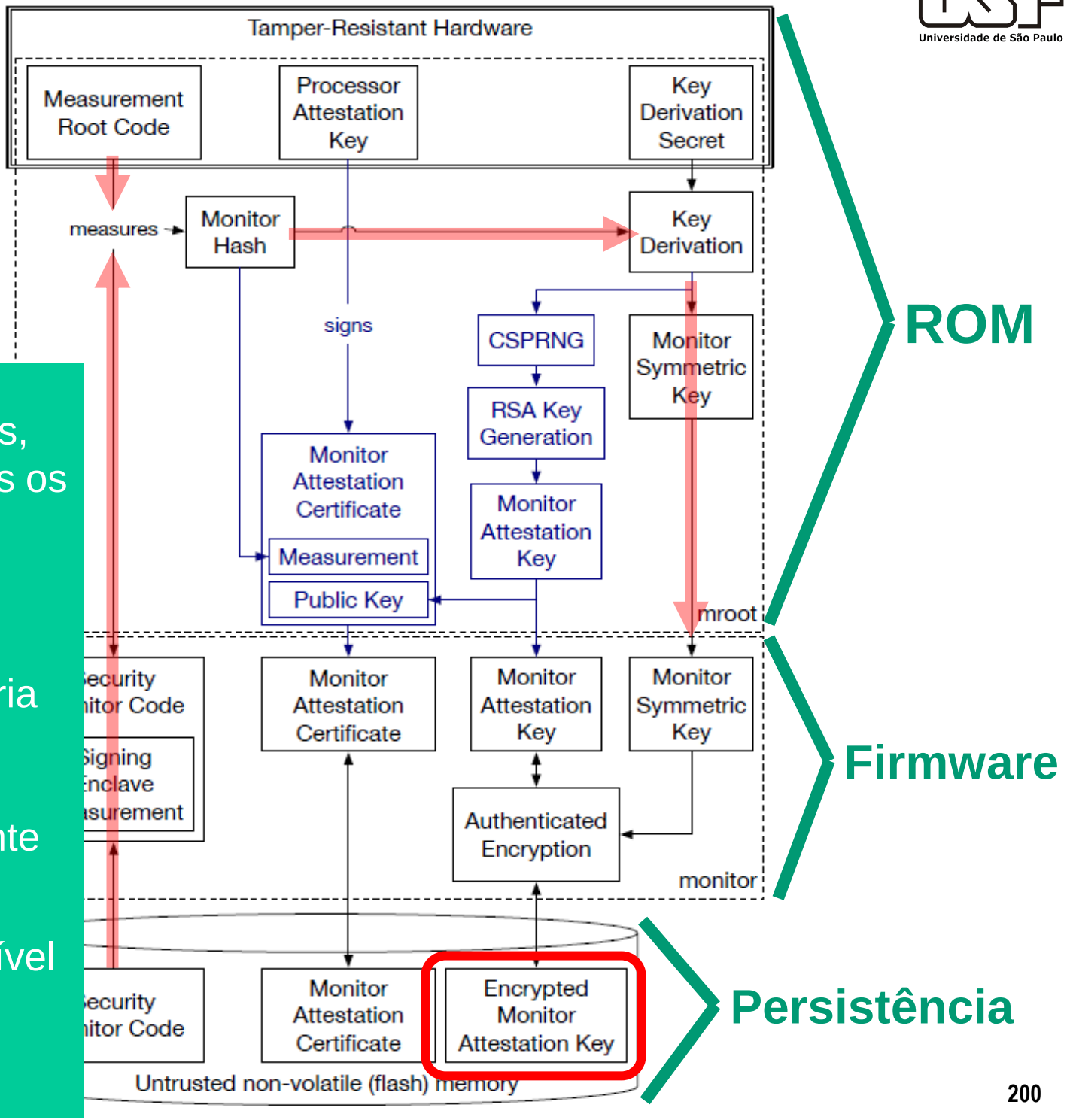
RISC-V Sanctum

Atestação - Boot

Nas próximas inicializações, somente serão necessários os passos até a obtenção da chave simétrica (*Monitor Symmetric Key*).

Isso porque ela é necessária para recuperar (decifrar) a chave privada (*Monitor Attestation Key*) previamente persistida.

Deste modo estará disponível o par de chave pública e privada para a atestação.



RISC-V Sanctum

Atestação:

- O Sanctum tem como premissa o fato de que há uma cadeia de certificados (e chaves) provenientes do fabricante do hardware (processador), sendo que:
  - O fabricante possui um certificado público válido e disponível
  - A partir certificado do fabricante, é criado um certificado (e chave privada) para cada um dos processadores RISC-V, que seja resistente a extração (*tamper resistant hardware*)
  - O *Monitor Attestation Certificate* (visto anteriormente) é assinado pela chave privada do processador (*Processor Attestation Key*), compondo o último elemento na cadeia

RISC-V Sanctum

Atestação (cont.):

- Processo:
  - O *measurement* de um enclave é obtido
  - Esse dado é passado para o *Signing Enclave* por meio de *Mailboxes* (técnica para não permitir o acesso direto ao Enclave)
    - Obs.: o Sanctum utiliza um enclave para a realização da assinatura de um enclave. Algo muito semelhante ao *Quoting Enclave* do Intel SGX
  - A assinatura de enclave é realizada
  - Os dados de atestação e o *measurement* são encapsulados de modos que possam ser remotamente verificados

RISC-V Sanctum

- **Contribuições**
  - Observações com relação à segurança: algo que foi explorado por outros autores – CacheZoom por exemplo.
  - Cenário proposto pelos autores (computação remota) se aplica ao Sanctum.

