Capítulo 10

Segurança baseada em hardware

Este capítulo apresenta as principais tecnologias de segurança baseadas em *hardware*, fazendo uma revisão dos diversos mecanismos de segurança que foram ou são utilizados ao longo dos anos para manter a segurança das aplicações e confidencialidade dos dados, com o objetivo de apresentar uma evolução destas técnicas. O texto deste capítulo foi parcialmente baseado no minicurso [Will et al., 2017].

O hardware provê diversos mecanismos para garantir a confidencialidade e integridade, tanto para o código quanto para os dados. Podemos organizar esses mecanismos em diversas categorias:

Mecanismos básicos: são os mecanismos fundamentais de restrição de acesso à memória e a instruções sensíveis providos pela maioria dos processadores, como memória virtual e níveis de privilégio núcleo/usuário.

Hardware criptográfico: hardware dedicado à execução de algoritmos de criptografia e proteção de suas chaves.

Processadores confiáveis: processadores que podem ser usados como "raiz de confiança" para garantir a integridade do sistema e para o armazenamento e manipulação segura de dados, como o TPM.

Ambientes de execução confiável: também chamados TEEs (*Trusted Execution Environments*), são processadores que proveem confidencialidade e integridade do código e dos dados que manipulam, através de cifragem da memória.

Integridade do fluxo de controle: são mecanismos de hardware usados para garantir a integridade da execução de um código, bloqueando ataques de software usuais como *buffer overflow* e *return-oriented programming* (ROP).

Monitoração: são mecanismos de hardware usados para observar a execução e coletar dados que permitam detectar anomalias, como a presença de *malware*.

As próximas seções apresentam com detalhes cada uma dessas categorias.

10.1 Mecanismos básicos

Os ambientes computacionais modernos proveem algumas funcionalidades básicas de segurança, que usualmente são providas pelo hardware e gerenciadas

pelo sistema operacional (SO) como parte do modelo de execução. Exemplos dessas funcionalidades básicas são:

Níveis de privilégio: a separação da execução em modo núcleo e modo usuário, controlada por *flags* do processador, permite restringir o acesso ao hardware e proteger o núcleo do SO de aplicações maliciosas. Enquanto o núcleo do SO executa em modo privilegiado e tem acesso a todas as funcionalidades do hardware, as aplicações executam em modo usuário, com acesso a um subconjunto das instruções, registradores e portas de entrada/saída.

Memória virtual: cada aplicação usa uma ou mais áreas de memória RAM próprias, acessadas por um espaço de endereçamento virtual e gerenciadas pela Unidade de Gerência de Memória (MMU - *Memory Management Unit*), um componente do hardware configurado pelo núcleo. Tentativas de acesso fora dessas áreas são bloqueadas pela MMU e informadas ao processador através de interrupções.

Controle de acesso à memória: a MMU permite configurar permissões de acesso a cada área de memória usada por uma aplicação; por exemplo, uma área contendo código binário pode ter permissões de leitura e execução, enquanto áreas de dados ou de pilha podem ter permissões de leitura e escrita. Um exemplo típico desse controle é o flag NX (*No eXecute*) usado nas tabelas de páginas de processadores Intel, que impede a execução do conteúdo de uma página. Esse flag é frequentemente usado pelo SO para implementar uma política denominada $W \wedge X$, na qual áreas com permissão de escrita não podem ter permissão de execução e vice-versa.

Além dessas funcionalidades básicas presentes na maioria dos processadores (com exceção dos microcontroladores mais simples usados em sistemas embarcados), outras funcionalidades podem ser frequentemente encontradas, como:

IOMMU: em sistemas com acesso direto à memória (DMA - *Direct Memory Access*), um dispositivo externo malicioso ou defeituoso pode acessar áreas da RAM indevidamente durante operações de DMA. De forma similar à MMU, uma IOMMU (*Input/Output Memory Management Unit*) permite criar um espaço de endereçamento virtual distinto para cada dispositivo de entrada/saída, bloqueando esses acessos indevidos.

Virtualização por hardware: a virtualização permite executar um sistema operacional "convidado" (*guest*) e suas aplicações sobre um sistema operacional "hospedeiro" (*host*) de forma transparente e isolada. Assim, um SO convidado acredita estar executando sobre um hardware exclusivo dele e não consegue acessar recursos do hospedeiro ou de outros SOs convidados na mesma plataforma. A virtualização é gerenciada por um hipervisor (ou monitor de virtualização) e implementada usando mecanismos de hardware como Intel VT-x, AMD-V ou ARM-VHE.

Nas próximas seções serão discutidos mecanismos de hardware mais específicos, implementados em hardware separado ou embutidos em processadores mais avançados.

10.2 Hardware Criptográfico

Operações criptográficas são naturalmente custosas, portanto a utilização de um *hardware* dedicado para esta tarefa provê um ganho de desempenho. Processadores deste tipo são utilizados há muito tempo em aplicações específicas, como sistemas bancários e aplicações militares. No entanto, com a crescente necessidade da proteção de dados sensíveis em diversas aplicações, cripto-processadores têm ganho maior relevância e diferentes implementações têm sido empregadas. Dentre estas implementações, [Bossuet et al., 2013] destaca os seguintes mecanismos:

Extensão criptográfica: a maioria dos processadores de uso geral em computadores pessoais e servidores dispõe de extensões criptográficas que implementam em hardware as operações de criptografia mais usuais. Como exemplos podem ser citadas as extensões Intel AES-NI e VAES (algoritmo AES), a instrução RDRAND (geração de números realmente aleatórios) e as instruções SHA (que implementam os algoritmos SHA) nos processadores ARM, RISC-V e Intel. Estas extensões melhoram o desempenho das operações criptográficas, mas não aumentam a segurança do sistema, uma vez que as chaves criptográficas são armazenados em memória e tratadas como dados ordinários da aplicação, podendo sofrer ataques por software.

