9. Remote Attestation②

Ao Sakurai

2026年度セキュリティキャンプ全国大会
L1 – 暗号・CVMビルド&スクラップゼミ

本セクションの目標

- TDXのRAについて、より詳細な仕様の説明を行っていく

RAフローのおさらい

- TDXにおけるRAのフローを簡単に処理順に以下に示す：
  - [Attester] ロード時におけるTD及びTCBの測定
  - [Attester] プラットフォーム登録
  - [Attester] TDQE AK（Attestation Key）のプロビジョニング
  - [Attester] TD Reportの生成
  - [Attester] TDQEとAttester TDとのLocal Attestation（LA）の実施、TD Quoteの生成
  - [Verifier] TD Quoteの検証
  - [Relying Party] RAの受理判断

- これらの処理において主にTDX固有の説明すべき具体的な要素について、この処理の流れに沿う形で順番に説明していく

TD及びTCBの測定

TDXにおける測定レジスタ

- TDはその構築（TDイメージからのロード）時、TDX Moduleによりその初期メモリ配置を測定される

- さらに、TDが立ち上がった後に、ランタイムコンポネントの測定に用いる事ができる測定レジスタも存在する

- また、TDX Module自身も、ロードされる際にTDX Moduleのロードを行うP-SEAM Loaderにより測定される[2]

TDXにおける測定レジスタ

- TDX Moduleは、TDの初期メモリ配置の測定レジスタとして**MRTD**と呼ばれるものをTDCS内で管理する
  - **MRTD**: Measurement of Trust Domain
- 一方、ランタイムコンポネントの測定レジスタとして4つの**RTMR**というものを同じくTDCS内で管理する
  - **RTMR**: Runtime Measurement Register
- また、TDX Moduleの動作定義に対するハッシュ値は**MRSEAM**としてCPU内で管理される
  - 参考文献[3]の図12.3を参照

TDXにおける測定レジスタ

- TDXがTDにロードしてくれるのは、初期メモリ配置に相当するファームウェア（FW）の静的部分だけである

- それ以降はFWがOSローダを測定、OSカーネルがユーザワークロードを測定し、といった具合に、より**上位のコンポネントが次のコンポネントを測定**する
  - このような測定を連鎖させながらの起動の事を**Measured Boot**と呼ぶ

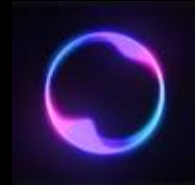
- この初期メモリ配置分をMRTD、それ以降を4つのRTMRがその測定値を持つ事で、上手く**測定値保管の分担**をしている
  - ちなみに、SGXではEnclaveという初期メモリ配置で全てが決まったのでMRENCLAVEという測定値1つでコード動作定義に関しては表せた

TDXにおける測定レジスタ

- MRTDもRTMRもMeasured Bootの一種として使われるものであるため、TPMにおける**プラットフォーム構成レジスタ（PCR）**と同等の立ち位置にあると言える

- PCRは、TPMにおいて各種測定値を格納するための測定レジスタであり、かなりMRTDやRTMRに近い

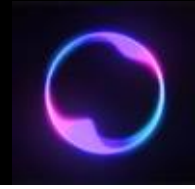
TD測定値とTPM PCRの比較



- MRTDやRTMRが実際にどのような測定値を格納するのかを、TPM PCRと比較しながら以下の表に概観レベルで示す：

TDX測定レジスタ	TPM PCR	用途
MRTD	PCR[0]	仮想FW（ファームウェア）
RTMR[0]	PCR[1, 7]	仮想FWのデータ、仮想FWの設定
RTMR[1]	PCR[2-5]	OSカーネル、initrd、起動変数
RTMR[2]	PCR[8-15]	OSアプリケーション
RTMR[3]	-	予約領域