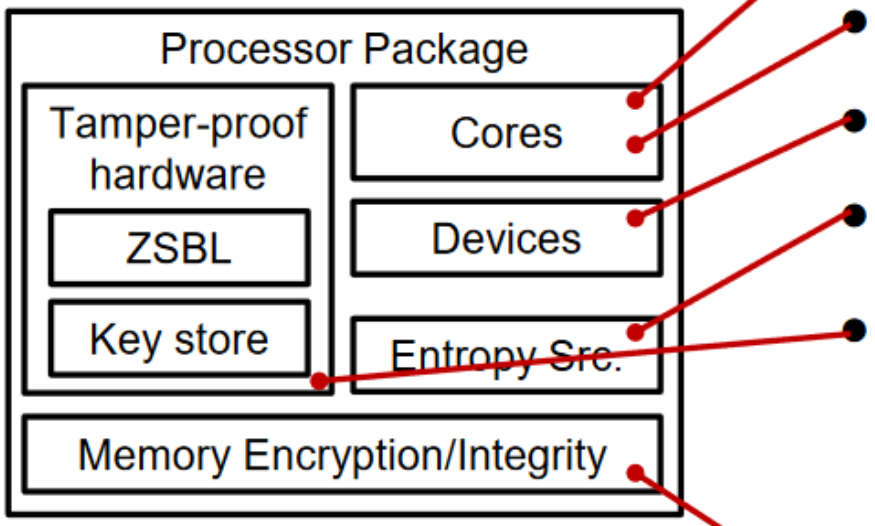
Tecnologia – RISC-V - Keystone

RISC-V Keystone

- Keystone: Proposta de enclave para RISC-V
- UC Berkley – Berkley Architecture Research
- Inspiração e continuação a partir do Sanctum (compartilham colaboradores)
- Utilizam extensão não-padrão do RISC-V:
  - PMP (*Physical Memory Protection*) introduzida em 2017 (RISC-V Priv. v1.10)
- Desenvolvida utilizando a placa “SiFive Unleashed”, que é uma implementação da arquitetura RISC-V pelo fabricante SiFive
- 3 modos de privilégio: user (U), supervisor (S) e machine (M)
- Última publicação do projeto em 2020.

RISC-V Keystone

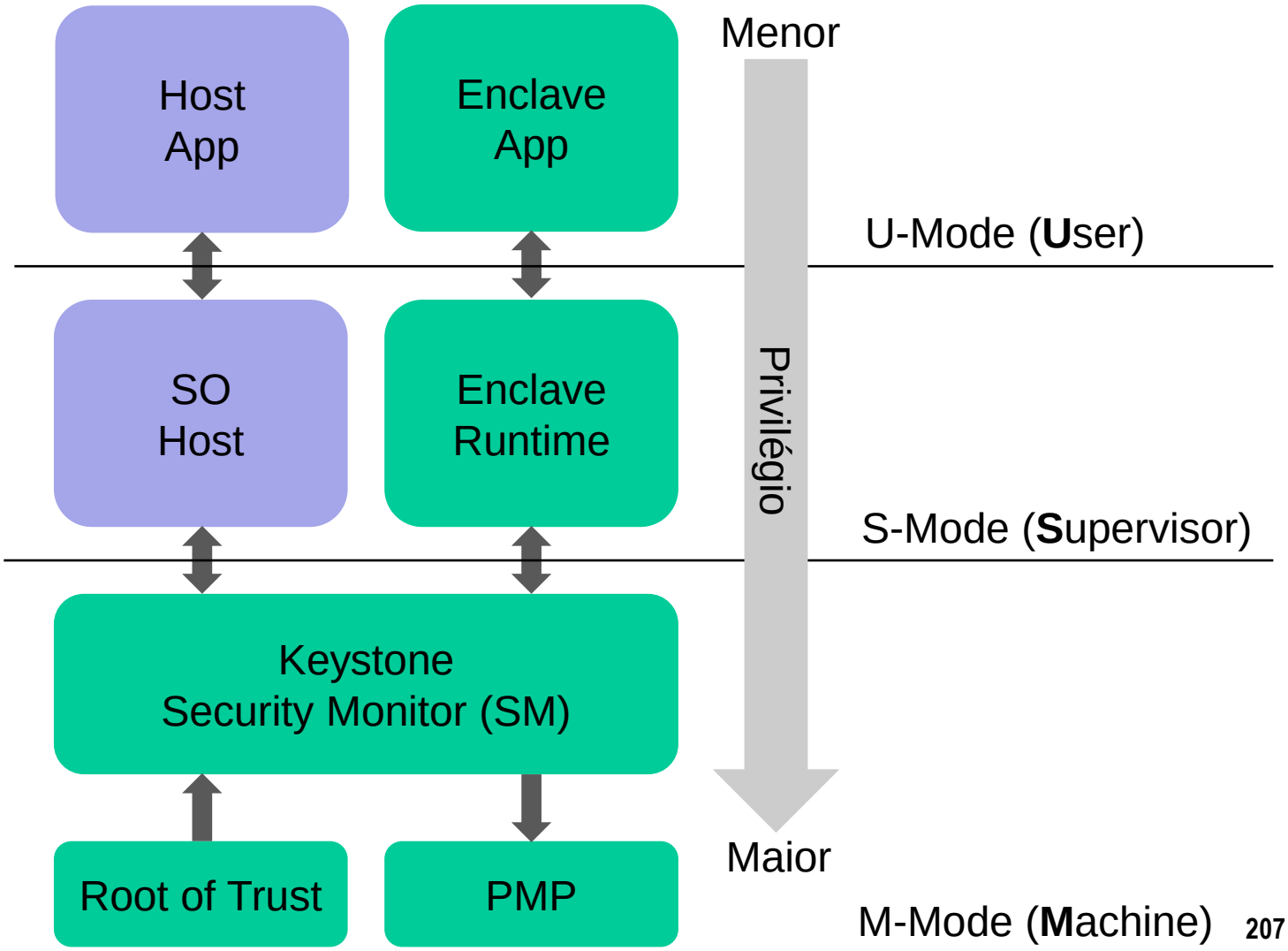
- **Componentes de hardware do processador RISC-V Keystone**
  (componentes já presentes bem como componentes opcionais)



- RISC-V Physical Memory Protection (PMP)
- RISC-V U-, S-, and M-mode
- (RISC-V) Device Gasket PMP (i.e., iopmp)
- An Entropy Source available at boot
- Root of Trust (preferably a crypto engine)
  - Measuring & signing the security monitor
  - Platform key store
- If untrusted/external DRAM – memory encryption/integrity engine (not implemented yet)