Cripto-coprocessador (ou acelerador criptográfico): Implementação em *hardware* customizado, altamente eficiente para funções criptográficas específicas. Eles podem conter um ou mais núcleos de processamento, mas não são programáveis e sim controlados pelo processador ao qual estão vinculados. Também não realizam armazenamento seguro de chaves ou dados. Seu uso é indicado para aplicações com necessidade de alto desempenho em criptografia. Pode ser citado como exemplo o chip Samsung S3SSE2A, que tem suporte à criptografia pós-quântica.

Cripto array: É um acelerador criptográfico que, assim como os cripto-processadores, precisa ser acoplado a processadores de propósito geral. Esta arquitetura usa núcleos de processamento vetorial, como GPUs (*Graphic Processing Units*) para realizar operações criptográficas com alto desempenho.

Cripto-processador: É um módulo de hardware que se difere do coprocessador por ser autônomo e programável, com memória própria isolada da memória principal e um conjunto de instruções dedicadas para obter eficiência em funções criptográficas. Cripto-processadores contam com uma ou mais unidades lógico-aritméticas (ULAs) projetadas especificamente para cálculos criptográficos; por isso, para realizar operações convencionais é necessário o uso de um processador de propósito geral. Esta arquitetura é mais enxuta que as duas anteriores, podendo ser considerada relativamente flexível, uma vez que as ULAs podem ser reconfiguradas. Outra grande diferença é que as chaves de criptografia são armazenadas internamente no cripto-processador e não mais na memória principal do sistema.

10.3 Processadores confiáveis

Dois requisitos fundamentais para o estabelecimento de um ambiente computacional seguro são: a) garantir que a plataforma pode provar que é quem ela diz ser (autenticidade); e b) garantir que a plataforma não foi adulterada (integridade) [TCG, 2026]. Alguns conceitos são necessários para estabelecer uma plataforma de computação confiável:

Mensuração: Consiste em construir um relatório descrevendo um componente de hardware ou de software, incluindo identificação, fabricante, versão, configurações e um resumo (*hash*) de seu conteúdo, de modo a permitir a verificação do mesmo posteriormente. Uma mensuração deve ser assinada digitalmente, para garantir sua autenticidade e integridade.

Atestação: consiste em usar mensurações para verificar a autenticidade de um componente do sistema (hardware, firmware ou software), ou seja, assegurar que ele é o código previsto pelo seu fabricante ou criador e não foi indevidamente substituído ou adulterado. A atestação pode ser local, quando um componente atesta outro na mesma plataforma, ou remota, quando o componente é verificado remotamente. A atestação remota permite, por exemplo, ao usuário de um serviço na nuvem verificar se o código do servidor é autêntico.

Raiz de confiança: para assegurar a confiabilidade de uma mensuração, ela deve ter sido gerada e assinada por um componente confiável, cuja autenticidade e integridade possam ser verificadas. O componente confiável inicial de uma plataforma é denominado *raiz de confiança* (RoT - *Root of Trust*) e geralmente é provido pelo fabricante em um módulo de hardware imutável, podendo ser consultado e executado, mas não alterado.

Cadeia de confiança: a raiz de confiança pode atestar outros componentes do sistema usando suas mensurações previamente armazenadas e assinadas. Isso permite construir uma "cadeia de confiança" (*Chain of Trust*) entre componentes de hardware ou de software do sistema, de forma similar a uma cadeia de certificados de chave pública na Internet.

A Figura 10.1 ilustra esses conceitos. Um processador confiável é um hardware especialmente construído para prover uma raiz de confiança e suportar a atestação dos componentes de um sistema computacional. O processador confiável mais difundido é o *Módulo de Plataforma Confiável*, apresentado a seguir.

10.3.1 Módulo de Plataforma Confiável

O TPM (*Trusted Platform Module*) um *hardware* dedicado capaz de armazenar e processar informações usadas para verificar a autenticidade e integridade de uma plataforma computacional, o que inclui chaves e certificados criptográficos [TCG, 2026]. Além de computadores pessoais e servidores, TPMs são frequentemente encontrados em outros equipamentos, como *smartphones*, dispositivos de rede e centrais eletrônicas automotivas.

Em sua forma mais simples, o TPM é construído como um único chip acoplado ao sistema (tipicamente um PC), composto por processador, memória volátil (RAM),

