解　説　論　文

Trusted Execution Environmentの実装とそれを支える技術

Implementation of Trusted Execution Environment and Its Supporting Technologies

須崎有康 Kuniyasu SUZAKI

アブストラクト　現在の CPU ではクリティカルな処理を既存の OS から物理的に隔離して実行するために Trusted Execution Environment (TEE) 機能が提供されている．有名なところでは Intel の SGX や AMD の SEV，Arm の TrustZone であるが，オープンアーキテクチャである RISC-V でも TEE 開発が進められている．残念ながらこれらの TEE の機能は CPU によって大きく異なる．TEE の共通機能は隔離実行を提供するのみであり，その信頼性を支える技術は別に用意する必要がある．本論文では，各 TEE 実装詳細を解説するとともに TEE で使う機密情報を厳密に管理するための信頼の基点 (Root of Trust) や TEE を含むプラットフォーム及び実行するコードの真正性を確認するリモートアテステーションなど，TEE の信頼性を支える技術についても解説する．また，TEE の開発環境，脆弱性，規格活動も紹介する．

キーワード　Trusted Execution Environment，信頼の基点，リモートアテステーション

Abstract　Current CPUs have a Trusted Execution Environment (TEE) mechanism to run a critical process in isolatation from the operating system. Famous TEEs are Intel SGX, AMD SEV, and Arm TrustZone. In addition, the open architecture "RISC-V" has some proposals for TEE implementation. Unfortunately, TEE functions depend on CPU implementation. The common function of a TEE is isolated execution only, which requires supporting technologies for secure processing. In this paper, the details of each TEE implementation as well as its security-supporting technologies, i.e., Root of Trust for critical information and Remote Attestation for verifying CPU and code integrity, are discussed. The software build environment, vulnerability, and standardization activities are also introduced.

Key words　Trusted Execution Environment, Root of Trust, Remote Attestation

1. はじめに

近年の OS はサポートしなければならないデバイスの多量化，アプリケーションに提供する機能の多様化などにより，大規模かつ複雑になっている．一方，インターネットにつながることが当然であり，OS 及びその上のアプリケーションがインターネットからの攻撃に晒されている．現状では鍵管理などのクリティカルな処理も OS の一部，あるいはアプリケーションとして実行されているため，OS の脆弱性が突破されると機密情報の漏洩や処理の改ざんの恐れがある．このため，OS とは独立してクリティカルな処理が安全に実行できる機能である Trusted Execution Environment (TEE) が近年の CPU に含まれるようになった．ただし，TEE は基本的に隔離実行を提供するのみであり，暗号鍵などの機密情報を個々のプラットフォームで安全に保有する技術や TEE を含むプラットフォーム及び実行されるバイナリの真正性をリモートで第三者が確認する技術が含まれることで信頼できる実行環境となりうる．本稿ではそれぞれの TEE 実装詳細を述べるとともに，信頼性を支える技術である信頼の基点 (Root of Trust) 及びリモートアテステーション (Remote Attestation) も含めて解説する．また，実行・開発環境，活用事例，既知の脆弱性，規格や関連団体などについても動向を述べる．

TEE は隔離実行環境であるが，TEE が初めてというわけではなく，PC やスマートフォンには既に幾つかの隔離実行機能が備わっている．文献(1) では PC の隔離実行を Hardware Isolated Execution Environment (HIEE) としてまとめられている．TEE はその中の一つの機能にすぎないが，一般ユーザに開放されているという点でほかと大きく異なる(注1)．

図 1 に PC やスマートフォンに備わる隔離実行機能，及びそれを操作できる関係者を示す (緑背景はメイン CPU とは異なるハードウェア，灰色背景はメイン CPU 内の機能)．Trusted Platform Module (TPM) は通常の PC に標準で採用されている完全に CPU から独立した別チップである．全ての機能はハードコードされて更新はできないが，鍵生成，機密情報保存，乱

須崎有康　正員　セキュアオープンアーキテクチャ・エッジ基盤技術研究組合&国立研究開発法人 産業技術総合研究所
E-mail k.suzaki@aist.go.jp
Kuniyasu Suzaki, Member (Technology Research Association of Secure IoT Edge application based on RISC-V Open architecture, 3 Kanda-Neribeichou, Chiyoda-ku, 101-0022 Japan & National Institute of Advanced Industrial Science and Technology, 2-3-26 Aomi, Koto-ku, Tokyo 135-0064 Japan).

電子情報通信学会 基礎・境界ソサイエティ
Fundamentals Review Vol.14 No.2 pp.107–117 2020 年 10 月

(注1)：GPU も隔離実行を提供しているが，実行されるコードがセキュリティとは関係なく，セキュリティ対策もされていないために対象から外れる．

数生成などの機能を提供し，Trusted Boot や BitLocker などの PC と連携した重要なセキュリティ機能が TPM に依存している．TPM は PC の電源投入時から BIOS やペリフェラル，ブートローダ，カーネル，アプリケーションなど，全ての状態をハッシュのチェーンとして TPM 内のレジスタに保存する信頼の基点であり，外部のサーバで検証するリモートアテステーションの機能を提供する．この検証方法は電源投入時から始まることを前提とするため，Static Root of Trust Measurement (SRTM) と呼ばれている．

Secure Element (SE) は多くのスマートフォンに使われている HIEE である．TPM のように別チップであり，SIM カードに含まれている実装もある．SE は元々IC カードでの信頼の基点であり，スマートフォンの「おサイフケータイ」を代表とする電子マネーの決済情報管理で活用されている．SE ではソフトウェアの更新可能なものもあるが，更新者は製品提供者に限られている．

Intel ME (Management Engine) や AMD PSP (Platform Security Processor) は CPU と対になっている HIEE である．具体的なハードウェアは Intel ME はチップセット上の Intel Quark X86-based (32 bit) であり，AMD PSP は CPU メインダイに乗った Arm Cortex-A5 (32 bit) である．Intel ME では Minix が動作していることが公表されており，電源管理技術である Active Management Technology (AMT) のために HTTPS サーバとしても機能している．また，後述するリモートアテステーションのための鍵管理なども行っており，信頼の基点にもなる．これらのソフトウェアの更新はできるが，提供ベンダーのみに限られている．

CPU 内にも HIEE の機能がある．System Management Mode (SMM) は Intel アーキテクチャで最も古い隔離実行環境である．1990 年にリリースされた Intel 80386SL より導入され，BIOS 専用のメモリを確保する．このメモリは OS から隔離され，BIOS 関連のコードのみ実行することができる．更新もできるが BIOS/UEFI ベンダーに限られている．

Trusted Execution Technology (TXT) は Intel x86-64 に導入された隔離実行機能である．TPM の信頼の起点が電源投入時からに限定されていたのに対して，TXT は信頼の起点を CPU の実行の途中からに変更する Dynamic Root of Trust Measurement (DRTM) の機能を提供する．残念ながら保護されるコードが使えるメモリは 64 KB であり，汎用に使われることはなかった．基本的に完全性検証のみに活用され，tboot [2] や Flicker [3] が活用事例である．

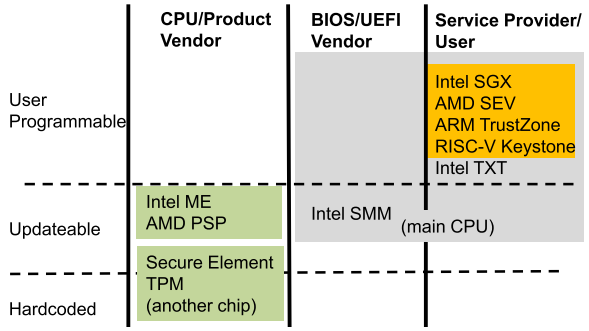

図 1　PC やスマートフォンで使える隔離実行機能 (オレンジ部分が TEE).