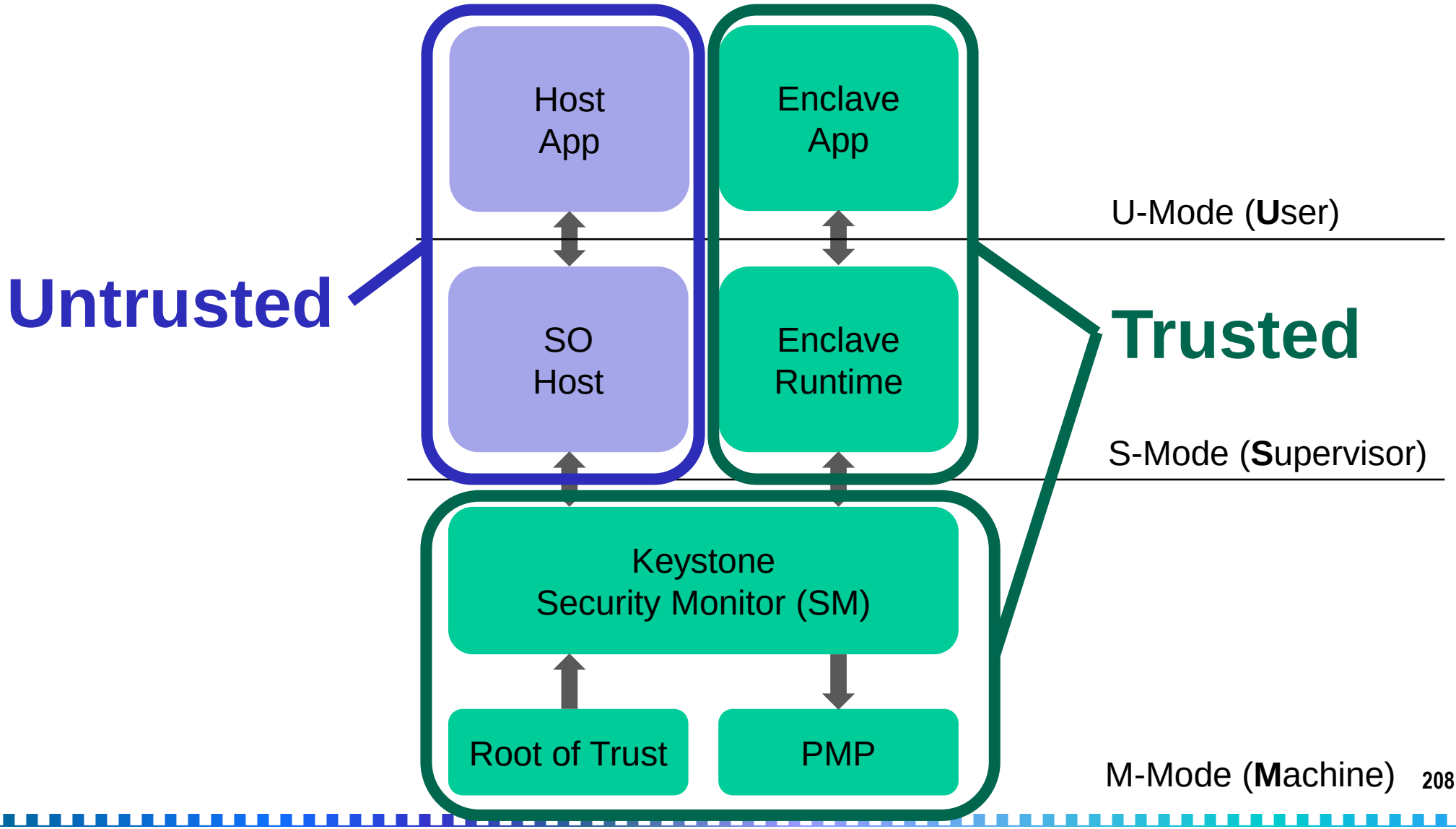
RISC-V Keystone

- **Principais componentes – Grau de privilégio**



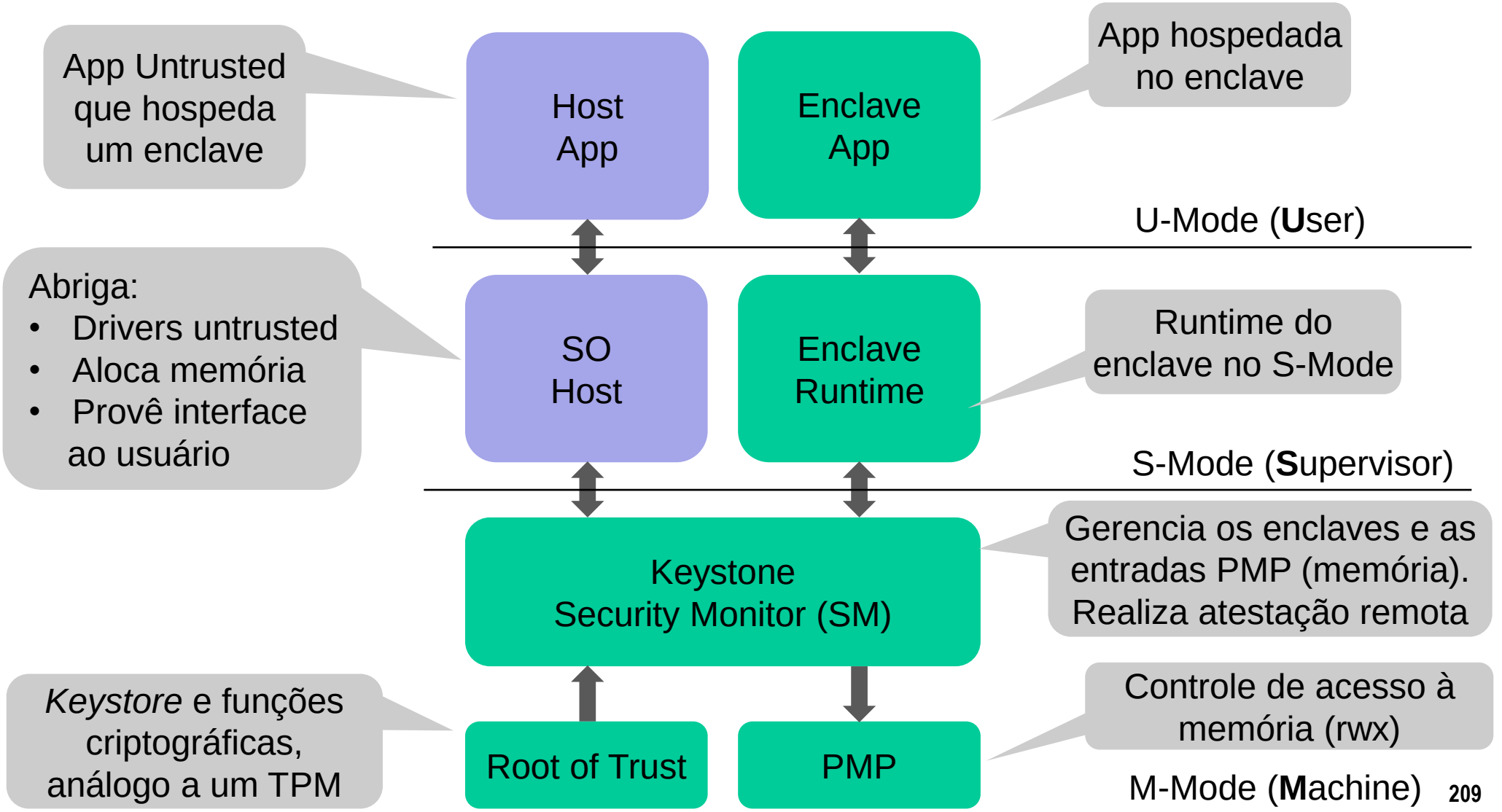
RISC-V Keystone

- **Principais componentes – Partes Trusted/Untrusted**