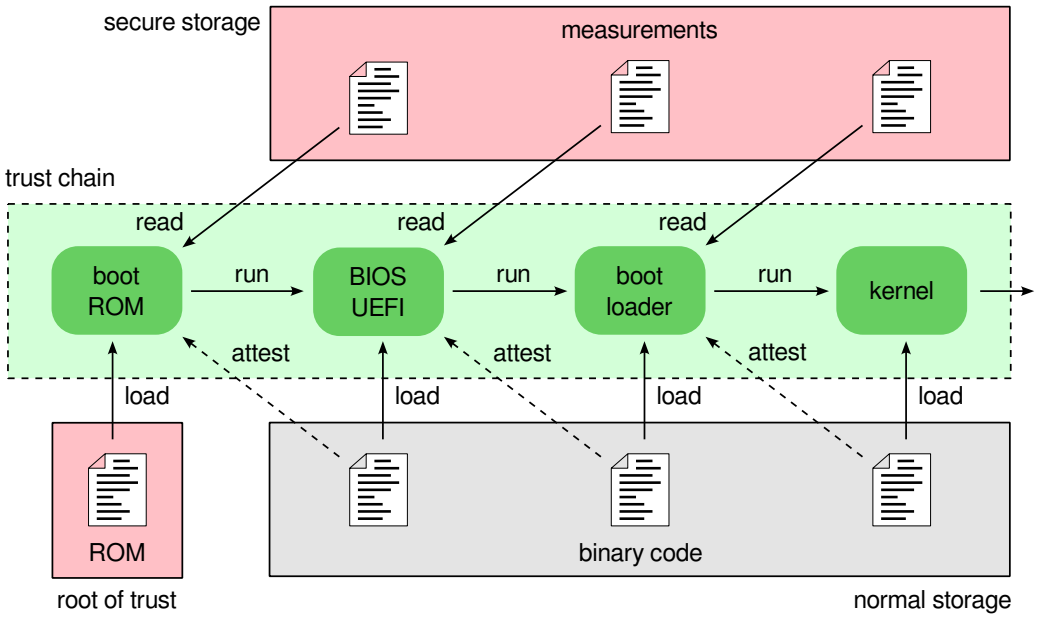

Figura 10.1: Construção da cadeia de confiança entre componentes na inicialização de um sistema computacional.

memória não-volátil (ROM e *flash*) e uma interface de barramento. O processador principal da plataforma não pode acessar diretamente o conteúdo do TPM; toda interação com o TPM deve ser feita através de comandos e dados enviados/recebidos pela interface que o conecta ao restante do sistema. A Figura 10.2 mostra a arquitetura básica de um TPM.

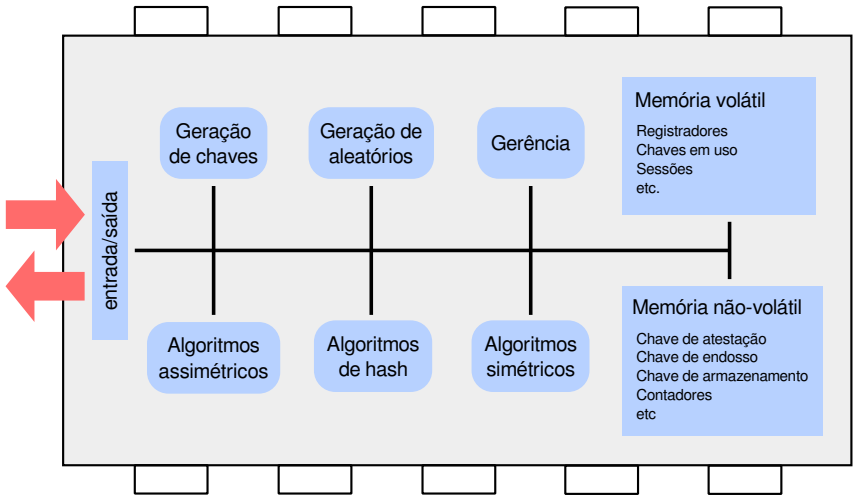

Figura 10.2: Arquitetura de um TPM (adaptado de [TCG, 2026]).

O código interno do TPM contém implementações de algoritmos de criptografia simétricos e assimétricos, resumos criptográficos, assinaturas digitais, um gerador de aleatórios e geradores de chaves criptográficas. Em sua versão inicial (v1.1, 2003), o TPM suportava os algoritmos RSA e SHA-1. Na versão atual (v2.0, 2014) foram acrescentados os algoritmos ECC, AES e SHA-256, além de outras funcionalidades e mais possibilidades de configuração.

Cada TPM contém um par de chaves de criptografia assimétrica, definido durante a fabricação do chip e denominado *Chaves de Endosso* (EK - *Endorsement Keys*). O TPM fornece um certificado da chave pública de endosso através da interface, assinado pelo fabricante do chip, mas a chave privada de endosso só é usada internamente e nunca sai dele. Essa chave de endosso identifica unicamente cada chip e é usada para derivar (produzir) outras chaves pelo TPM. Além da EK, o TPM possui outras chaves primárias, como a *Chave Raiz de Armazenamento* (SRK - *Storage Root Key*), que serve para selar (cifrar) conteúdos que serão armazenados fora do TPM (na memória RAM ou em disco).

Além de prover funcionalidades de criptografia e de números aleatórios, o TPM pode ser usado para armazenar de forma segura *hashes* e chaves de criptografia para o *firmware*, o sistema operacional e as aplicações, evitando que essas informações fiquem expostas na memória RAM do computador.

Um uso importante do TPM consiste em validar a integridade do software da plataforma computacional, através da *inicialização segura* (*secure boot*). Nesse processo, um resumo criptográfico (*hash*) de cada componente do software de inicialização da plataforma (BIOS/UEFI, *bootloader*, *drivers* e núcleo do SO) é previamente calculado e armazenado no TPM. Durante a inicialização, cada componente de software recalcula o *hash* do código do próximo componente e só transfere o controle para ele se esse *hash* for igual ao armazenado no TPM. Isso garante que esses componentes críticos da inicialização não foram alterados de forma indevida.

Existem várias implementações possíveis para o TPM [TCG, 2026]:

TPM discreto: chip dedicado e separado do processador principal da plataforma. É a versão mais segura, devido ao seu isolamento do restante do sistema. O chip Infineon Optiga SLB 9670 é um exemplo desta abordagem.

TPM integrado: é um TPM implementado por hardware, mas fisicamente dentro de outro chip, como um processador ou controlador. Um bom exemplo dessa abordagem é a tecnologia Intel *Platform Trust Technology* (PTT), presente nos processadores da linha *Coffee Lake*.

TPM em firmware: a funcionalidade de TPM é implementada em software no *firmware* da plataforma (BIOS ou UEFI), usando outros mecanismos de segurança do processador. Está disponível em várias plataformas Intel e AMD.