Intel TXT に対して TEE は既存の OS と共存できる隔離実行機能である (図 1 のオレンジ色で囲った部分)．現在の利用可能な CPU で提供されるものは Intel SGX，AMD SEV，Arm TrustZone，RISC-V Keystone などである．TEE は CPU 自体が OS から隔離した別の実行環境を提供し，一般のユーザや外部のサービスプロバイダが使える点が大きく異なる．隔離実行環境という点で TEE と名前がつく機能は共通しているが，その詳細は異なっているため，多くの誤解を生んでいる．なお，TEE 実装の違いは既に解説論文 [4], [5] がある．

2. 各 TEE アーキテクチャ

本章ではまず TEE の概念を理解するために GlobalPlatform が定義している TEE アーキテクチャを紹介する．この定義が全てに当てはまるわけではないが，基本的な概念を理解するのには役に立つ．その後，実際の CPU で実装されている TEE アーキテクチャArm TrustZone，Intel SGX，AMD SEV, RISC-V Keystone を説明する．なお，それぞれの実装において用語の統一が取れておらず，TEE は Secure World あるいは Enclave とも呼ばれる．可能ならば Trusted OS が用意され，その上で Trusted Application (TA) が動く．また，TEE に対する通常実行環境は Rich Execution Environment (REE) であり，Normal World とも呼ばれ，通常の OS やアプリケーションが実行される．本解説では実装において用語を使い分ける．

2.1 GlobalPlatform の TEE 定義

IC カードのセキュリティ仕様などを定義している GlobalPlatform では TEE についても定義 [6] している．ここでは CPU を Normal World と Secure World の二つのワールドに分けることを想定しており，それぞれのワールドで通常の OS と Trusted OS を実行する．この様子を図 2 に示す．Secure World の TA は通常の OS でアプリケーションとして動く TA Client からロード，起動，停止される．

この GlobalPlatform の TEE アーキテクチャでは以下の七つの要件を定義している．

（1） OS とは独立に実行すること．
（2） ほかの TA とは独立して実行すること．
（3） 信頼されたもののみが Trusted OS と TA を修正でき

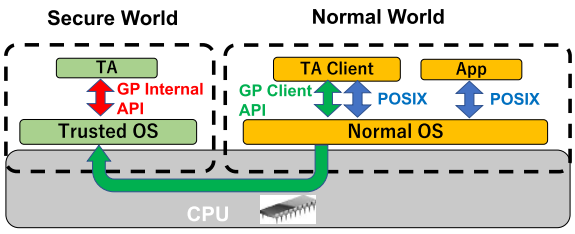

図 2　GlobalPlatform の TEE アーキテクチャ

ること．

（4） ブートプロセスが System on Chip (SoC) にバインドされ，TEE ファームウェアと TA の真正性と完全性を強制できること．

（5） 信頼できるストレージを用意すること．

（6） ペリフェラルには安全にアクセスすること．

（7） 最新の暗号化技術を使うこと．

GlobalPlatform では更に二つの API を定義している．一つは通常 OS で実行されるアプリケーション (図 2 中の TA-Client) が使う TA のロード，起動，停止などを扱う TEE client API である．もう一つは TA が Trusted OS とやり取りを行う TEE Internal Core API である．TEE Internal Core API は通常の UNIX で使われている POSIX と大きく異なり，暗号などのセキュリティに関する API を提供する．GlobalPlatform は元々 Arm TrustZone との関係が深く，両者はかなり似た構造になっている．

2.2 Arm TrustZone

Arm TrustZone は最も広く使われている TEE であり，スマートフォン，ゲームマシン (例：Nintendo Switch)，セットトップボックスなどで活用されている．CPU を Normal World と Secure World に分割する 2 World View モデルの TEE であり，Normal World では通常の OS，Secure World では Trusted OS を起動する．この詳細は論文(7)，(8) でまとめられている．

Arm の TrustZone の歴史は古く 2003 年の ARMv6K から導入されているが，実際に使われるようになったのは ARMv7, ARMv8 からである．また，スマートフォンで使われている高性能の Cortex-A と組み込みで使われている Cortex-M ではアーキテクチャが全く異なる．Cortex-M では高い応答性を想定した特殊な TEE になっている．本稿では Cortex-A を中心に TrustZone を説明する．

図 3 に Cortex-A の TrustZone のアーキテクチャを示す．Cortex-A では特権レベル (Exception Level) が 4 つある．Secure World と Normal World それぞれで全ての特権レベルを利用可能であり，OS を動かすことができる．Normal World と Secure World の双方を行き来するには Secure Monitor Call (SMC) 命令を用い，セキュアモニタを通して通信・切り替えを行う．つまり CPU Core は SMC 命令により Secure World と Normal World のモードを動的に切り替える．

Secure World に割り当てるメモリ領域やペリフェラルなどのハードウェア資産は起動時に固定的に設定されている．それぞれのワールドに割り当てたペリフェラルの割り込みは Secure World と Normal World それぞれで別の割り込みリクエストになっている．Normal World では Interrupt ReQuest (IRQ) として割込みを受けつけ，Secure World では Fast Interrupt reQuest (FIQ) を受け付ける．ハンドラは通常の OS あるいは Trusted OS のものが使われる．

Cortex-A の TrustZone 起動手順を図 4 に示す．図中の BL は

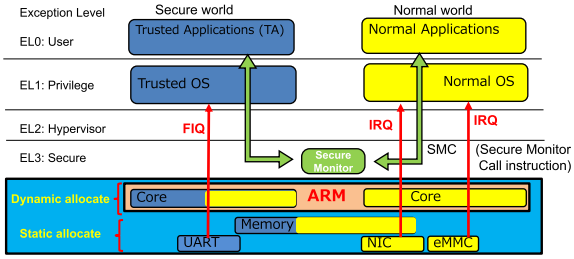

図 3 TrustZone の構成図

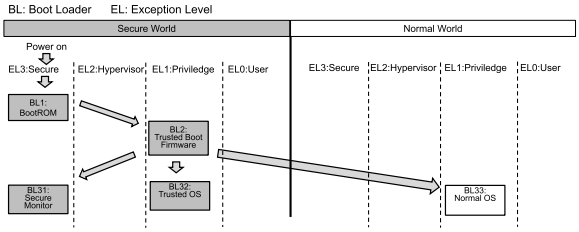

図 4 TrustZone の起動手順

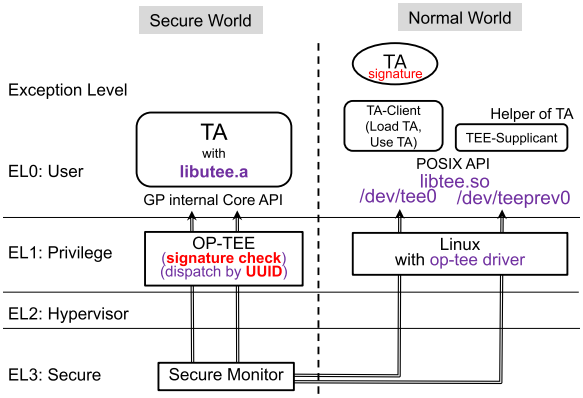

図 5 OP-TEE の構成図

BootLoader のレベルを表しており，BL1，BL2，BL31，BL32，BL33 の 5 種類で構成される．電源投入直後に実行される BL1 は SoC 固有の BootROM であり，Secure World で最も特権レベルが高い EL3 で処理される．BL1 では BL2 である Trusted Boot Firmware を検出して，EL1 で制御を渡す．BL2 では Secure World で使うメモリの確保やペリフェラルの割り当てを行う．その後は BL31 として Secure World の EL3 でセキュアモニタを起動，BL32 として Secure World の EL1 で Trusted OS を起動，BL33 として Normal World の EL1 で通常の OS を起動し，定常状態へ移行する．