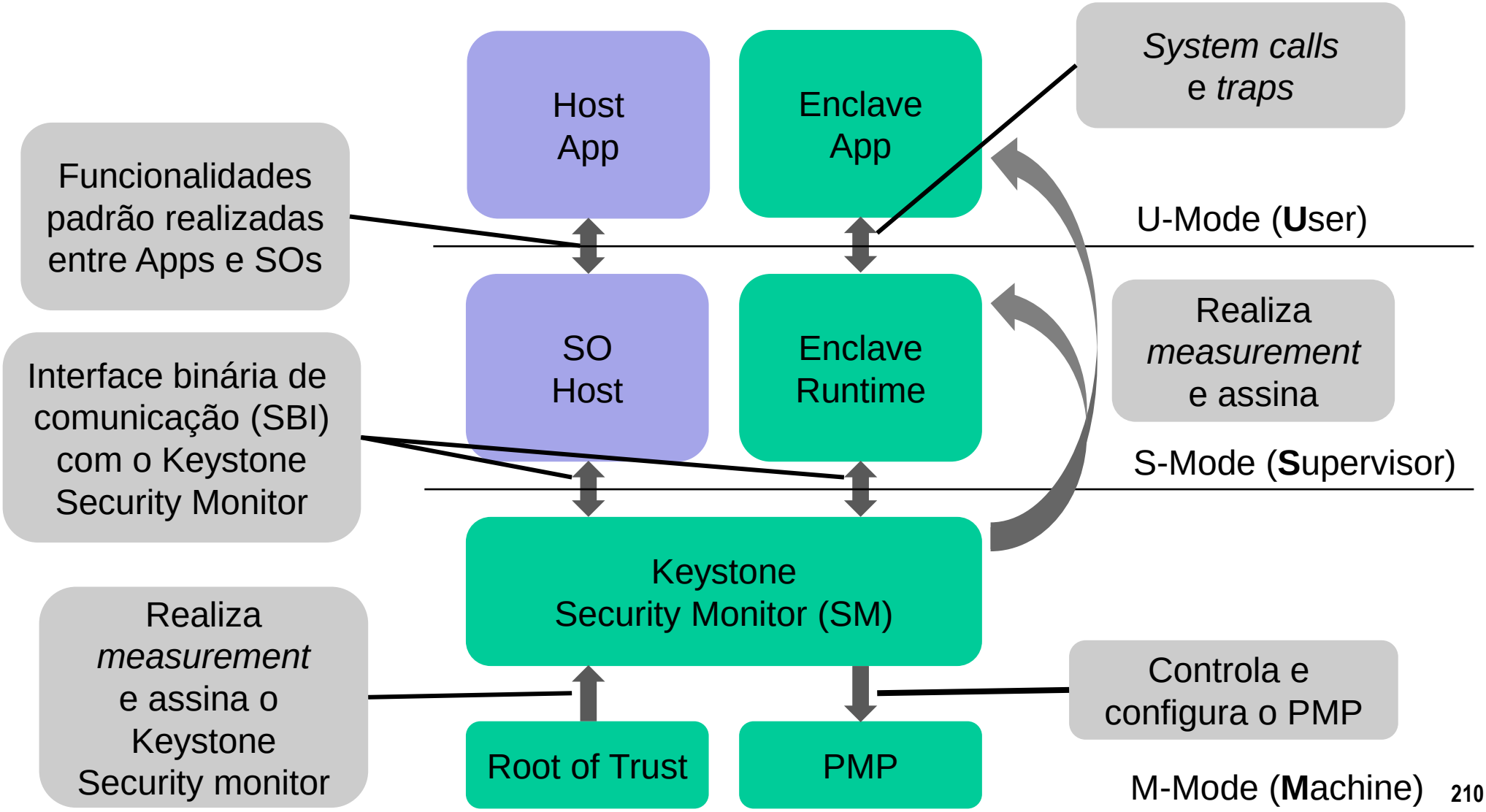
RISC-V Keystone

- **Principais componentes – Definições**



RISC-V Keystone

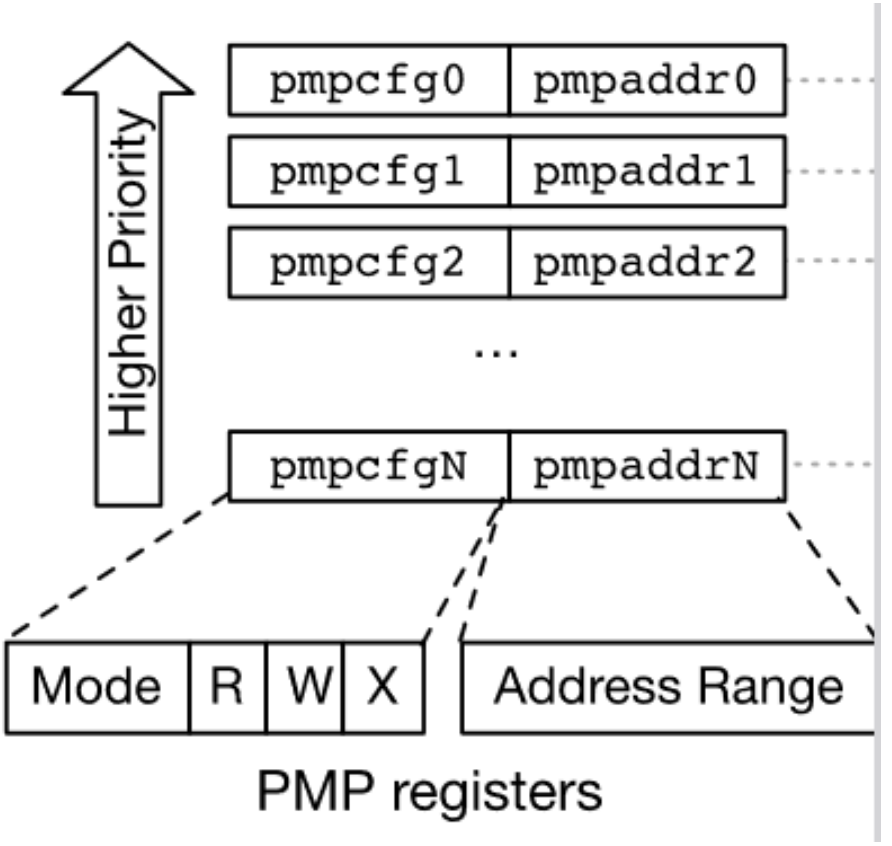
- **Principais componentes – Funcionalidades e comunicações**