TPM virtual: disponível em ambientes de virtualização. O TPM é construído em software e sua segurança é garantida pelo isolamento provido pelo hipervisor, cujo estado interno é inacessível às máquinas virtuais. Ambientes de virtualização como VMWare, VirtualBox e KVM oferecem esta funcionalidade.

TPM emulado: implementado em software, executa junto com as demais aplicações. Não provê nenhuma segurança adicional, mas pode ser útil em ambientes de desenvolvimento.

10.3.2 AMD *Platform Security Processor*

O *Platform Security Processor* (PSP) da empresa AMD é um subsistema de segurança em *hardware* dedicado integrado ao chip do processador, que executa de modo independente dos núcleos do processador principal. Esse subsistema proporciona

um ambiente para armazenamento de dados sensíveis e define uma raiz de confiança para a inicialização segura (*secure boot*), de forma similar ao TPM (Seção 10.3.1). Para tal, o PSP faz uso de um microcontrolador ARM *TrustZone* de 32 *bits* independente e com memória própria, mas fisicamente integrado ao processador principal [Coppolino et al., 2019].

O PSP contém um coprocessador criptográfico composto de um gerador de números aleatórios, mecanismos para processamento de algoritmos criptográficos (AES, RSA e outros) e um bloco para armazenamento de chaves. Esse bloco é composto de duas áreas de armazenamento: uma usada por *software* privilegiado, cuja leitura não é possível, e outra onde chaves podem ser carregadas, utilizadas e descartadas, em operações convencionais de *software* executando no PSP ou no sistema operacional convencional. Também há uma lógica implementada em *hardware* para inicialização segura do núcleo da CPU e uma chave única da plataforma, que é distribuída para o bloco de armazenamento do coprocessador criptográfico durante o processo de inicialização.

Para garantir uma inicialização confiável, o PSP implementa o *Hardware Validated Boot* (HVB), que é uma forma de inicialização segura não-modificável, implementada na ROM do chip, que verifica a integridade da BIOS. A ROM faz a validação de uma chave de inicialização e então usa essa chave para validar o *firmware* do PSP que, por sua vez, é carregado e inicia a execução da aplicação do sistema.

10.3.3 Intel *Trusted Execution Technology*

De forma análoga ao sistema PSP da AMD, a tecnologia Intel TXT (*Trusted Execution Technology*) [Greene, 2013] provê funcionalidades para definir uma plataforma segura através da definição de uma raiz de confiança, mensuração e atestação local dos componentes do sistema. Essa tecnologia depende fortemente do *Intel Converged Security and Management Engine* (CSME), subsistema presente no *chipset* dos processadores da Intel desde 2008 [Gehler et al., 2022]. O subsistema CSME usa um processador independente embutido dentro do processador principal que executa seu próprio sistema operacional (Minix 3). O processador CSME implementa diversas funcionalidades, como um TPM em *firmware* com armazenamento seguro e funções criptográficas para suporte à inicialização segura.

Além das funcionalidades de segurança, o Intel CSME implementa vários aspectos de gerência interna do processador principal. Além disso, ele tem acesso à interface de rede e possui seu próprio endereço MAC, o que permite a ele se comunicar com outros sistemas sem o conhecimento do sistema operacional principal da plataforma.

10.3.4 *Hardware Security Module*

O Módulo de Segurança em Hardware (HSM - *Hardware Security Module*) é um dispositivo dedicado ao armazenamento e manipulação de informações críticas, como chaves de criptografia. Usualmente também realiza operações criptográficas, como cifragem/decifragem e assinatura. O hardware deve ser fisicamente separado do sistema computacional, ter processador e armazenamento próprios e ser resistente a violação física (arrombamento) e lógica (monitoração de interfaces, emanações eletromagnéticas, etc.).

Módulos de hardware confiável são frequentemente usados em sistemas bancários, militares e na proteção das chaves em autoridades certificadoras. As carteiras físicas de criptomoedas podem ser consideradas HSMs. A Figura 10.3 mostra alguns módulos de segurança em hardware típicos.

Figura 10.3: Módulos de segurança em hardware (HSM). Da esquerda para a direita: módulo HSM autônomo, placa HSM e carteira física de criptomoedas.

10.4 Ambientes de Execução Confiável

Um ambiente de execução confiável (do inglês TEE - *Trusted Execution Environment*) é um espaço reservado dentro de um sistema computacional que provê as propriedades básicas de segurança (confidencialidade, integridade e autenticidade) às aplicações ou trechos de código que executam dentro do mesmo. Por suas características de isolamento, esse ambiente de execução segura é usualmente denominado *enclave*. Algumas arquiteturas suportam múltiplos enclaves isolados entre si.

O código e os dados contidos dentro de um enclave são protegidos pelo hardware de acessos ou alterações indevidas vindas de código não-confiável, mesmo se esse código estiver executando como administrador, no núcleo do SO, em *drivers* maliciosos ou em outros enclaves. Dessa forma, enclaves são usados para proteger código e dados sensíveis, como funções de autenticação, senhas, dados biométricos e chaves de criptografia.

Uma aplicação pode ser construída de forma a separar suas partes confiáveis (que manipulam informação crítica) e não-confiáveis (o restante), sendo as partes confiáveis executadas dentro de enclaves. O código dentro de um enclave pode se comunicar com o código externo através de mecanismos restritos providos pelo hardware.