Trusted OS としてはオープンソースの OP-TEE (Linaro がサポート) (9), (10)，Open-TEE (Aalto University がサポート) (11), (12)，Trusty (Google がサポート) (13) などがある．商用としては Qualcomm の QTEE (旧 QSEE) (14) Samsung の Knox (15) や Teegris (16)，TrustedKernel 社の TrustKernel (17)，Trustonic の Kinibi，Huawei の TrustedCore (18) がある (注2)．

図 5 に Trusted OS 実装の一つとして OP-TEE の構造を示す．REE には Linux を想定しており，ドライバ経由でセキュアモニタと通信する．Linux 側は TA を活用するアプリである TA-Client

(注2)：Apple の Secure Enclave は別 Core 実装のため，本文から外した．

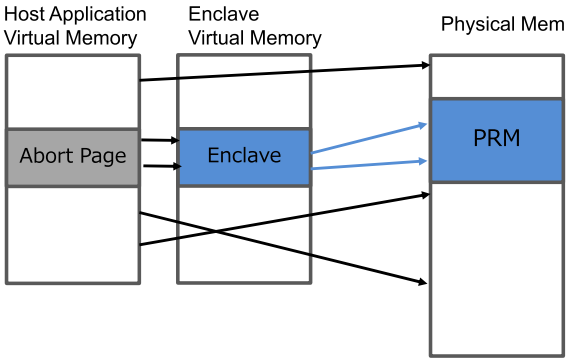

図 6　Intel SGX のメモリマップ

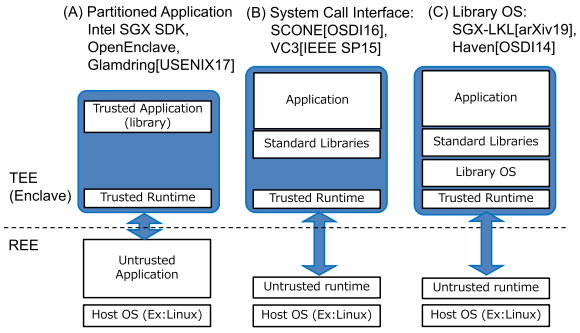

図 7　SGX のアプリケーション実行方式

と TA からの OS 側への要求 (ファイルへの書き込みやネットワークの通信) などをプロキシ的に行う TEE-Supplicant で構成される．Linux 側には二つのデバイス/dev/tee0 と/dev/teeprev0 が作られる．tee0 は TA-Client，teeprev0 は TEE-Supplicant が使う．TA-Client には TA との通信を管理する libtee がダイナミックリンクされる．TA は TA-Client からロードされる．TA には署名が行われており，OP-TEE でその署名検証を行う．TA には libutee.a のライブラリが静的リンクされ，GlobalPlatform の TEE Internal Core API を使う．OP-TEE 上の TA を識別するために UUID が割り当てられ，TA-Client からは UUID を用いて必要な TA と通信を行う．

2.3　Intel SGX

Software Guard Execution (SGX) は Intel Skylake 以降に付加された隔離実行機能である．SGX の実装で特別なハードウェアはメモリを暗号化する Memory Encryption Engine (MEE) 程度で，ほとんどの機能はマイクロコードで実装されている．この実装については文献(19)～(21) が詳しい．SGX は Enclave と呼ばれる隔離された実行環境を動的に提供するが，ここでの実行レベルはユーザレベルである Ring3 のみである．Enclave が使うメモリは起動時に確保された最大 128 MB の Processor Reserved Memory (PRM) のみであり，暗号化される．

SGX では OS 上のアプリのメモリ空間の一部が隠されるシングルアドレスモデルを採用している．その様子を図 6 に示す．通常のアプリケーション内で Enclave 用に作ったライブラリが Enclave のメモリにマップされる．この Enclave のメモリは起動時に確保した PRM のみを使い，OS とは完全に分離している．

Enclave 内では OS を提供するための特権レベルがなく，基本的な OS 機能は SGX 用の命令セットと Architectural Enclave と呼ぶ特別な Enclave によって処理される．Architectural Enclave には鍵管理を行う Provisioning Enclave，生成管理を行う Launch Enclave，署名管理を行う Quoting Enclave がある．

アプリケーションの実行方式は図 7 に示すように三つのタイプに分けられる．アプリケーションからクリティカルな処理を切り出す分割タイプ (A)，アプリケーション自体を Enclave 内で実行可能とするが，システムコールは Enclave 外の OS に依存するタイプ (B)，できるだけ Enclave 内で実行する LibraryOS タイプ (C) に分けられる．

タイプ (A) を代表する実装としては Intel SGX SDK や Open Enclave [22]，Glamdring [23] がある．アプリケーションからクリティカルな部分を切り出してライブラリとして実装する必要があり，アプリケーションごとに作り直しの必要がある．できるだけクリティカルな処理を少なくして脆弱性を抑える Trusted Computing Base (TCB) の方針に沿う実装になっている．

タイプ (B) を代表する実装としては SCONE [24] や VC3 [25] がある．このタイプでは Library 以下のシステムコールは通常の OS に依存する．アプリケーションは作りやすくなっているが，TCB の視点からアプリケーション自体にクリティカルな処理以外が含まれることや Library の整備のためにタイプ (A) より Enclave 内のコード量は大きくなる．

タイプ (C) を代表する実装としては Haven [26], [27], SGX-LKL (Linux Kernel Library) [28], Graphene [29] がある．このタイプも (B) と同じくアプリケーションにできるだけ変更を求めない．また，Enclave 内の LibraryOS がシステムコールの代用できる部分を実行することで効率化を図っている．しかし，タイプ (B) よりコード量が多く，脆弱性の危険が増える．

2.4　AMD SEV

AMD が提供する TEE は Secure Encrypted Virtualization (SEV) と呼ばれ，EPYC シリーズで利用できる．SEV は名前が示すとおりに仮想化を拡張した TEE であり，クラウド環境の仮想マシン (VM) ホスティングで使われることを主眼としている．今まで挙げた TEE とは異なり，TEE 用に専用のメモリ領域やペリフェラルを用意するのではなく，VM の実行自体を秘匿化する．つまり，VM のメモリと仮想ディスクの暗号を管理することでサーバ上にハイパーバイザから覗き見できない VM を実現する．この VM が TEE として振る舞う．SEV を活用するソフトウェアとしては Redhat 社が開発を進める Enarx [30] がある．

SEV は信頼の基点を実現する Platform Security Processor (PSP) とリモートアテステーションの技術を組み合わせて，TEE を実現している．ここでは VM (すなわち TEE) の安全な起動とマイグレーションを保証する．

安全な起動にはリモートアテステーションにより AMD CPU の真正性を確認したのち，PSP とリモートのユーザ間でセキュ

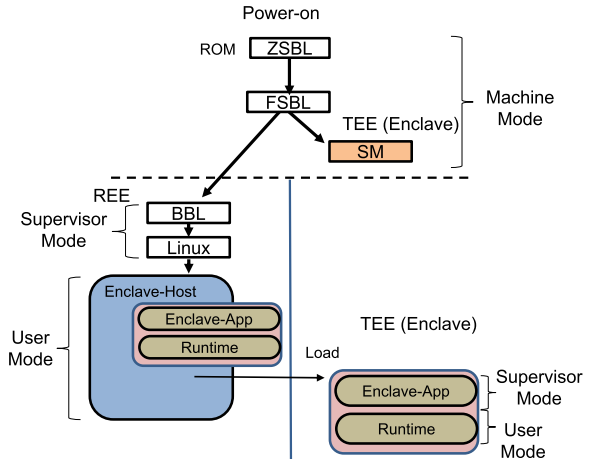

図 8　RISC-V Keystone 起動手順

アな通信チャネルを確立して，暗号化された仮想ディスクの復号鍵を渡す．メモリは暗号化されており，仮想ディスクもメモリもハイパーバイザからも覗き見ることはできない．

安全な VM マイグレーションにはメモリの安全な転送が要求されるが，SEV がハイパーバイザに依存することなくメモリ，及び鍵の転送を保証する．また，CPU のスナップショットを取るための拡張である Secure Encrypted Virtualization-Encrypted State (SEV-ES) では CPU レジスタの内容を暗号化して保存する．