RISC-V Keystone

Componente PMP

- Registradores para indicar permissões (rwx) nos modos **U** (user) e **S** (supervisor)
- Quantidade de entradas de PMP pode variar (uma implementação padrão possui 8)
- Priorizada estaticamente pela ordem do índice
- Baseado em *whitelist:*
  - se há permissão configurada → é utilizada
  - se não há permissão configurada → a operação é negada
- Modo M dinamicamente configurável
- Modo de endereçamento NAPOT*

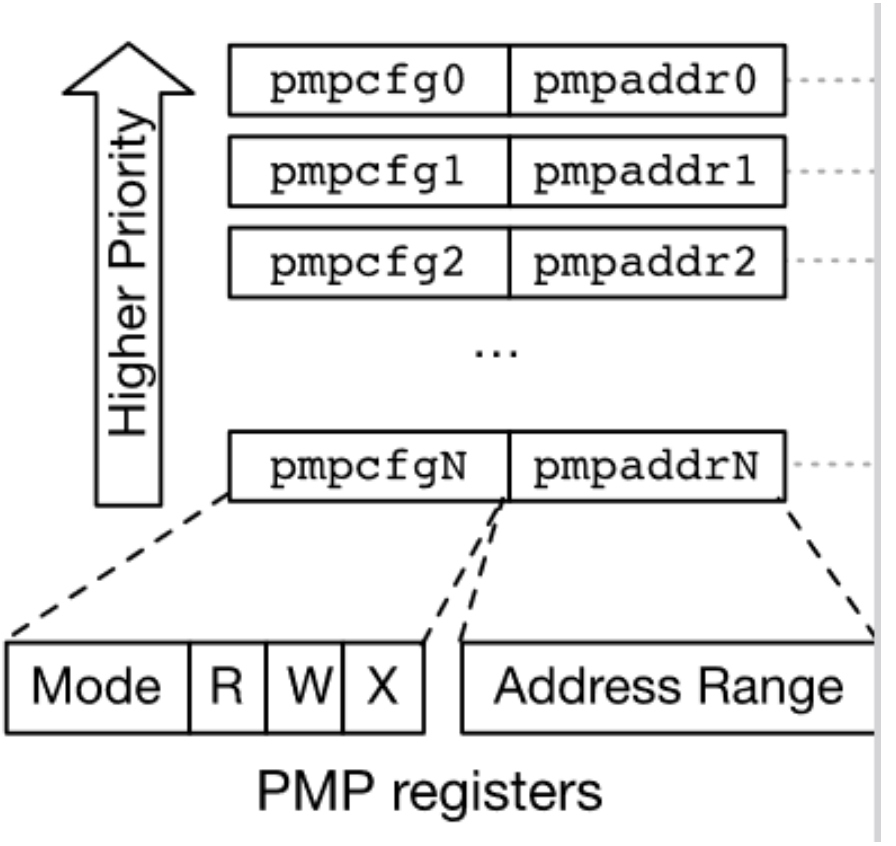


*NAPOT (*Naturally-aligned power-of-two*): Meio de representar uma faixa (*range*) de endereços (endereço inicial e tamanho)

RISC-V Keystone

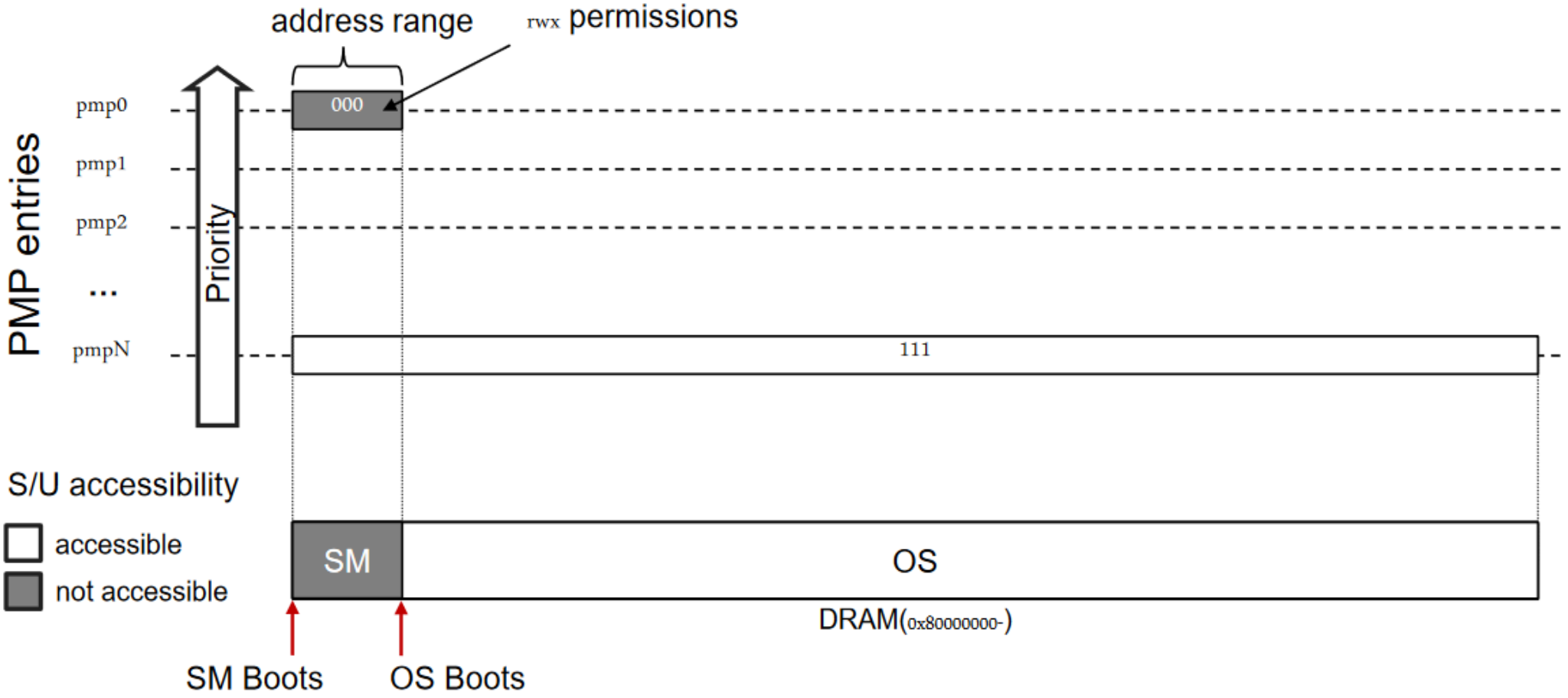
Como Keystone utiliza PMP:

- As entradas de “topo”/”baixo” são respectivamente reservadas para o SM e o SO:
  - SM (topo) → maior prioridade
  - SO (baixo) → menor prioridade
- 1 entrada PMP para cada enclave ativo
- Tamanho das faixas (*range*) representadas por NAPOT >= 4 KB



RISC-V Keystone

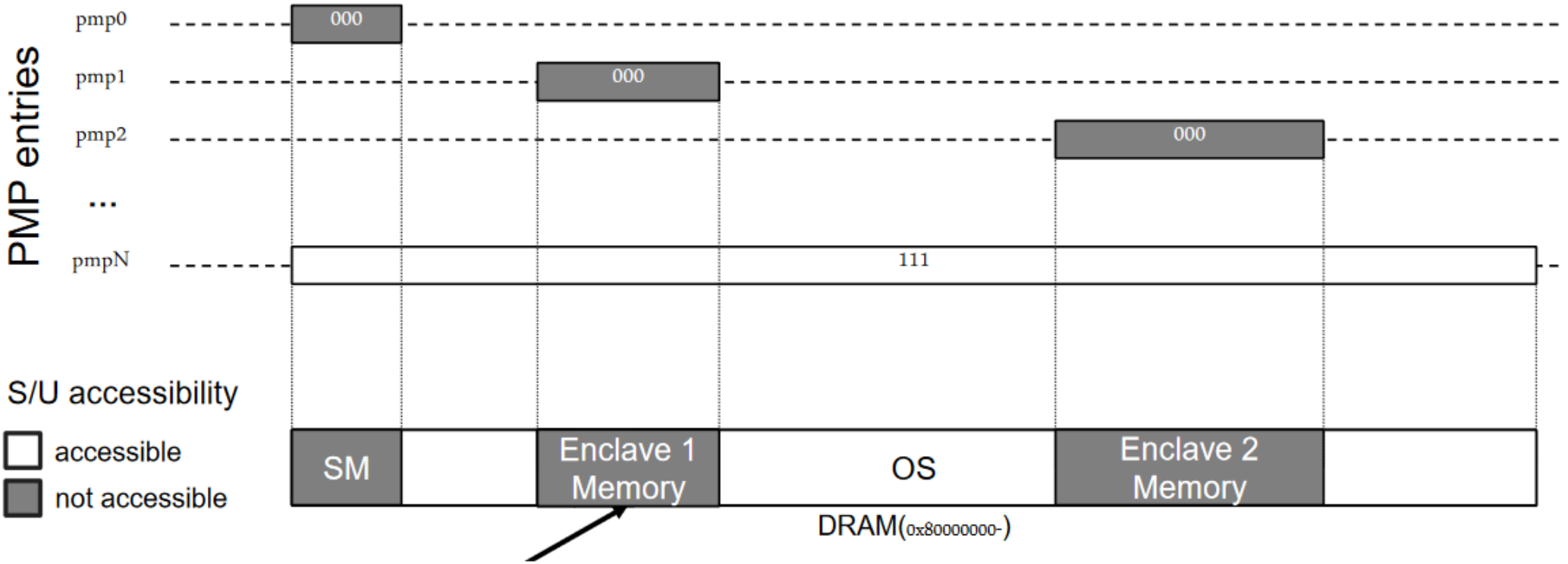
- **Componente PMP – Funcionamento**



O SM "garante" uma faixa de memória de alta prioridade que o SO não pode ler ou escrever