Ambientes de execução confiáveis são usados em muitos sistemas, como telefones celulares, consoles de jogos e TVs inteligentes, além de computadores pessoais e servidores. Existem diferentes arquiteturas para a construção de ambientes de execução confiáveis, sendo mais difundidas as arquiteturas ARM *TrustZone*, muito usada em

telefones celulares iPhone e Android, e Intel SGX/TDX, presente em servidores de nuvem. Essas arquiteturas são apresentadas brevemente nas próximas seções.

10.4.1 ARM *TrustZone*

TrustZone é uma tecnologia de segurança baseada em hardware disponível nos processadores ARM Cortex-A, Cortex-M23 e Cortex-M33, muito usados em dispositivos móveis, como *smartphones*. Ela define dois domínios de execução isolados entre si: um *domínio normal* ou inseguro (*normal world*), que abriga as aplicações normais (inseguras), e um *domínio seguro* (*secure world*), que abriga aplicações e bibliotecas críticas para a segurança do sistema.

Além do software, áreas de memória e recursos como dispositivos de E/S e registradores da CPU são associados a cada um dos domínios pelo hardware: o controlador de espaço de endereçamento (TZASC - *TrustZone Address Space Controller*) define as áreas normais ou seguras da memória RAM e o controlador de proteção (TZPC - *TrustZone Protection Controller*) regula o acesso aos dispositivos periféricos e gerencia suas interrupções. É importante ressaltar que no *TrustZone* o conteúdo da memória do domínio seguro não é criptografado.

Por default, o código executando no domínio seguro pode acessar a memória e os recursos do domínio normal, mas o contrário não é permitido. Aplicações no domínio normal somente podem acessar funcionalidades e serviços providos pelo domínio seguro através de uma API específica (*GlobalPLatform TEE API*) [Markantonakis and Mayes, 2003].

O código em cada domínio é dividido entre os níveis de usuário (aplicações) e de núcleo (sistema operacional). Portanto, cada domínio possui seu próprio sistema operacional e suas próprias aplicações. Além disso, um modo *monitor* ainda mais privilegiado é usado para abrigar o código que gerencia a transição entre os domínios normal e seguro. A Figura 10.4 apresenta uma visão geral da arquitetura ARM *TrustZone*:

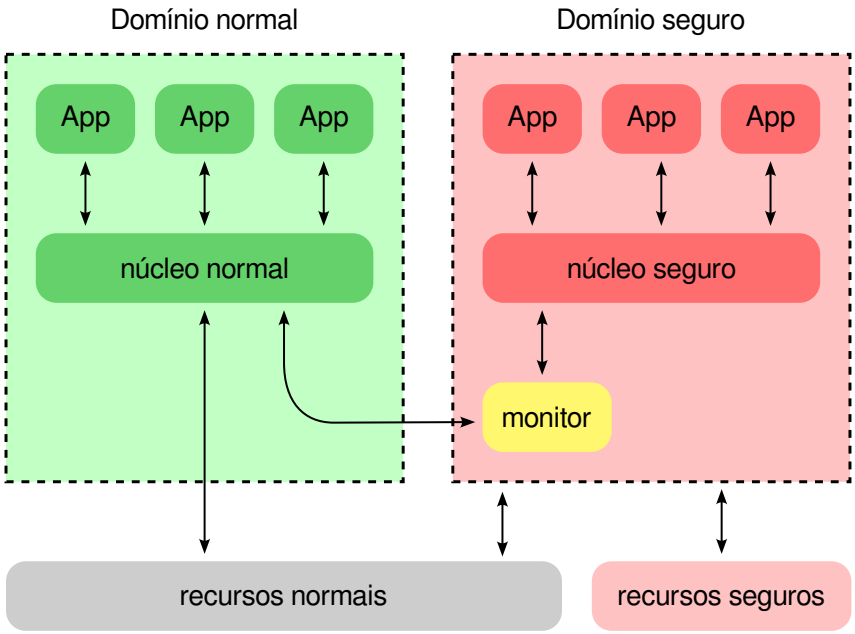

Figura 10.4: Visão geral da arquitetura ARM *TrustZone* (adaptado de [Coppolino et al., 2019]).

Um sistema computacional usando *TrustZone* deve portanto particionar suas aplicações, bibliotecas e periféricos entre os domínios normal e seguro. Cada domínio deve ter seu próprio sistema operacional e sua própria inicialização. Plataformas comerciais geralmente usam pequenos sistemas operacionais proprietários no domínio seguro, como OP-TEE (vários fabricantes), QSEE (Qualcomm) e Knox (Samsung).

Os processadores usados em dispositivos *Apple/iOS* dispõem de um hardware de segurança similar ao *Trustzone*, chamado *Apple Secure Enclave* (ASE) [Apple Inc, 2026]. Ele é provido de um coprocessador responsável pelo domínio seguro, executando uma versão customizada do núcleo de sistema operacional L4. Ao contrário do *TrustZone*, no ASE o conteúdo da memória do domínio seguro é criptografado.

10.4.2 Intel *Software Guard Extensions*

A tecnologia *Software Guard Extensions* (SGX) [Will and Maziero, 2023] é uma extensão dos processadores Intel que cria áreas de memória RAM protegidas por criptografia, denominadas *enclaves*. Ao contrário do *TrustZone* (Seção 10.4.1), SGX permite criar diversos enclaves independentes.

Com SGX, uma extensão da memória RAM denominada *Enclave Page Cache* (EPC) é reservada durante a inicialização para abrigar o código e dados de cada enclave. Essa área é criptografada para proteger seu conteúdo de outros programas executando na mesma plataforma, mesmo aqueles com privilégios elevados (*root*, núcleo, *drivers* ou hipervisor). Além disso, a EPC mantém *hashes* de verificação para detectar alterações indevidas na memória de cada enclave [Costan and Devadas, 2016].