TD測定値とTPM PCRの比較



- さらに詳細な対応表を、Intel公式文書[4]の情報をもとにアレンジして以下に示す：

PCR番号	主な用途	TDレジスタ	Extendする主体
0	FWコード（Boot Firmware Volume; BFV、これにはinit page tableも含む）	MRTD	VMM。 MRTDなので、これのみSEAMCALLのTDH.MR.EXTENDで拡張
1	FWデータ（Configuration Firmware Volume; CFV、TD HOB、ACPIテーブル） ※TD HOB：VMMからTDVFに情報を渡す用の構造。TD Hand-off Block。	RTMR[0]	TDVF。 以下、RTMRは全てTDCALLのTDG.MR.RTMR.EXTENDで拡張
2	追加的なROMコード。 NICのようなPCIオプションROM等。	RTMR[1]	TDVF
3	追加的なROMデータ	RTMR[1]	TDVF
4	OSローダコード	RTMR[1]	TDVF
5	構成設定。GPTや起動変数等	RTMR[1]	TDVF
7	Secure Boot設定。CFVに含まれている	RTMR[0]	TDVF
8～15	TDゲストOS測定値（IMA対象含む）	RTMR[2]	TD OSローダまたはTD OS

TDのロードと測定

- 実際にこれらのMRTDやRTMRを用いてどのようにTDを測定しているのかを、具体的なロード処理の流れに沿う形で説明する

- まず、HVは**SEAMCALL[TDH.MEM.PAGE.ADD]**を呼び出す

- これによりTDイメージからTDのメモリにページが追加され、さらに**ページアドレス情報に関して**の測定が実施される
  - 具体的には、ASCII文字列「MEM.PAGE.ADD」及びページのGPA（ゲスト物理アドレス）に対するSHA384 Updateを計算しMRTDをExtendする
  - この処理により、そのページはSEPTで利用可能となる

TDのロードと測定

- 次に、HVは**SEAMCALL[TDH.MR.EXTEND]**を複数回呼び出し、**ページコンテンツについて**の測定を実施する
  - このページの測定はは256Bサイズの「ブロック」単位で実施される
- 具体的にはまず、ASCII文字列「MR.EXTEND」及びブロックのGPAに対するSHA384 Updateを計算する
- 次に、そのブロックのコンテンツに対する別のSHA384 Updateを計算する
- そして、これら2つのハッシュ値によりMRTDをExtendする

TDのロードと測定

- TDH.MEM.PAGE.ADDもTDH.MR.EXTENDも双方共にハッシュを計算しており、EXTEND側は何ならさらに2つのハッシュを計算しているためややこしさが否めない

- しかし、**ADD**と**EXTEND**ではそれぞれ別のものを主に測定している（前者は**ページアドレス情報**、後者は**ページコンテンツ**）事に注意するべきである

TDのロードと測定

- 具体的な実装の例で言えば、例えばLinuxのHVであるKVMは、このTDCALLを通してTDVF（TDの仮想FW）コードの測定値をMRTDにExtendする

- ちなみに、TDR、TDCS、TDVPS、SEPTのような**制御構造**はこの測定対象には**入らない**

TDのロードと測定

- TDの初期メモリ配置分全体に対するページ一式を追加及び測定し終えたら、HVは**SEAMCALL[TDH.MR.FINALIZE]**を実行する事でMRTD測定を確定させる
  - この初期化後はMRTDをExtendさせるような全ての操作が無効化される

- FINALIZEによりMRTDはロックされてしまうため、以降のアッパーレイヤーのコンポネント（FWデータ、OSカーネル、ユーザワークロード等）の測定は、代わりに**RTMR**を用いる事によって行う

- RTMRは0～3のラベル付けがされた4つの汎用測定レジスタであり、初めは0に初期化されている

TDのロードと測定

- TDは、**TDCALL[TDG.MR.RTMR.EXTEND]**を呼び出す事によりRTMRのコンテンツをExtendする事ができ、この命令（リーフ関数）の引数は以下の2つである：
  - 測定レジスタのインデックス（0～3）
  - 拡張するハッシュ値を含む48Bのバッファに対する、64B単位でアラインされた物理アドレス