2.5　RISC-V Keystone

RISC-V はオープンアーキテクチャであり，命令セット仕様 (ISA) ばかりでなくプラットフォーム仕様や実装までオープンソースとして公開されているため，多くの研究のベースに使われている．TEE の研究も MIT の Sanctum [31]，MI6 [32]，UC Berkeley の Keystone [33]，Hex-Five 社の組込み用 Multi-Zone [34]，グラーツ工科大学の TIMBER-V [35] などがある．このうち，本稿では最も開発が活発な Keystone を取り上げて比較する．

図 8 に Keystone を想定した起動手順を示す．RISC-V では特権レベルの高い順にマシンモード，スーパーバイザモード，ユーザモードがある．電源投入後はマシンモードで ROM 化された Zero Stage Boot Loader (ZSBL) が動作し，ストレージにある First Stage Boot Loader (FSBL) をロード・起動する．この後，Enclave を永続的に管理するセキュアモニタ (SM) が作られる．FSBL は更に一つ Enclave を作り，通常の OS を起動させる．図中では Berkeley Boot Loader (BBL) と Linux が書かれている部分である．ここでは複数のプロセスが作成される．図中では Enclave を使う Enclave-Host が書かれている．Enclave-Host はセキュアモニタに Enclave の作成を要求し，TA に相当する Enclave-App と Trusted OS に相当する Runtime をまとめて作成された Enclave にロード・起動する．

Enclave 作成のためにメモリの割り当てが必要である．図 9 にその様子を示す．Keystone は RISC-V International で規格化されている Physical Memory Protection (PMP) の機能を

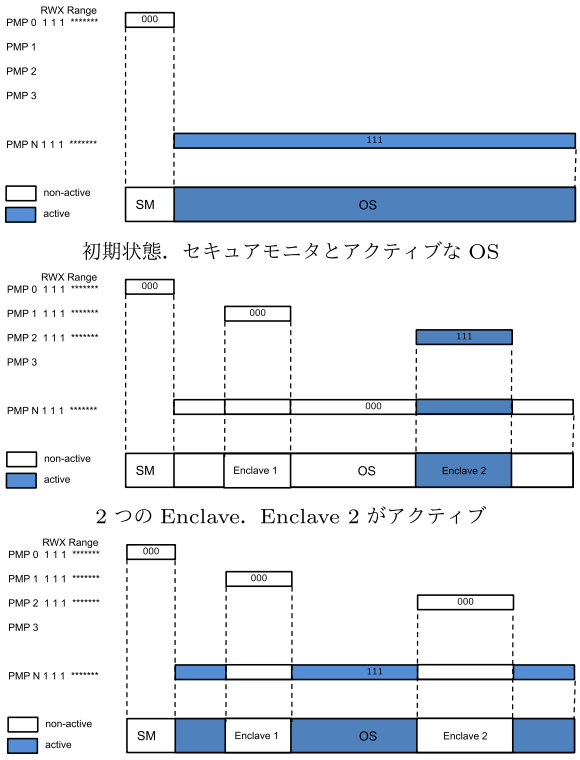

図 9　PMP による Enclave メモリ管理

活用してメモリの領域分割を行う．PMP は現在の仕様で最大 16 個のレジスタで物理メモリの領域を管理するメカニズムである．Keystone では特権レベル最上位の PMP レジスタ 0 にセキュアモニタを割り当て，特権レベル最下位の PMP レジスタ N を OS (Linux) で使う．この間の PMP レジスタがユーザの Enclave に使われる．初期状態を図 9 上段に示す．この状態ではほとんどのメモリを OS (Linux) が保持している．図中で青色のアクティブな PMP の領域のみがアクセス可能で，ほかの領域にはアクセスできない．Enclave 生成が要求された場合，Linux はメモリを開放してセキュアモニタに通知する．セキュアモニタは使われていない PMP レジスタにそのメモリ領域を割り当て，Enclave を生成する．図 9 中段に二つの Enclave が生成されたのちに，Enclave 2 がアクティブな状態を示す．図 9 下段では OS がアクティブであるが，使えるメモリ領域は分断されている．

Keystone ではセキュアモニタがマシンモードを使う．Enclave 内ではスーパーバイザモードで軽量 OS 相当の Runtime が実行される．Runtime はいろいろな実装が可能であり，デフォルトでは Eyrie だが，形式検証されたマイクロカーネルである seL4 [36] を使った実装もある．通常の Linux と Enclave 内のプログラムが通信する場合はセキュアモニタ及び Eyrie を通して行われる．

3.　信頼の基点 (Root of Trust)

先に説明したように基本的に TEE は隔離実行環境を提供するのみである．TEE で使われる鍵は「信頼の基点」に保存される

のが一般的である．この信頼の基点を基に次章で紹介する TEE を含むプラットフォームや実行するバイナリの真正性を確認するリモートアテステーションを行うことができる．

信頼の基点は軽量なコプロセッサとして実装されることが多く，鍵管理のほかに鍵生成，乱数生成などの機能を有する．基本的にサイドチャネル攻撃に対する耐タンパ技術で作成されるのが望ましいが，脅威分析から物理攻撃がない（例えば，クラウド環境なので物理的侵入は不可能）と想定されれば，この条件を外すことができる．GlobalPlatform では信頼の基点の定義と要件を文書化している[37]が，個々の実装によって大きく異なる．それぞれの違いを以下で紹介する．

TPM は信頼の基点の一つである．鍵生成，乱数生成ばかりでなく，リモートアテステーションで使われるレジスタなどの機能を有する．TEE が現れる以前に規格が決まっており，TEE との連携はない．

Secure Element (SE) はスマートフォンでは多く使われる信頼の基点である．SE では耐タンパ技術を適用したセキュア領域を有しており，機密情報はそこで処理される．GlobalPlatform では TEE から SE への API[38]も定義している．

Arm CPU を使ったスマートフォンなどでは SE が信頼の基点として使われる例が多いが，Arm の CPU ライセンスからハードウェアの SoC を作成する際に付加される Intellectual Property (IP) 製品に信頼の基点となりうるものがある．製品は Arm 社が扱う CryptoCell や Rambus 社の CryptoManager などである．Qualcomm 社は CryptoManager を 2014 年に導入して Snapdragon に組み込んでいる[39]．

Intel 及び AMD が提供する x86-64 CPU にはそれぞれ独自の信頼の基点を入れている．Intel の ME や AMD の PSP である．Intel の SGX では先に説明したように Architectural Enclave で ME の管理する鍵にアクセスすることができる．また，AMD の SEV では SEV Firmware が PSP の管理する鍵にアクセスすることができる．

RISC-V については Rambus 社の CryptoManager RISC-V や SilexInsight 社の eSecure がある．オープンソースの提案としては Google 社が開発を進める Open Titan RISC-V がある．