As páginas de memória da EPC transitam criptografadas nos barramentos e caches, sendo somente decifradas dentro da CPU; toda escrita na memória RAM do enclave é novamente cifrada antes de deixar a CPU. A chave de criptografia é armazenada em registradores dentro da CPU, não sendo acessível a componentes externos à mesma, e é trocada aleatoriamente a cada evento de hibernação ou reinício do sistema. A Figura 10.5 traz uma visão geral do SGX.

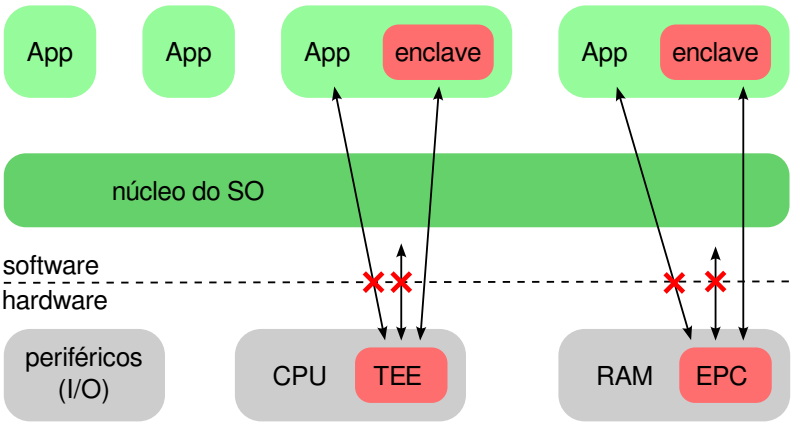

Figura 10.5: Visão geral do ambiente SGX (adaptado de [Coppolino et al., 2019]).

Cada enclave SGX está mapeado no espaço de memória de uma aplicação e pode ser usado para proteger os dados e operações sensíveis da mesma. Uma aplicação pode conter diversos enclaves. A interação entre a parte insegura da aplicação (fora do enclave) com a parte segura (dentro do enclave) é feita através de chamadas de funções predefinidas denominadas *ECalls* (*Enclave Calls*) e *OCalls* (*Outside Calls*), como mostra

a figura 10.6. As *ECalls* permitem à aplicação solicitar serviços seguros ao enclave, enquanto as *OCalls* permitem ao código do enclave realizar chamadas de sistema para o núcleo (*syscalls*), que são proibidas dentro do mesmo.

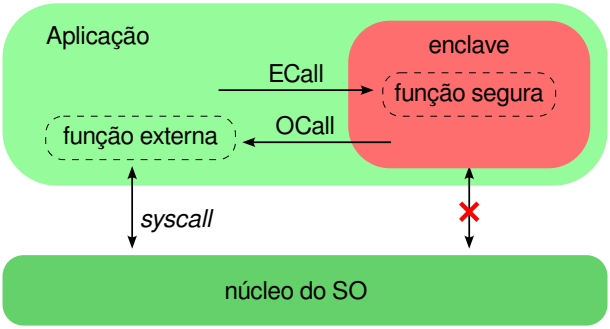

Figura 10.6: Chamadas entre aplicação e enclave no SGX.

A extensão SGX provê registradores especiais e 18 novas instruções ao processador para criar e gerenciar os enclaves, divididas entre instruções para código privilegiado (núcleo ou hipervisor) e para código de usuário (aplicações). Algumas instruções relevantes são:

- `ECREATE`: cria a estrutura de controle de um enclave.
- `EINIT`: inicializa um enclave.
- `EENTER` ou `ERESUME`: entra ou retorna ao enclave.
- `EEXIT`: sai do enclave.
- `AEX`: saída assíncrona do enclave (por interrupção ou exceção).
- `EREPORT`: gera uma mensuração do enclave, usada nos procedimentos de atestação local e remota (vide abaixo).
- `EREMOVE`: destrói o enclave.

Além da proteção provida pela memória criptografada, o SGX oferece outras funcionalidades importantes para a segurança do sistema, como:

Mensuração: o conteúdo de cada enclave (código e dados iniciais) é mensurado através de resumos criptográficos assinados, de forma a permitir identificá-lo e verificar sua integridade.

Atestação: Usando as mensurações, a instanciação correta e integridade do conteúdo de um enclave podem ser verificadas por programas externos ao mesmo, local ou remotamente, sem revelar o conteúdo do enclave. A atestação remota de um enclave exige a participação de uma entidade externa confiável (autoridade certificadora), como a *Intel Trust Authority*.

Selagem: cada enclave mantém chaves de criptografia para "selar" (criptografar) dados sensíveis do enclave em dispositivos de armazenamento externos, como discos. Os dados selados só podem ser acessados a partir do enclave que os gerou e só são decifrados dentro da CPU.

A plataforma SGX, lançada em 2015, trouxe diversas inovações na segurança baseada em hardware e foi muito explorada na pesquisa acadêmica [Will and Maziero, 2023]. Entretanto, seu uso pelos desenvolvedores mostrou-se complexo, dificultando sua adoção pelo mercado. Uma grande barreira é a necessidade de reescrever ou alterar as aplicações para separar suas áreas seguras e inseguras, a fim de executar as áreas seguras dentro de enclaves.

10.4.3 Intel *Trusted Domain Extensions*

Como o uso da tecnologia SGX se mostrou complexo, e observando a crescente adoção de ambientes virtualizados na nuvem para a construção de aplicações, em 2023 a Intel lançou a tecnologia *Trusted Domain Extensions* (TDX) [Cheng et al., 2024], suportada pelos processadores da linha *Xeon* mais recentes. Essa tecnologia visa construir ambientes de execução seguros para máquinas virtuais e contêineres.