- 以上から、あるindexに対し、新たな測定値であるvalueを拡張する場合、以下のような式で表す事ができる：
  - $RTMR[index] = \mathrm{SHA384}(RTMR[index]||value)$

TD Reportの生成

TD Reportの生成

- プラットフォーム登録とプロビジョニングについては前回スライドにて十分に説明したため、続いてはTD Report作成に関する詳細について説明する

- TD ReportはTDX Moduleが提供する**TDCALL[TDG.MR.REPORT]**をTDが呼び出す事により生成を要求する事ができる
  - この時、新しく初期化されたReport構造体と、64BのReport Dataを入力として提供する

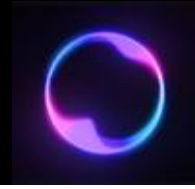
- Report Dataは、以前説明の通り鮮度（Freshness）保証、暗号通信路確立、特定のデータの完全性保証等、様々な用途に用いる事ができる

TD Reportの生成

- 上記のTDCALLによりTDX Moduleに制御が移ったら、TDX ModuleはTDの測定値とReport Dataを携えて**SEAMOPS[SEAMREPORT]**命令を発行する
  - SEAMOPS：SEAM VMX rootモードの主体によってのみ実行される、SEAM特有の処理を行う命令。簡単に言えば、TDX Module等が内部で呼び出すコア中のコアの命令[5]

- ただし、もしそのマシンが**TD-Preserving**機能に対応している場合は、代わりに**SEAMOPS[SEAMDB_REPORT]**を発行する
  - **TD-Preserving**: TDX Moduleを含むTCBを、TDの状態を維持したままでアップデートする事のできる機能

TD Reportの生成手順の難解な説明



- TD Reportの具体的な生成方法について、敢えてIntel公式文書[3]の内容をもとに説明してみる：

1. CPUは、SEAMについての測定値であるSEAM Reportを生成する。SEAM Reportには、SEAMについての同一性情報（**TEE_TCB_INFO**）の**SHA384ハッシュ値**が含まれ[5]、これは**TEE_TCB_INFO_HASH**と呼ばれる[3]
   - 参考文献[5]等も見ると、TDに固有な情報であるTD Info（**TD_INFO**または**TEE_INFO**、あるいは**TDINFO_STRUCT**）も取り扱え、SHA384ハッシュ値という形でReportに同梱できる[5]。このハッシュ値は**TD_INFO_HASH**または**TEE_INFO_HASH**と呼ばれる[3][5]
   - 同じく[5]曰く、SEAMOPS[SEAMREPORT]命令に、TEE_INFO_HASHを引数で指定できる。よって、TDX ModuleがSHA384を取り、それをSEAMOPSの入力とする事で、TD Infoを同梱させる事もできる。同梱させない場合、当該引数を0埋めして呼び出せば良い

TD Reportの生成手順の難解な説明

2. 上記ハッシュを含む**REPORTMACSTRUCT**構造体に、256bitの**Report Key**[5]を用いてHMACを付与する[3][5]
   - このREPORTMACSTRUCT構造体にTEE_TCB_INFO構造体を連結したものを**SEAMREPORT構造体**と呼ぶ