4. リモートアテステーション

TEE 単体では隔離実行するのみであり，そのプラットフォームが信頼できるものであるか，ユーザが意図したコードが実行されるかを確認する仕組みがない．これを補完するのがリモートアテステーションである．リモートアテステーションでは信頼の基点に保存された機密情報を使って，TEE を含むプラットフォーム及び TA の真正性判定を行う．また，リモートアテステーションの判定後はリモートアテステーションサーバと TEE 間でセキュリティチャネルが確立され，鍵など機密情報の提供を行うこともできる．リモートアテステーションについては Asia CCS 2017 で行われたリモートアテステーションチュートリアルの資料[40]が参考になる．以下，各 TEE で使われるリモートアテステーションについて解説する．Arm の TrustZone には単体のリモートアテステーション機能がないので割愛する．最後の 4.5 節ではリモートアテステーションは「誰が誰を信用しないのか，あるいは信用するのか」をサーバ系とクライアント系で異なることをまとめた．

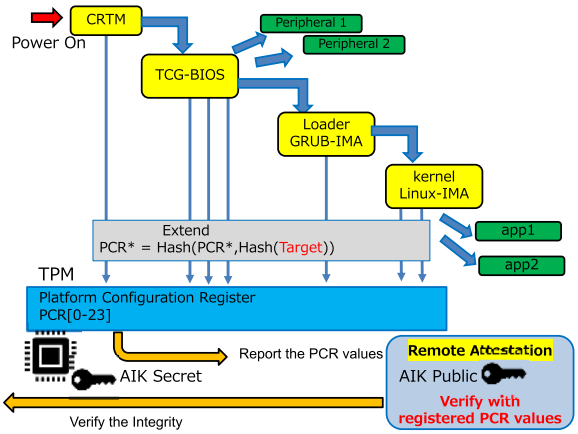

図 10　TPM でのリモートアテステーション

4.1 TPM でのリモートアテステーション

TPM は TEE が普及する以前にリモートアテステーションの機能を提供しており，その歴史は古い(注3)．図 10 に TPM によるリモートアテステーションの様子を示す．リモートアテステーションの事前準備として，TPM から送られてきたことを検証するための署名鍵 Attestation Identity Key (AIK) を TPM 内で作成し，事前にリモートアテステーションサーバに AIK 公開鍵を登録する．TPM には Platform Configuration Register (PCR) と呼ばれるレジスタがあり，電源投入時からハードウェアプラットフォームやソフトウェアのハッシュ値を取り，記録する機能がある．記録は extend と呼ばれる PCR 値に計測したハッシュ値を足し合わせて，再度ハッシュを取り，その値を PCR に書き込む手続きで更新される（図 10 中央）．つまり，ハッシュ値からハッシュを取るチェーン計算であり，一部でも値が異なると最終結果が異なる．PCR は複数のレジスタから構成され，BIOS が使うレジスタ，Linux カーネルが使うレジスタなどが分けられている．起動は Trusted Boot と呼ばれ，電源投入直後は Core Root of Trust Measurement (CRTM) から始まることが規定されている．CRTM では BIOS を extend した後，BIOS に制御を移す．Trusted boot では BIOS やブートローダ，カーネルなどにもハッシュ値を計測する機能がある必要がある．Trusted BIOS ではペリフェラルやブートローダを extend して制御を移す．この手順がカーネルまで続く．Linux カーネルではアプリケーションのロード前に計測する Integrity Measurement Architecture (IMA) に対応する必要がある．このような起動の記録を外部のリモートアテステーションサーバに依頼する．検証には PCR レジスタの値に AIK で署名して，リモートアテステーションサーバに送る．検証が正しければ，そ

(注3)：Direct Anonymous Attestation (DAA) が使われているが，プライバシーの問題，スケーラブルでない，などの指摘があり，再考が行われている[41]．

れに対するサービス提供を行う．有望と思われたサービスにネットワークの接続判断を行う Trusted Network Connect (TNC) があったが想定されたオフィスより制御システムで活用されている．リモートアテステーションではないが，Trusted Boot を最も活用しているのは Windows の Bitlocker であり，正しい起動が行われないと Windows が起動しない．

4.2 Intel SGX でのリモートアテステーション

Intel SGX ではローカルアテステーションとリモートアテステーションの二つの機能を提供する．ローカルアテステーションはプラットフォーム内 (Intra-Platform) アテステーションとも呼ばれ，Enclave 間の認証を行う機能である．ここでは個々の CPU がもつ固有鍵で作成されたレポートを検証することで，Enclave が同じ CPU で動いていることを認証する．認証後はセキュアチャネルを確立し，Enclave 間でのセキュア通信が可能になる．この詳細は Intel の HP [42]で解説されている．

リモートアテステーションはプラットフォーム間 (Inter-Platform) アテステーションとも呼ばれ，外部のアテステーションサーバで正規の CPU 上で意図したソフトウェアが Enclave で動作することの認証を行う機能である(注4)．TPM とは異なり，OS など Enclave 外で動作しているソフトウェアは信頼しておらず，検証する作業も行われない．ただし，プラットフォームを認証するためには想定するバージョンの BIOS/UEFI が入っていることが検証対象になっている．

Intel SGX のリモートアテステーションを使うためには Intel が提供する Intel Attestation Service (ISA) を信頼する必要がある．しかし，第三者機関にその機能を委譲することも検討されている．Intel からは Data Center Attestation Primitives (DCAP) が提供されており，OPERA (OPEn Remote Attestation) [43]の研究もある．

4.3 AMD SEV でのリモートアテステーション

AMD SEV においてもリモートアテステーションが想定され，暗号化された仮想ストレージの複号鍵の提供を行う．現在の AMD SEV においても Intel SGX と同様に AMD が提供する認証サーバ (AMD Key Server) を信頼する必要がある．

AMD Key Server は各 CPU に埋め込まれた Chip Endorsement Key (CEK) に対して署名しており，信頼の輪の元になっている．信頼の基点である PSP では ECDSA 鍵を生成することができ，リモートアテステーションのために Platform Endorsement Key (PEK) を作る．この PEK は CPU 内の CEK によって署名され，AMD Key Server でプラットフォーム認証される．認証されたプラットフォーム上にはユーザがセキュアチャネルを構築して，暗号化された仮想ディスクの復号鍵を渡すことができる．

(注4)：SGX では DAA を拡張した Enhanced Privacy ID (EPID) をベースにしている．

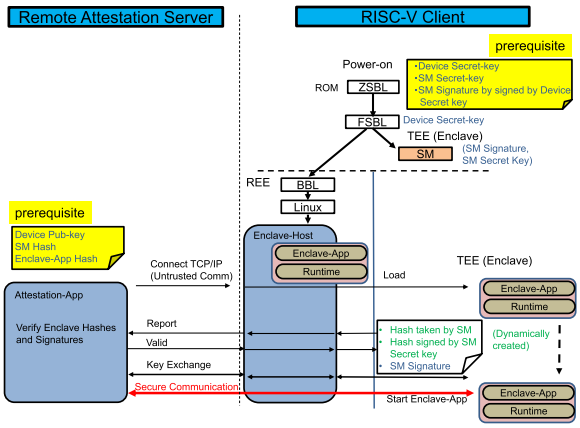

図 11　RISC-V でのリモートアテステーション

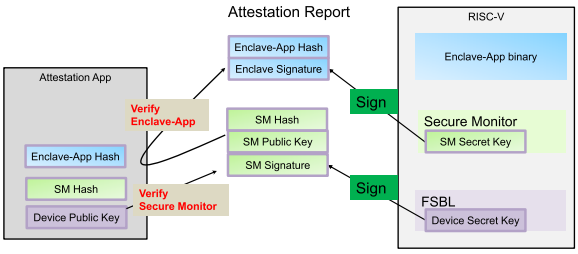

図 12　RISC-V でのリモートアテステーションで使われるレポート

4.4 RISC-V Keystone でのリモートアテステーション

RIC-V Keystone においてもリモートアテステーションが利用可能であるが，まだハードウェアベースの信頼の基点がなく，現在の実装ではデバイス固有鍵も SD カードからロードされる．