RISC-V Keystone

- **Componente PMP – Funcionamento**

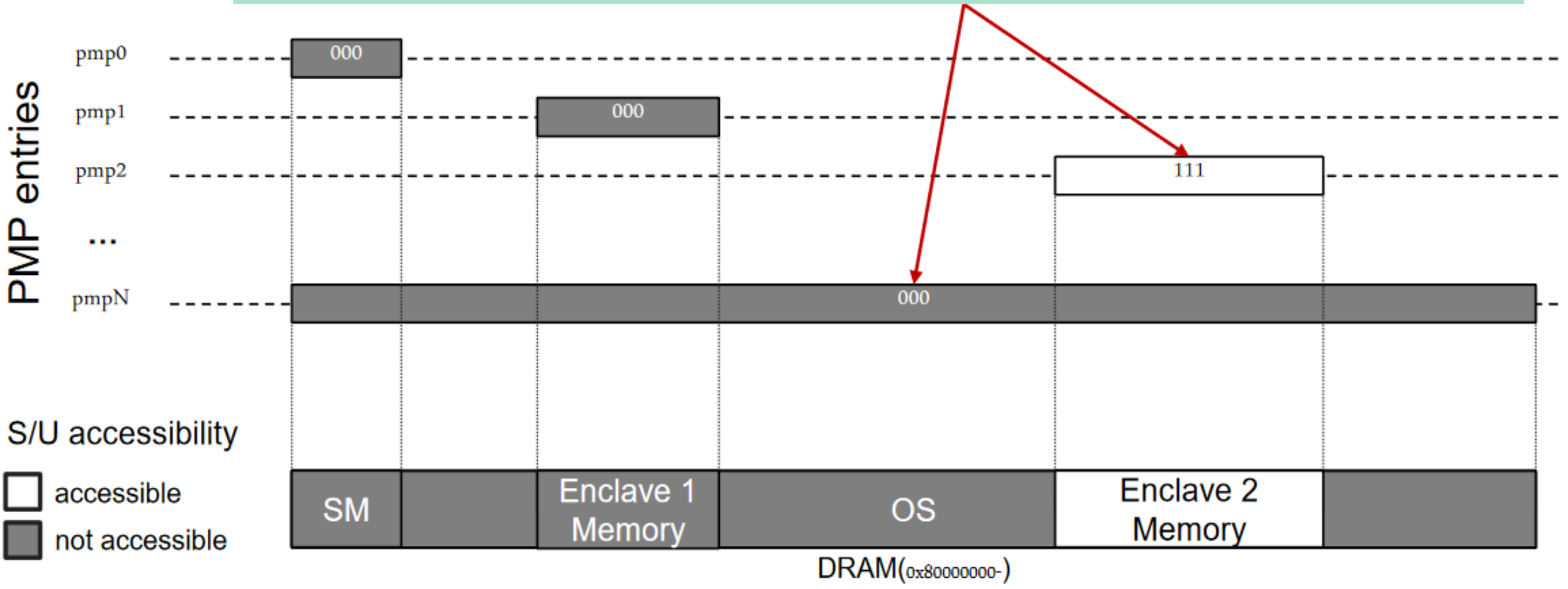


O SO pode alocar memória para um enclave e seu runtime

RISC-V Keystone

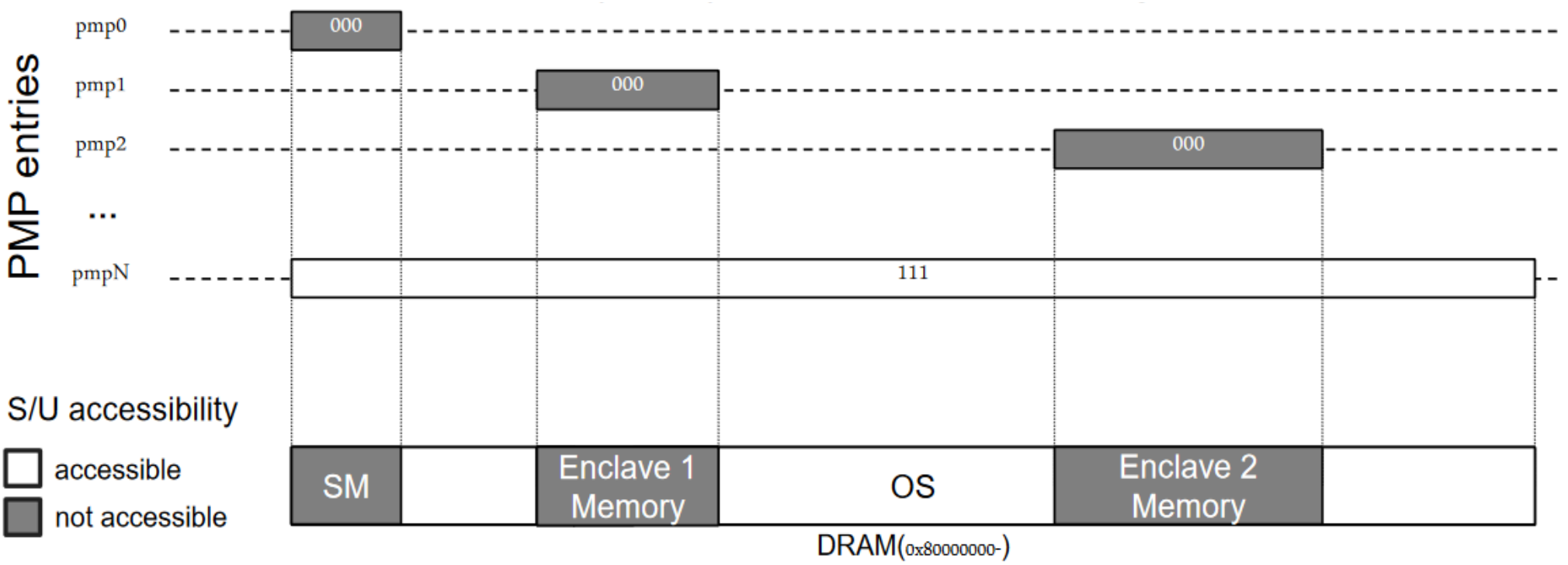
- **Componente PMP – Funcionamento**

Antes de executar o Enclave 2, o SM troca as permissões `pmp2 <-> pmpN`



RISC-V Keystone

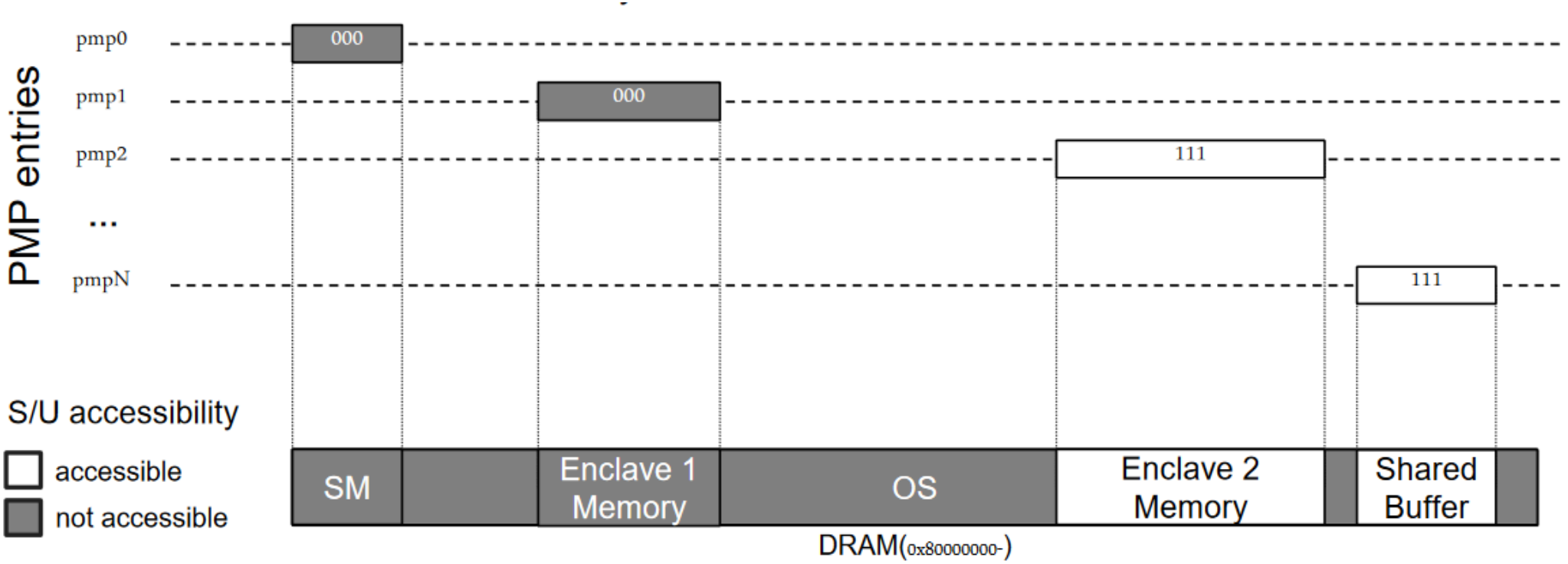
- **Componente PMP – Funcionamento**



Quando o Enclave 2 concluir, o SM troca novamente as permissões `pmp2 <-> pmpN`

RISC-V Keystone

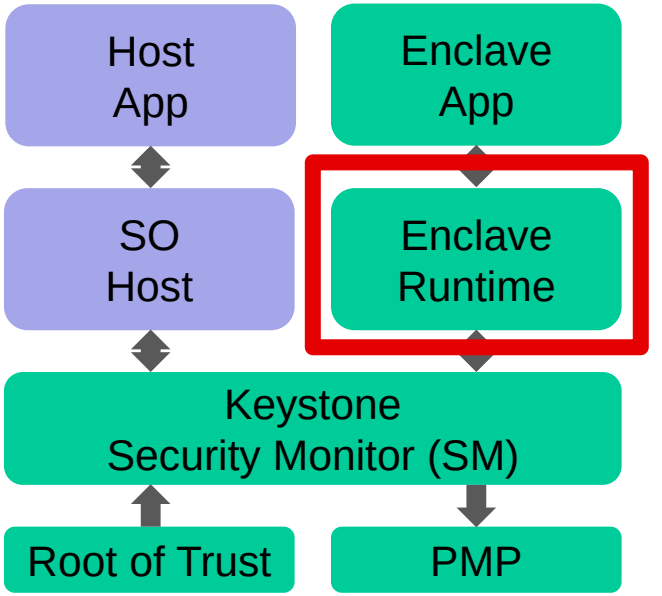
- **Componente PMP – Funcionamento**

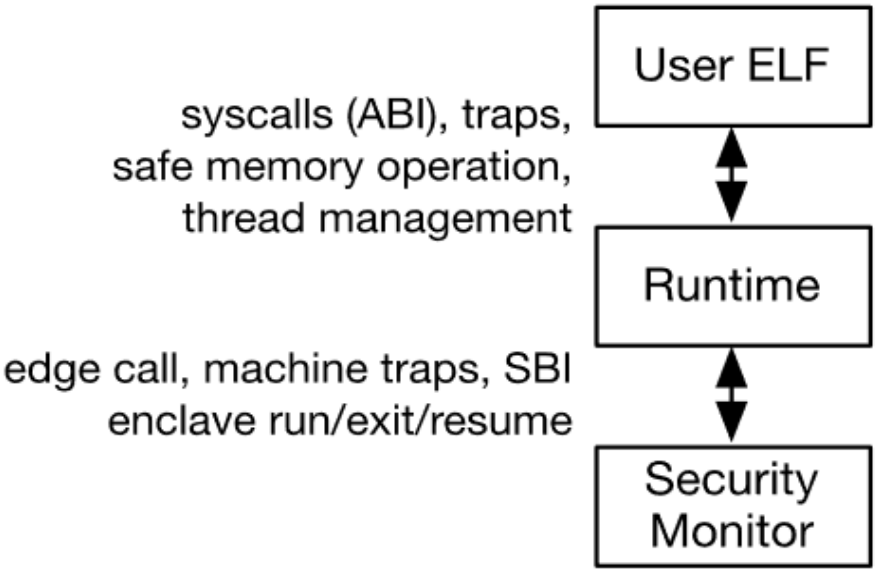


O SO pode criar memória compartilhada com os enclaves.
Para isso, o SM fornece permissões de acesso ao enclave.
Na figura acima, o Enclave 2 possui permissão de r/w no *Shared Buffer*

RISC-V Keystone

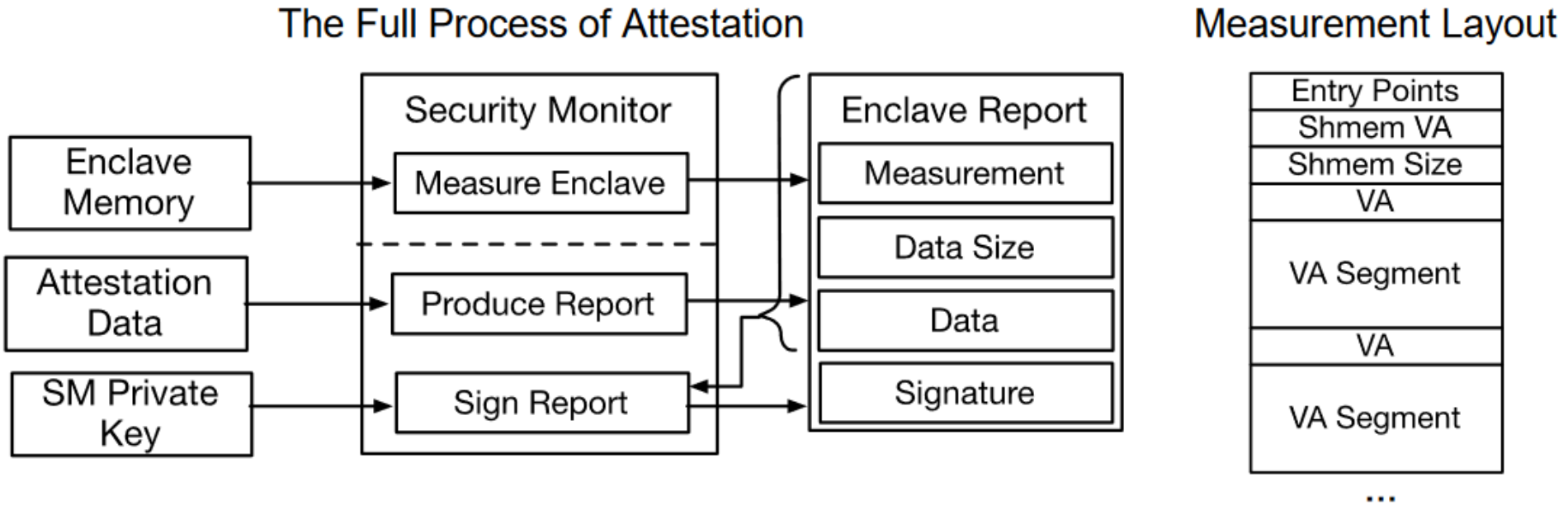
- **S-Mode Enclave Runtime**:
  - Proporciona funcionalidades análogas às do kernel
    - Syscalls, traps (exceções)
    - Gestão de páginas e threads
  - Camada útil de abstração
    - Funcionalidades adicionais sem aumentar a complexidade do SM
    - SM < 2K LoC + 5K LoC crypto lib
  - Executa programas binários – ELF:
    - Que podem ser desenvolvidos “do zero” para Keystone; ou
    - Podem ser utilizados os já existentes (binários para RISC-V)





RISC-V Keystone

- **Remote Attestation:**
  - SM faz *measurement* do enclave no momento da criação
  - O enclave vincula uma chave ao *report* do enclave
  - SM assina o *report* do enclave e envia ela juntamente com o SM *report* ao usuário



RISC-V Keystone

- **Contribuições:**
  - Meio de compartilhar memória (*shared buffer*) entre enclave e outros programas;
  - Controle de permissões de acesso modelo ACL usando “rwx”

Obrigado

https://github.com/larc-sbseg-2024-demo/amb-comp-segura