3. TDX Moduleに制御が戻り、TDX Moduleはさらにその SEAMREPORT構造体をパースし、最終生成物である **TDREPORT_STRUCT**（TD Report）を構築してTDに返す

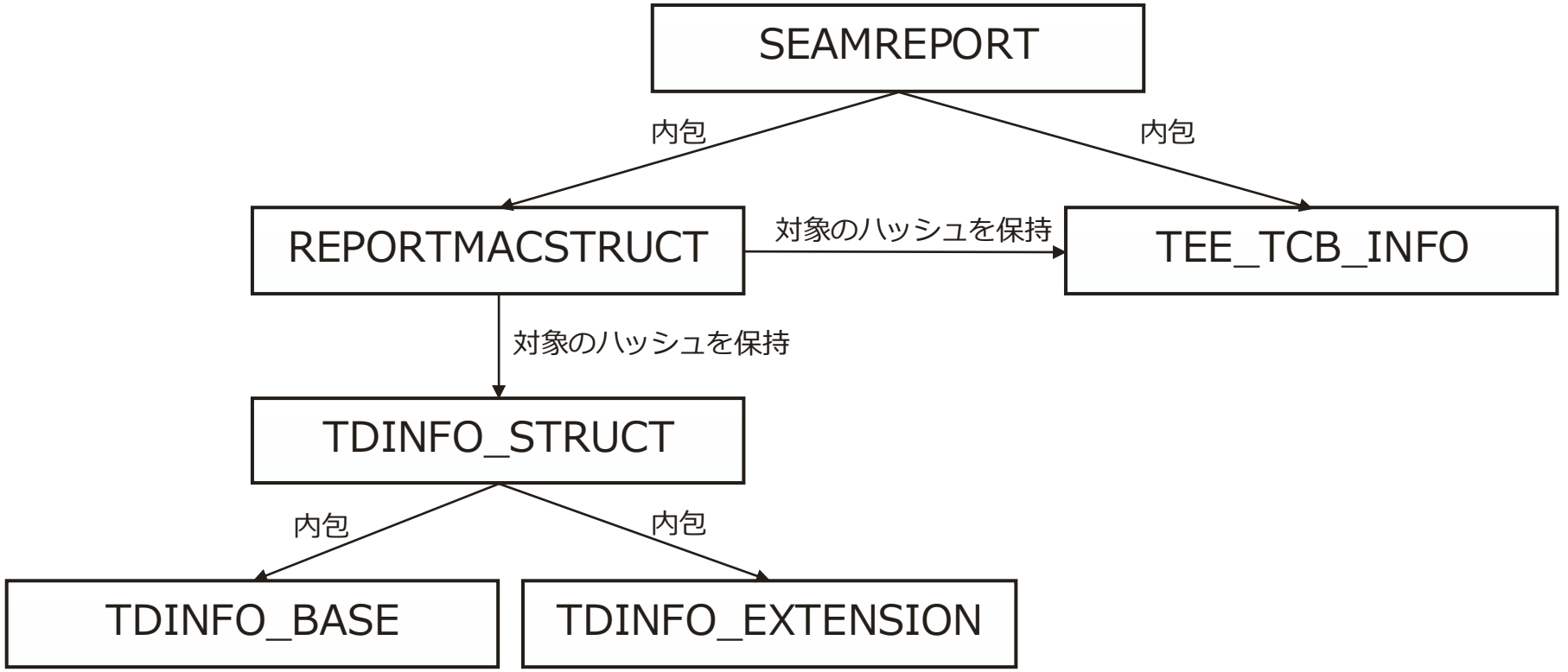
TD Reportの生成手順の難解な説明

- ではこの説明を見てTD Reportの作り方が分かったか？
  - 少なくとも筆者は**こんなものを見せられても理解できない**

- よって、このTD Report周りに関してはより分かりやすく理解できるようにもう少し掘り下げて整理する

TD Reportの生成手順のより分かりやすい説明

- まずはTD Reportの内、SEAMOPS[SEAMREPORT]が生成するパーツであるSEAMREPORT構造体における各コンポネントの関係図を以下に示す：



TD Reportの生成手順のより分かりやすい説明

- まず、**SEAMOPS[SEAMREPORT]**の結果返ってくるものが**SEAMREPORT構造体**である
  - リーフ関数名と名前が同じだが混乱しないようにする必要がある
- 次に、このSEAMREPORTは**REPORTMACSTRUCT**と**TEE_TCB_INFO**を内包する
  - 前者はTDの測定値、後者はSEAM（TDX Module）の測定値である
- さらに、REPORTMACSTRUCTは、後者のTEE_TCB_INFOに対するハッシュ値である**TEE_TCB_INFO_HASH**を内包する