図 11 に起動からリモートアテステーションを行うまでの手続きを示す．この図は 2.5 節のブート図 8 を詳細化している．ここで重要なのは，黄色で示されている事前準備の部分である．クライアント側ではデバイス固有秘密鍵，セキュアモニタ公開鍵・秘密鍵が事前に設定されている．検証するサーバ側ではデバイス固有公開鍵，セキュアモニタのハッシュ値，Enclave で実行される Enclave-App のハッシュ値を有する．

図 11 では TA のロード要求はリモートアテステーションサーバにある Attestation-App からネットワーク経由で要求されることを示している．Enclave-Host は TA である Enclave-App をロードするが，その際にリモートアテステーションが行われる．

リモートアテステーションで取り交わされるレポートの詳細を図 12 に示す．プラットフォーム認証を行うためのセキュアモニタのレポートと TA コード認証を行うためのレポートが作られる．セキュアモニタのレポートとしては，セキュアモニタのハッシュ値，セキュアモニタ公開鍵，及びそれに対して行ったデバイス固有秘密鍵による署名が含まれている．Attestation-App では受け取ったレポートを既知のデバイス固有公開鍵，セキュアモニタのハッシュ値で検証する．これにより，プラットフォームの真正性とその後に使うセキュアモニタ公開鍵が正しいことが分か

る．この後に TA コード認証のためのレポートをセキュアモニタ公開鍵と既知の Enclave-App のハッシュ値で検証し，正しければ図 11 下に示したように Enclave-App と Attestation-App が鍵交換した後，セキュアチャネルを確立する．

4.5 リモートアテステーションの活用者の違い

上記で各リモートアテステーションの機能を紹介したが，クラウドなどのサーバ系と IoT，PC，スマートフォンなどのクライアント系ではその活用者が大きく異なる．どちらもリモートの TEE を安全に実行する目的は同じだが，クラウドでは活用者が一般ユーザであり，クラウド管理者や OS を信頼せず，アプリケーションが TEE で実行されることを確認する．IoT では活用者がリモートの管理者であり，物理的に分散して管理が難しい機器においてその機器の真正性と意図するアプリケーションが TEE で実行されることを確認する．また，PC やスマートフォンではユーザが管理者であるが，TEE の活用者は課金などサービス提供者であり，ユーザ自体を信頼せず，課金システムが TEE で実行されることを確認する．

5. 実行・開発環境

TEE ではその実行・開発環境に幾つかの制約がある．

実行方式：多くの TEE 内では fork や exec を実行することができない．これは TA が REE からロードする際に検証を行い，認証された TA のみの実行に限定するためである(注5)．

REE からロードする際の検証では暗号化されないままのコードが基本である．ただし，この方法ではバイナリ自体の機密性を守ることができない．これを解決するために Intel SGX では暗号化されたバイナリをロードする機能 Protected Code Loader がある．この場合にはリモートアテステーションにより復号鍵を共有する必要がある．また，OP-TEE では事前にセキュアストレージに保存された TA については実行できる仕組みがある．この本来の目的は OS が起動する前に TA を実行することである．

デバッグ：TEE の目的は処理の隠蔽であり，その中で実行される TA のデバッグは容易ではない．これに対処するために Intel SGX ではデバッグモードを用意している．また，OP-TEE では gdb による debug 用のインタフェースを提供している．また，QEMU による開発を想定している場合も多い．この辺りは OP-TEE Documentation [10] が詳しい．

セキュリティ：TA のプログラム自体もセキュリティを考慮して作られる必要がある．特に Arm TrustZone ではメモリサイズも小さく，Trusted Computing Base (TCB) を考慮したプログラミングが行われている．最近ではメモリ安全を保証する Rust 言語を使う事例も多くなった．また，TEE と REE 側での通信 (TEE から REE へは OCALL，REE から REE へは ECALL) にはポインタ操作やバッファサイズに脆弱性がないことを保証するため，Enclave Definition Language (EDL) と呼ぶツールでコードが自動作成される．それぞれの実装は Intel SGX SDK では edger8r，RISC-V Keystone では kedger と呼ばれる．

性能計測：TEE 内の処理は REE から隠蔽されるため，詳細なパフォーマンスを測定することが難しい．これを解決するため，SGX-Perf [44] や TEE-Perf [45] などの研究があるが，オーバーヘッドが大きかったり，一部のパフォーマンスしか継続できない問題がある．Intel Processor Trace (PT) などのハードウェア機能もあるが，デフォルトでは Enclave を対象外にしている．

6. 活用事例

TEE を利用するソフトウェアには暗号処理に関するものが多かったが，利用できるメモリサイズが大規模化したことで活用範囲も広くなった．つまり，「隠すもの」が鍵からコードやデータに移っている．また，リモートアテステーションの章でも述べたようにクラウドなどのサーバ系と IoT，PC，スマートフォンなどのクライアント系では管理者の想定が異なり，TEE の活用方法が異なっている．

鍵管理の代表には Android OS の鍵管理を行う Keymaster がある．Keymaster は TrustZone での実装を想定しており，QTEE や OP-TEE ではそのための TA を提供している．

メモリ大規模化により機械学習や機密データ管理を要する分野での活用も増えている．機械学習では実行コードばかりでなく，その重み付けデータが重要であり，それらを秘匿する必要がある．VC3 [25] や M2R [46] では Intel SGX 内で MapReduce を実行しており，TensorSCONE [47] では SCONE をベースに TensorFlow を実行している．SkSES [48] では SGX のメモリが足りないゲノム解析に対して効率的なデータ構造を提案している．ゲノム解析としては SGX 外のメモリアクセスに対して，アクセスパターンを隠す oblivious RAM (ORAM) を組み合わせた提案 [49] もある．

通常 OS 側を守る技術も出てきている．REE 側の OS 用の侵入検知システムを SGX 内に含めた SGmonitor [50] や Dropbox などのインターネットサービスを安全に監査する LibSeal [51] がある．DelegateTEE [52] では TEE で安全性を確保して，オンラインサービスにログインの権限移譲を可能にする．ここでは PayPal や電子メールの管理 (SMTP/IMAP) などで利用可能なことが示されている．SGX 対応した WebAssembly の AccTEE [53] は更なる応用を期待させる．

更に，暗号機能自体を安全に行うために TEE 内で実装する提案もある．関数型暗号を SGX 内で実装する Iron [54] や暗号文と秘密鍵の妥当性を検証できる Verifiable Functional Encryption (VFE) と SGX を組み合わせた提案 [55] である．これらのような暗号技術と TEE の組み合わせは，双方で足りない部分を補完し，論理的な裏付けがある実装となるため，今後の展開が期待できる研究分野である．

(注5)：GlobalPatform が定義する Secure Domain では TA から TA が実行できることになっているが，筆者が知る範囲ではこれを正しく実装した例はない．

7. 既知の脆弱性

TEE は CPU 内の隔離実行機能であるが，物理的に別 Core ではなく，Core の状態を変えるのみである．その脆弱性をついた攻撃が知られている．例えばキャッシュ共有を悪用した Prime+Probe が SGX でも有効であることが論文[56]で示されている．ForeShadow (別名 L1TF: L1 Terminal Fault) [57]では，Spectre と同じくキャッシュの共有と投機的実行を組み合わせた攻撃も SGX で有効であることが示されている．RISC-V でも Out-of-Order で実行する BOOM では同様な攻撃が可能であることが分かっている[58]．また，Boomerang Attack [59]は TEE 内で TEE 外のメモリにアクセスできる機能を悪用する．REE から TEE のメモリアドレスにはアクセスできないが，その逆は可能である．すなわち，TEE 内でポインタでアクセスするプログラムのポインタを書き換えることで，本来のアプリケーションから隔離されているカーネル内の情報を取得する攻撃である．