O objetivo do TDX é executar permitir a execução de cada máquina virtual em um domínio confiável (TD - *Trusted Domain*), Uma máquina virtual (VM) em um domínio confiável tem sua memória criptografada e está protegida de outras VMs, do hipervisor, do sistema operacional e do *firmware* do servidor em termos de confidencialidade e integridade. Assim, o software executando dentro de um domínio confiável na nuvem não pode ser observado nem modificado pelo proprietário da nuvem.

A arquitetura TDX faz uso de tecnologias anteriores da Intel, como SGX e VMX (virtualização por hardware). Ela introduz um novo modo de operação da CPU chamado SEAM (*SEecure-Arbitration Mode*), com mais privilégios que o modo núcleo, e instruções para entrar e sair desse modo de forma controlada (`SEAMCALL`/`SEAMRET`). Um módulo de software provido e atestado pelo fabricante (*Intel TDX module*) executa no modo SEAM, sendo responsável por gerenciar os domínios confiáveis. A figura 10.7 traz uma visão geral simplificada da arquitetura TDX.

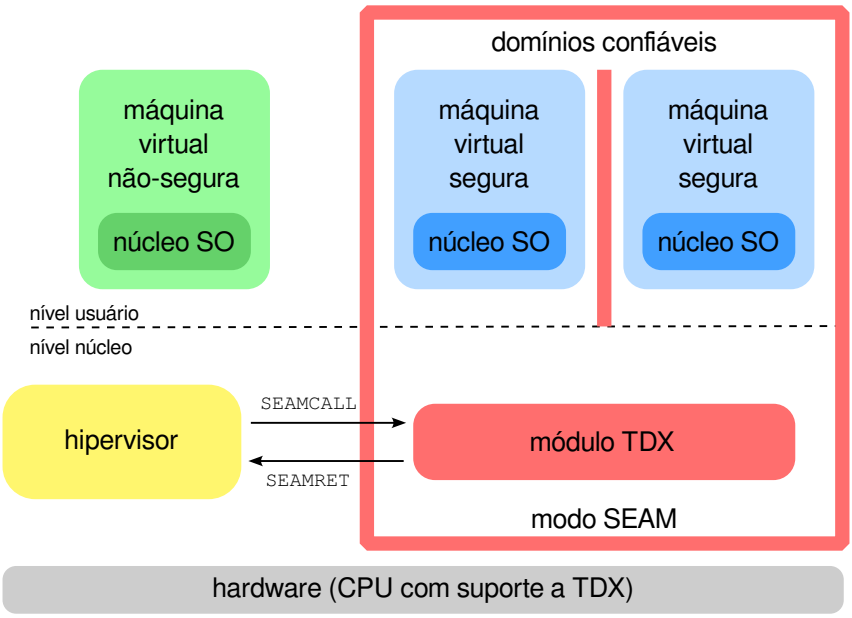

Figura 10.7: A arquitetura Intel TDX (adaptado de [Cheng et al., 2024]).

TDX também suporta os mecanismos de cadeia de confiança, selagem de dados e atestação remota. Uma grande vantagem da arquitetura TDX sobre sua antecessora

SGX consiste no suporte a aplicações sem necessidade de reescrita do código das mesmas. Somente o hipervisor de virtualização e os sistemas operacionais das VMs seguras precisam ter o suporte adequado a essa tecnologia.

Além da Intel, diversos fabricantes de hardware proveem soluções de isolamento de máquinas virtuais similares a TDX, como *AMD Secure Encrypted Virtualization* (SEV), *ARM Confidential Compute Architecture* (CCA), *IBM Secure Execution*, *IBM Protected Execution Facility* (PEF) e *RISC-V Confidential VM Extension* (CoVE).

10.5 Integridade do fluxo de controle

Além do suporte a chaves e operações criptográficas, memória segura e isolamento de código apresentadas nas seções anteriores, existem mecanismos de hardware que se propõem a garantir a integridade da execução. Isso implica em garantir que o fluxo de execução de uma aplicação obedeça ao previsto no código, impedindo ataques de desvio da execução como *buffer/stack overflow* e *return-oriented programming*, além de acesso a regiões de memória do processo de forma imprevista. Os mecanismos de hardware descritos a seguir são considerados os mais relevantes [Coppolino et al., 2019; Li and Gao, 2025]:

- *Intel Control-flow Enforcement Technology* (CET): provê dois mecanismos para proteger o fluxo de execução da aplicação: *Shadow Stack* e *Indirect-Branch Tracking*.
  - O mecanismo *Shadow Stack* (SS) cria uma pilha secundária da aplicação para armazenar os endereços de retorno das chamadas de função: a cada chamada de função, uma cópia do endereço de retorno é empilhada nessa pilha "sombra". Ao encerrar a função, o endereço de retorno contido na pilha principal é comparado com o endereço contido na pilha "sombra", permitindo detectar ataques de estouro de pilha (*stack overflow*).
  - O mecanismo *Indirect-Branch Tracking* (IBT) introduz uma nova instrução chamada `ENDBRANCH` para marcar destinos válidos para saltos de execução (operações *jmp*). Caso o destino de um salto não seja uma instrução `ENDBRANCH`, uma interrupção é gerada para sinalizar uma violação do fluxo de execução. Essa técnica permite detectar ataques de injeção e de reuso de código, como ROP (*Return-Oriented Programming*). A tecnologia *ARM Branch Target Identification* (BTI) funciona de forma similar à Intel IBT. O exemplo a seguir mostra um código em C e o código *assembly* correspondente usando instruções Intel `ENDBRANCH`:

```c
void func() {}

int main()
{
    void (*f)();

    f = func;
    f(); // chamada indireta
}
```

```
main:
    endbr64
    movl $func, %edx
    call *%edx
    ret

func:
    endbr64
    ret
```

- *ARM Pointer Authentication* (PA): associa a cada ponteiro (variável do programa contendo um endereço de memória) uma assinatura que permite detectar se o mesmo foi alterado de forma indevida. Essa assinatura é calculada a partir do valor do ponteiro, de uma chave secreta mantida no processo e de um contexto, que pode ser o valor do ponteiro da pilha. A assinatura é mantida no próprio ponteiro, usando seus bits mais significativos (em sistemas ARM de 64 bits, apenas os 40 bits menos significativos dos ponteiros são usados). Instruções adicionais são definidas para gerar e testar as assinaturas.