TD Reportの生成手順のより分かりやすい説明

- ところでSEAMOPS[SEAMREPORT]発行の際、TDの測定値を内包した、TDX ModuleによるTD測定である**TEE_INFO**に対するハッシュ値を引数として渡せる
  - 困ったことにTEE_INFOは**TD_INFO**や**TDINFO_STRUCT**とも呼ばれる
- これにより、TEE_INFOのハッシュ値をREPORTMACSTRUCTに関連付ける事ができる
  - つまりはREPORTMACSTRUCTはTEE_INFOのハッシュ値を持つ
  - ただしTEE_INFO自体を含むわけではない点に注意
- このTEE_INFOには、TDの基本的な測定値を内包する**TDINFO_BASE**と、拡張的な測定値を内包する**TDINFO_EXTENSION**が含まれる

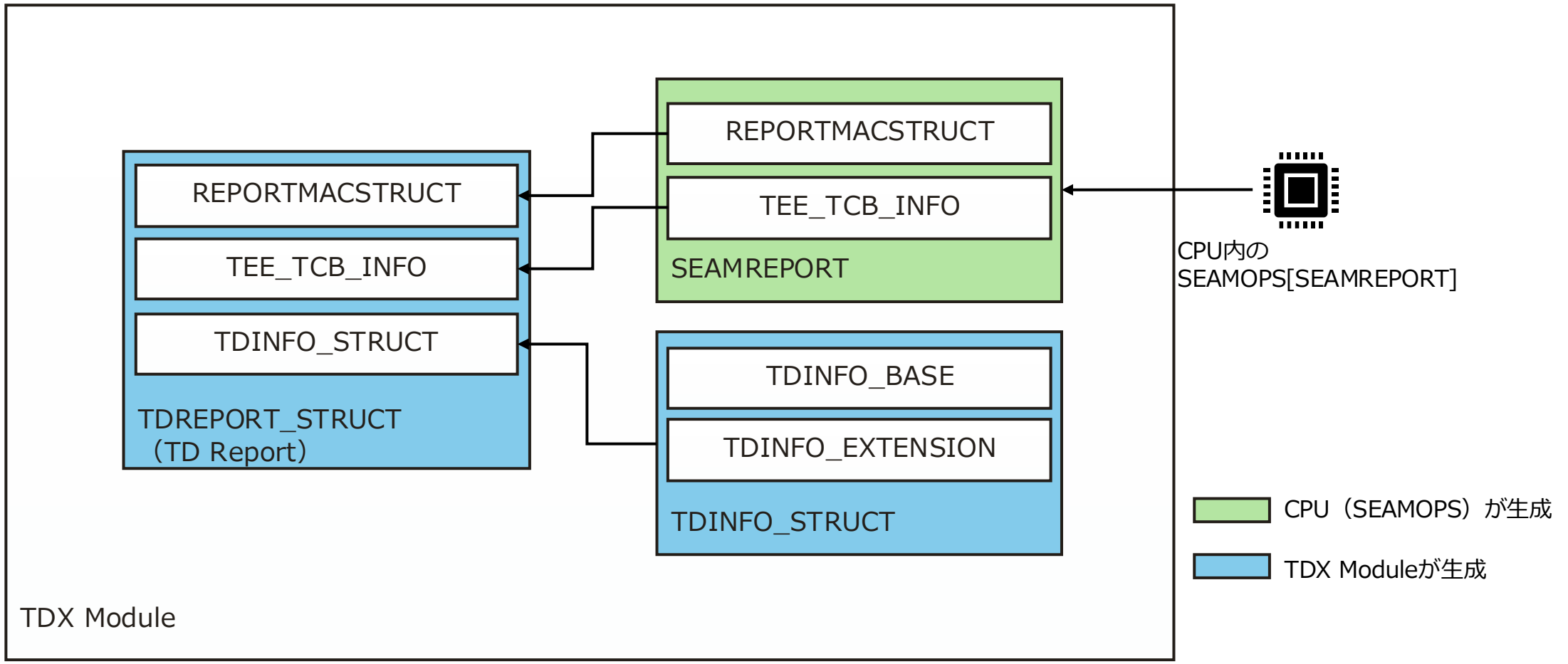
TD Reportの生成手順のより分かりやすい説明

- TDX Moduleは、SEAMOPS[SEAMREPORT]の結果としてこのようなSEAMREPORT構造体を受け取る

- その後、TDX ModuleはSEAMREPORT構造体と、TDX Module管理のTEE_INFO構造体本体から、以下の情報をパースしそれらで構築した最終生成物**TDREPORT_STRUCT**(**TD Report**) を生成する[6] :
  - REPORTMACSTRUCT（Report Keyによる署名付き）
  - TEE_TCB_INFO（REPORTMACSTRUCTがこれのハッシュを持つ）
  - TDINFO_STRUCT（TEE_INFO ; 同じくREPORTMACSTRUCTがこれのハッシュを持つ）

TD Reportの生成手順のより分かりやすい説明

- このTDREPORT_STRUCTの構成を図に表すと以下のようになる：

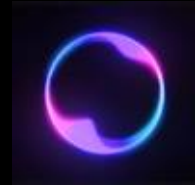


TD Reportの生成手順のより分かりやすい説明

- こうしてみると、最終的なTD Reportは以下の2つから生成しているだけであり、字面よりは分かりやすい（各種ハッシュ値の同梱等はややこしいが）：
  - SEAMOPS[SEAMREPORT]が生成したSEAMREPORT構造体。メインはTDX Moduleの測定値
  - TDX Moduleが管理するTDINFO_STRUCT。メインはTDの測定値