実装の脆弱性については多くの発表があるが，既存 TEE の API/ABI の脆弱性サーベイ論文[60]は特に重要である．TEE と REE との通信については先に紹介した Edger などの形式検証を使ったデータ検証が活用されているが，それでも脆弱性を含んでいた場合が紹介されており，脆弱性除去の難しさを示している．

8. 関連組織・規格

TEE はビジネス的にも活用が見込まれ，CCC (Confidential Computing Consortium) が業界団体として立ち上がっている．また，GlobalPlatform の Security Evaluation Standard for IoT Platforms (SESIP) や Arm 社が主導する PSA (Platform Security Architecture) Certificate などで TEE，信頼の基点，リモートアテステーションに関係する仕様や認証制度が計画されている．インターネットのプロトコルを決める IETF においても，TEE 内のアプリケーションを管理するプロトコルである TEEP (Trusted Execution Environment Provisioning) やリモートアテステーションに関する RATS (Remote ATtestation ProcedureS) が議論されている．

9. 課　　題

クラウドでは仮想化技術を使うことがデフォルトであるが，TEE に対する対応は不十分である．仮想化の課題として，(1) TEE 自体を仮想化する技術と (2) 既存の仮想化から TEE を使うための技術に分けられる．(1) に関しては TEE 自体を仮想化して一つの CPU で複数の TEE を利用可能にする vTZ [61]や TEEv [62]などの拡張可能性が示されている．これらの技術を活用すれば，一つの Secure World しか使えなかった Arm TrustZone でも複数の Secure World が活用可能になり，異なる Trusted OS を実行できる．更に ARM v8.4A では TrustZone の仮想化対応が発表されており[63]，将来的に仮想化による複数の Trusted OS が TEE 内に実行可能になる．(2) に関しては TEE は Xen, KVM 及び Docker コンテナから利用可能であるが，VMware は不可となっている．これらは一つの VM から使うことを想定しており，クラウド環境のマルチテナントで使うためにはそのセキュリティやスケジューリングを考慮する必要がある．また，(1)，(2) ともにマイグレーションも想定する必要がある．既に VM マイグレーションは AMD SEV では想定しており，MigSGX [64]ではコンテナ内の SGX Enclave もマイグレーション可能にする．しかし，制約も多く，今後の課題である．

10. お わ り に

筆者が所属するセキュアオープンアーキテクチャ・エッジ基盤技術研究組合においても RISC-V をベースとする TEE の研究を行っている．信頼の基点やリモートアテステーションの機能を付加し，国内でもセキュリティ基盤となるような開発を行っている．今後は広く成果を公開して，その技術が活用されることを目指す．

謝辞　この成果は，国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）の委託業務（JPNP16007）「セキュアオープンアーキテクチャ基盤技術とその AI エッジ応用研究開発」の結果得られたものです．また，数々の議論に参加して多くのコメントを頂いた塚本明氏 (産総研)，中嶋健太氏 (TRASIO)，大居司氏 (TRASIO) に感謝します．

文　　献

(1) F. Zhang and H. Zhang, “SoK: A Study of Using Hardware-assisted Isolated Execution Environments for Security,” Proceedings of the Hardware and Architectural Support for Security and Privacy 2016 (HASP), 2016.
(2) tboot, https://sourceforge.net/projects/tboot/, 2007.
(3) J.M. McCune, B.J. Parno, A. Perrig, M.K. Reiter, and H. Isozaki, “Flicker: An Execution Infrastructure for TCB Minimization,” European Conference on Computer Systems (EuroSys), 2008.
(4) 須崎有康，塚本明，小島一元，中嶋健太，T.T. Hoang，師尾彬，“TEE 比較，” 暗号と情報セキュリティシンポジウム (SCIS), 2020.
(5) 須崎有康，佐々木貴之，“Trusted Execution Environment によるシステムの堅牢化，” 情報処理，vol. 61, no. 6, 2020.
(6) GlobalPlatform, Introduction to Trusted Execution Environments, 2018.
(7) S. Pinto and N. Santos, “Demystifying Arm TrustZone: A Comprehensive Survey,” ACM Computing Surveys (CSUR), vol. 51, no. 6, 2019.
(8) D. Cerdeira, N. Santos, P. Fonseca, and S. Pinto, “SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-assisted TEE Systems,” IEEE Symposium on Security and Privacy (IEEE S&P), 2020.
(9) OP-TEE, https://github.com/OP-TEE, 2017.
(10) OP-TEE Documentation, https://optee.readthedocs.io/, 2019.
(11) OpenTEE, https://open-tee.github.io/, 2015.
(12) B. McGillion, T. Dettenborn, T. Nyman, and N. Asokan, “Open-TEE –An Open Virtual Trusted Exe-