- *ARM Memory Tagging Extension* (MTE): associa a cada área de memória e a cada ponteiro uma etiqueta (*tag*) de 4 bits. O acesso de uma área de memória por um ponteiro só é permitido pelo hardware se as etiquetas associadas a ambos forem iguais.

- *Intel Memory Protection Keys* (MPK): permite associar a cada página da memória de um processo um identificador de 4 bits (uma "chave"). Um registrador especial de 32 bits define as permissões de acesso para cada chave: escrita (1 bit) e leitura/execução (1 bit). Essas permissões são verificadas pelo hardware ao acessar cada página. Ao contrário da MMU convencional, o registrador de permissões é separado para cada *thread* e diretamente acessível no espaço de usuário. Dessa forma, cada *thread* pode definir proteções específicas para as áreas de memória que utiliza.

É importante observar que a maioria dos mecanismos de hardware apresentados nesta seção adicionam novas instruções e registradores ao processador para permitir seu uso efetivo. Em consequência, as aplicações precisam ser recompiladas com compiladores, ligadores e bibliotecas que tenham o suporte adequado às novas instruções para poder usufruir desses mecanismos.

10.6 Monitoração

Diversos mecanismos de hardware foram definidos para monitorar uma execução. Os dados extraídos dessa observação podem ser usados para detectar anomalias que podem indicar ataques ou a presença de *malware*. Os mecanismos de hardware para monitoração da execução mais conhecidos são:

- *Hardware Performance Counters* (HPC): a maioria dos processadores modernos dispõe de registradores especiais para a contagem de eventos internos como instruções executadas, acessos à memória, acertos/erros de cache, erros de predição de saltos, etc. Esses contadores podem ser usados para capturar o comportamento de cada aplicação e detectar anomalias.

- *Intel Processor Trace* (PT): permite capturar eventos do fluxo de execução de um programa (saltos, sequências de instruções, interrupções, etc) de forma eficiente, com baixo impacto no desempenho. O registro da execução é feito com compressão em uma área de RAM separada, para posterior análise. Esta tecnologia é uma evolução das tecnologias anteriores *Last Branch Record* (LBR) e *Branch Trace Store* (BTS).

- *Intel Threat Detection Technology* (TDT): usa o processador gráfico (GPU) da plataforma para fazer a varredura da memória RAM em busca de código malicioso. O hardware incorpora algoritmos de aprendizado de máquina para detectar anomalias que possa indicar a presença de *ransomware*, *spyware* e outros. Alguns produtos antivírus comerciais têm suporte a essa tecnologia [Intel Corp, 2022].

Referências

Apple Inc. Apple platform security. Technical white paper, Apple Inc., August 2026. URL https://help.apple.com/pdf/security/en_US/apple-platform-security-guide.pdf.

L. Bossuet, M. Grand, L. Gaspar, V. Fischer, and G. Gogniat. Architectures of flexible symmetric key crypto engines—a survey: From hardware coprocessor to multi-crypto-processor system on chip. *ACM Computing Surveys*, 45(4), Aug. 2013. ISSN 0360-0300.

P.-C. Cheng, W. Ozga, E. Valdez, S. Ahmed, Z. Gu, H. Jamjoom, H. Franke, and J. Bottomley. Intel TDX demystified: A top-down approach. *ACM Computing Surveys*, 56(9), Apr. 2024.

L. Coppolino, S. D'Antonio, G. Mazzeo, and L. Romano. A comprehensive survey of hardware-assisted security: From the edge to the cloud. *Internet of Things*, 6:100055, 2019. ISSN 2542-6605.

V. Costan and S. Devadas. Intel sgx explained. *Cryptology ePrint Archive*, 2016.

R. Gehler, S. Hasarfaty, Y. Moyal, and Y. Siam. Intel converged security and management engine (Intel CSME) security. Technical White Paper 1.5, Intel Corporation, 2022.

J. Greene. Intel trusted execution technology hardware-based technology for enhancing server platform security. White paper, Intel Corporation, 2013.

Intel Corp. Intel vPro PCs feature silicon-enabled threat detection. Technical white paper, Intel Corporation, 2022.

J. Li and M. Gao. A comprehensive survey of hardware-based security techniques from an architectural perspective. *Journal of Systems Architecture*, 168(C), Nov 2025.

K. Markantonakis and K. Mayes. An overview of the GlobalPlatform smart card specification. *Information Security Technical Report*, 8(1):17–29, 2003.

TCG, 2026. *Trusted Platform Module 2.0 Library Part 1: Architecture*. Trusted Computing Group, March 2026. Version 185.

N. Will and C. Maziero. Intel Software Guard Extensions applications: A survey. *ACM Computing Surveys*, 55(14s), July 2023. ISSN 0360-0300.

N. Will, R. Condé, and C. Maziero. *Mecanismos de Segurança Baseados em Hardware - Uma Introdução à Arquitetura Intel SGX*, pages 49–98. Sociedade Brasileira de Computação, 2017.