- 後は、生成したTD ReportをTDに返却して、次のQuoteの生成作業に入る事ができる

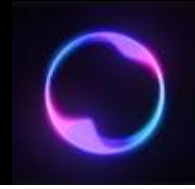
TDREPORT_STRUCTの構造



- まずは最終生成物であるTDREPORT_STRUCT（TD Report）の内訳を以下に示す[6]:

メンバ	オフセット（バイト）	サイズ（バイト）	説明
REPORTMACSTRUCT	0	256	REPORTMACSTRUCT構造体。
TEE_TCB_INFO	256	239	REPORTMACSTRUCT構造体内のTEE_TCB_INFO_HASHに紐づいているTEE_TCB_INFO構造体。
RESERVED	495	17	予約領域。オール0
TDINFO	512	(説明を参照)	別名TEE_INFOやTDINFO_STRUCT。後述の通り、バージョンによりサイズは変わってくる

SEAMREPORT構造体の構造



- 次は、SEAMOPS[SEAMREPORT]が生成し、その中身がTD Reportの一部として使われるSEAMREPORT構造体の構成を以下に示す[5]

メンバ	オフセット（バイト）	サイズ（バイト）	説明
REPORTMACSTRUCT	0	256	REPORTMACSTRUCT構造体。
TEE_TCB_INFO	256	239	REPORTMACSTRUCT構造体内のTEE_TCB_INFO_HASHに紐づいているTEE_TCB_INFO構造体。

REPORTMACSTRUCTの構造

- REPORTMACSTRUCTの構造は以下の通り[5]：

メンバ	オフセット（バイト）	サイズ（バイト）	説明
REPORTTYPE	0	4	TEEタイプを格納する構造体（内訳の詳細は省略）。そのTEEがSGXかTDXかを示す。
RESERVED	4	12	予約領域。オール0必須。
CPUSVN	16	16	CPUSVN。TCBを構成する16個のコンポネントそれぞれについてのセキュリティバージョン番号（SVN）を連結した128bitのバイト列。
TEE_TCB_INFO_HASH	32	48	TEE_TCB_INFOのSHA384ハッシュ。
TEE_INFO_HASH	80	48	TEE_INFO（TD_INFO）のSHA384ハッシュ。
REPORTDATA	128	64	Report Data。
RESERVED	192	32	予約領域。オール0必須。
MAC	224	32	上記7つのメンバの連結に対する、レポートキーにより計算された256bit HMAC[5]。