cution Environment," IEEE TrustCom, 2015.
(13) Trusty, https://open-tee.github.io/, 2016.
(14) QTEE, https://www.qualcomm.com/solutions/mobile-computing/features/security, 2016.
(15) KNOX, https://www.samsungknox.com/en, 2013.
(16) Teegris, http://developer.samsung.com/teegris, 2017.
(17) TrustKernel, https://www.trustkernel.com/en/, 2017.
(18) D. Shen, Attacking your Trusted Core, BlackHat USA, 2015.
(19) A. Baumann, "Hardware is the new Software," 16th Workshop on Hot Topics in Operating Systems (HotOS), 2017.
(20) V. Costan, I. Lebedev, and S. Devadas, "Secure Processors Part I: Background, Taxonomy for Secure Enclaves and Intel SGX architecture," Foundations and Trends in Electronic Design Automation, vol. 11, no. 1-2, pp. 1–248, 2017.
(21) V. Costan, I. Lebedev, and S. Devadas, "Secure Processors Part II: Intel SGX Security Analysis and MIT Sanctum Architecture," Foundations and Trends in Electronic Design Automation, vol. 11, no. 3, pp. 249–361, 2017.
(22) OpenEnclave, https://openenclave.io/, 2019.
(23) J. Lind, C. Priebe, D. Muthukumaran, D. O'Keeffe, P.-L. Aublin, F. Kelbert, T. Reiher, D. Goltzsche, D. Eyers, R. Kapitza, et al., "Glamdring: Automatic Application Partitioning for Intel SGX," USENIX Annual Technical Conference (USENIX ATC), 2017.
(24) S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'Keeffe, M.L. Stillwell, et al., "SCONE: Secure Linux Containers with Intel SGX," Symposium on Operating Systems Design and Implementation (OSDI), 2016.
(25) F. Schuster, M. Costa, C. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz, and M. Russinovich, "VC3: Trustworthy Data Analytics in the Cloud using SGX," IEEE Symposium on Security and Privacy (IEEE S&P), 2015.
(26) A. Baumann, M. Peinado, and G. Hunt, "Shielding Applications from an Untrusted Cloud with Haven," Symposium on Operating Systems Design and Implementation (OSDI), 2016.
(27) A. Baumann, M. Peinado, and G. Hunt, "Shielding Applications from an Untrusted Cloud with Haven," ACM Transactions on Computer Systems (TOCS), vol. 33, no. 3, pp. 1–15, 2015.
(28) C. Priebe, D. Muthukumaran, J. Lind, H. Zhu, S. Cui, V.A. Sartakov, and P. Pietzuch, "SGX-LKL: Securing the Host OS Interface for Trusted Execution," arXiv:1908.11143, 2019.
(29) C.-C. Tsai, D.E. Porter, and M. Vij, "Graphene-SGX: A Practical Library OS for Unmodified Applications on SGX," USENIX Annual Technical Conference (USENIX ATC), 2017.
(30) Enarx, https://enarx.io/, 2019.
(31) V. Costan, I. Lebedev, and S. Devadas, "Sanctum: Minimal Hardware Extensions for Strong Software Isolation," USENIX Security Symposium (USENIX Sec), 2016.
(32) T. Bourgeat, I. Lebedev, A. Wright, S. Zhang, S. Devadas, et al., "MI6: Secure Enclaves in a Speculative Out-of-Order Processor," IEEE/ACM International Symposium on Microarchitecture (MICRO), 2019.
(33) D. Lee, D. Kohlbrenner, S. Shinde, D. Song, and K. Asanović, "Keystone: An Open Framework for Architecting TEEs," arXiv:1907.10119, 2019.
(34) HexFive, https://hex-five.com, 2018.
(35) S. Weiser, M. Werner, F. Brasser, M. Malenko, S. Mangard, and A.-R. Sadeghi, "TIMBER-V: Tag-Isolated Memory Bringing Fine-grained Enclaves to RISC-V," Network and Distributed System Security Symposium (NDSS), 2019.
(36) G. Klein, K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, et al., "seL4: Formal Verification of an OS Kernel," Symposium on Operating Systems Principles (SOSP), 2009.
(37) GlobalPlatform, Root of Trust Definition and Requirements version 1.1, 2018.
(38) GlobalPlatform, TEE Secure Element API Version 1.1.1, https://globalplatform.org/specs-library/tee-secure-element-api-v1-1-1/, 2016.
(39) Rambus, Qualcomm Licenses Rambus CryptoManager Key and Feature Management Security Solution, https://www.rambus.com/qualcomm-licenses-rambus-cryptomanager-key-and-feature-management-security-solution/, 2014.
(40) N. Asokan and A. Paverd, "Remote Attestation — Building trust in things you can't see —," Tutorial of AsiaCCS, 2017.
(41) J. Camenisch, "Direct Anonymous Attestation Revisited," https://researcher.watson.ibm.com/researcher/files/zurich-JCA/2017-01-10-Direct-Anonymous-Attestation.pdf, 2017.
(42) Intel, Local (Intra-Platform) Attestation, https://software.intel.com/en-us/node/702983, 2016.
(43) G. Chen, Y. Zhang, and T.-H. Lai, "OPERA: Open Remote Attestation for Intel's Secure Enclaves," Conference on Computer and Communications Security (CCS), 2019.
(44) N. Weichbrodt, P.-L. Aublin, and R. Kapitza, "sgxperf: A Performance Analysis Tool for Intel SGX Enclaves," International Middleware Conference (Middleware), 2018.
(45) M. Bailleu, D. Dragoti, P. Bhatotia, and C. Fetzer, "TEE-Perf: A Profiler for Trusted Execution Environments," Dependable Systems and Networks (DSN), 2019.
(46) T.T.A. Dinh, P. Saxena, E.-C. Chang, B.C. Ooi, and C. Zhang, "M2R: Enabling Stronger Privacy in MapReduce Computation," USENIX Security Symposium (USENIX Sec), 2015.
(47) R. Kunkel, D.L. Quoc, F. Gregor, S. Arnautov, P. Bhatotia, and C. Fetzer, "TensorSCONE: a Secure Tensorflow Framework using Intel SGX," arXiv:1902.04413, 2019.
(48) C. Kockan, K. Zhu, N. Dokmai, N. Karpov, M.O. Kulekci, D.P. Woodruff, and S.C. Sahinalp, "Sketching algorithms for genomic data analysis and querying in a secure enclave," Nature Methods, vol. 17, no. 3, pp. 295–301, 2020.
(49) 岩田大輝，清水佳奈，"Intel SGX を用いた個人ゲノム情報解析システム，" 暗号と情報セキュリティシンポジウム (SCIS), 2020.
(50) 中野智晴，光来健一，"Intel SGX を用いた VM のメモリとディスクの安全な監視，" コンピュータセキュリティシンポジウム (CSS), 2019.
(51) P.-L. Aublin, F. Kelbert, D. O'Keeffe, D. Muthukumaran, C. Priebe, J. Lind, R. Krahn, C. Fetzer, D. Eyers, and P. Pietzuch, "LibSEAL: Revealing Service Integrity Violations Using Trusted Execution," European Conference on Computer Systems (EuroSys), 2018.
(52) S. Matetic, M. Schneider, A. Miller, A. Juels, and S. Capkun, "DelegaTEE: Brokered Delegation Using Trusted Execution Environments," USENIX Security Symposium (USENIX Sec), 2018.
(53) D. Goltzsche, M. Nieke, T. Knauth, and R. Kapitza, "AccTEE: A WebAssembly-based Two-way Sandbox for Trusted Resource Accounting," International Middleware Conference (Middleware), 2019.
(54) B. Fisch, D. Vinayagamurthy, D. Boneh, and S. Gorbunov, "Iron: Functional Encryption using Intel

SGX," Conference on Computer and Communications Security (CCS), 2017.
(55) 鈴木達也，江村恵太，面和成，大東俊博，"Intel SGX を用いた公開検証可能な関数型暗号の構成と実装評価," 暗号と情報セキュリティシンポジウム (SCIS)，2020.
(56) M. Schwarz, S. Weiser, D. Gruss, C. Maurice, and S. Mangard, "Malware Guard Extension: Using SGX to Conceal Cache Attacks," Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), 2017.
(57) J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T.F. Wenisch, Y. Yarom, and R. Strackx, "Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution," USENIX Security Symposium (USENIX Sec), 2018.
(58) A. Gonzalez, B. Korpan, J. Zhao, E. Younis, and K. Asanović, "Replicating and Mitigating Spectre Attacks on an Open Source RISC-V Microarchitecture," Workshop on Computer Architecture Research with RISC-V (CARRV), 2019.
(59) A. Machiry, E. Gustafson, C. Spensky, C. Salls, N. Stephens, R. Wang, A. Bianchi, Y.R. Choe, C. Kruegel, and G. Vigna, "BOOMERANG: Exploiting the Semantic Gap in Trusted Execution Environments," Network and Distributed System Security Symposium (NDSS), 2017.
(60) J. Van Bulck, D. Oswald, E. Marin, A. Aldoseri, F.D. Garcia, and F. Piessens, "A Tale of Two Worlds: Assessing the Vulnerability of Enclave Shielding Runtimes," Computer and Communications Security (CCS), 2019.
(61) Z. Hua, J. Gu, Y. Xia, H. Chen, B. Zang, and H. Guan, "vTZ: Virtualizing ARM TrustZone," USENIX Security Symposium (USENIX Sec), 2017.
(62) W. Li, Y. Xia, L. Lu, H. Chen, and B. Zang, "TEEv: Virtualizing Trusted Execution Environments on mobile platforms," Virtual Execution Environments (VEE), 2019.
(63) D. Banks, "Arm TrustZone with Secure Partitions and Armv8.4," Open Source Enclave Workshop (OSEW), 2019.
(64) 中島健児，光来健一，"Intel SGX を利用するコンテナのマイグレーション機構," コンピュータセキュリティシンポジウム (CSS)，2018.

（HWS 研究会提案，2020 年 6 月 15 日受付，
2020 年 7 月 10 日再受付）

須崎有康（正員）

1991 東京農工大学電子情報工学科博士後期課程中退．電子技術総合研究を経て，産業技術総合研究所で OS 及びセキュリティの研究に従事．2019 よりセキュアオープンアーキテクチャ・エッジ基盤技術研究組合の研究員．情報理工学博士 (東京大学)．