TEE_TCB_INFOの構造

- REPORTMACSTRUCTと共にSEAMREPORT構造体に含まれるTEE_TCB_INFO構造体の構成は以下の通り[5]：

メンバ	オフセット（バイト）	サイズ（バイト）	説明
VALID	0	8	TEE_TCB_INFO自体に対するビットマスク。上位i番目が1だと、8*iから始まる8バイトが有効。0だと無効。VALID自体は64**bit**サイズであるため、8*i, i=64により、この構造全体をカバーできている。
TEE_TCB_SVN	8	16	TDX ModuleのSVNと予約領域からなる構造体。
MRSEAM	24	48	TDX Moduleの測定値。
MRSIGNERSEAM	72	48	TDX Moduleの署名（者）に対する測定値。
ATTRIBUTES	120	8	属性情報。詳細は[6]参照。
TEE_TCB_SVN2	128	16	TD Preservingで使用されるTCBSVN。恐らく、更新前か後のSVNを格納するものと思われる。
RESERVED	144	95	予約領域。オール0必須。

TDINFO_STRUCTの構造

- TDX Moduleが生成し、TDREPORT_STRUCTの一部として格納されるTDINFO_STRUCT構造体は以下の通り[6]：

メンバ	オフセット（バイト）	サイズ（バイト）	説明
TDINFO_BASE	0	448	下記参照
TDINFO_EXTENSION	448	64または320	REPORTMACSTRUCT.REPORTTYPE.VERSIONが0または1の場合は64バイトで、オール0の予約領域。2の場合は320バイトで、後述参照。TDREPORT_STRUCTの説明にてTDINFOはバージョンによりサイズが異なると書いたが、このTDINFO_EXTENSIONの存在の有無が関与している

TDINFO_BASEの構造

- TDINFO_STRUCTに含まれる構造体の1つ。ATTRIBUTESとXFAM以外は全てSHA384ハッシュ値の形を取る

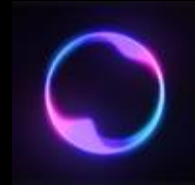
メンバ	オフセット（バイト）	サイズ（バイト）	説明
ATTRIBUTES	0	8	TDの属性情報。構造体としてはTEE_TCB_INFO内のそれと同一なはず。
XFAM	8	8	Extended Features Available Mask。HVからTD初期化時に入力されるもので、TDX Connect含む様々な機能の有効化状況等が同梱されている[6]。
MRTD	16	48	そのTDのMRTD。
MRCONFIGID	64	48	TD所有者以外による設定についての、SW定義のID。例えばランタイムコンポネントやOS設定に関するもの等。
MROWNER	112	48	TD所有者についてのSW定義のID。
MROWNERCONFIG	160	48	TD所有者による設定についてのSW定義のID。例えば、TD所有者がデプロイしたワークロードに関する設定等。

TDINFO_BASEの構造

- 前ページの続き：

メンバ	オフセット（バイト）	サイズ（バイト）	説明
RTMR	208	4 * 48	そのTDのRTMR。RTMR0～3全てが連結された状態で格納されている。
SERVTD_HASH	400	48	サービスTDがバインドされているかバインド事前処理状態（pre-bound）である場合は、バインドされたサービスTDのTDINFO_STRUCTのハッシュ値。それ以外の場合は0。

TDINFO_EXTENSIONの構造



- TDINFO_EXTENSIONの構造は以下の通り：

メンバ	オフセット（バイト）	サイズ（バイト）	説明
MRSIGROOT	0	48	MRSIGNERのSigner Rootに対する測定値。詳細は不明だが、署名鍵をPKIベースで管理し、その際のルートCA秘密鍵に対するハッシュ値として使用できる？SHA384ハッシュ値の形を取る。
MRSIGNER	48	48	TDSIGSTRUCTに対する署名（者）の測定値。SHA384ハッシュ値の形を取る。
PRODID	96	16	そのTDのプロダクトID。SGXにおけるISV Prod IDと同等と思われる。16バイト整数。
ISVSVN	112	2	そのTDのSVN。SGXにおけるISVSVNと同等と思われる。2バイト整数。
MRCONFIGSVN	114	2	MRCONFIGIDについてのSVN。
MROWNERCONFIGSVN	116	2	MROWNERCONFIGについてのSVN。
RESERVED	118	202	予約領域。オール0必須。

TDINFO_EXTENSIONの構造

- MRSIGROOT～ISVSVNの4つは**TDCALL[TDG.MR.ASSIGNSVNS]**によりTD側からExtendされる[3]

- MRCONFIGSVNとMROWNERCONFIGSVNは、**SEAMCALL[TDH.MNG.INIT]**時にTD_PARAMS経由で渡される[3]
  - この命令自体は、TD毎の制御構造を初期化する関数である[3]

TDSIGSTRUCT

- TDINFO_EXTENSIONの説明で登場した**TDSIGSTRUCT**は、TDイメージの作成者の署名情報と、そのTDに求めるリファレンス測定値が同梱されている構造体である

- TDSIGSTRUCT内のリファレンス測定値は、前述のTDCALL[TDG.MR.ASSIGNSVNS]をTD起動時にゲスト側から呼び出された際等に、実際のそのTDの測定値との比較が行われる[6]

TDSIGSTRUCT

- ただし、署名者情報に関してはTDイメージに直接署名がついているわけではない

- あくまでもこのTDSIGSTRUCTに署名がついており、そしてTDSIGSTRUCTをTDイメージに含める等をしておく

- そして、TDCALL[TDG.MR.ASSIGNSVNS]時にそれをロードしたゲスト物理アドレスを渡す事でチェックを行う事ができる[6]

TDSIGSTRUCT

- TDSIGSTRUCTの構造は以下の通り[6]：

メンバ	オフセット（バイト）	サイズ（バイト）	説明	署名対象か
HEADERTYPE	0	4	「モジュール」タイプ。2026/3現在では6である必要がある。「モジュール」については後述のMODVENDORの説明を参照。	はい
HEADERLEN	4	4	「暗号的フィールドを含むヘッダ長」とされており、2026/3現在では0x00E1である必要があるようだが、詳細は不明。	はい
HEADERVERSION	8	4	TDSIGSTRUCT構造体のバージョン。2026/3現在では0x00010000である必要がある。	はい
TYPE	12	4	bit31のみソフトウェアから何らかの目的で利用可能とされている。bit30-0はオール0でなければならない。	はい

TDSIGSTRUCT

- 前ページの続き：

メンバ	オフセット（バイト）	サイズ（バイト）	説明	署名対象か
MODVENDOR	16	4	「モジュールベンダ」と書かれており、要はTDSIGSTRUCTを生成する何らかのツール（「モジュール」）を作成したベンダの事と思われるが詳細不明。恐らくTDX Moduleを指しているものではない。Intelの場合は0x8086、その他ISV（ベンダ）の場合は0x0000。	はい
DATE	20	4	yyyymmdd形式での、「ビルド時」の10進数値。TDのビルド時なのかTDSIGSTRUCTのビルド時なのかは不明。	はい
SIZE	24	4	「モジュール」のサイズ。0x0540でなければならないらしい。	はい
KEYSIZE	28	4	[6]にはRSA公開鍵のモジュラスのサイズで0x0060である必要があると書かれている。MODULUSメンバの中身を見るに、誤植ではない可能性が高い。	はい

TDSIGSTRUCT

- 前ページの続き：

メンバ	オフセット（バイト）	サイズ（バイト）	説明	署名対象か
MODULUSSIZE	32	4	RSA公開鍵のモジュラスのサイズで、0x0060でなければならない	はい
EXPONENTSIZE	36	4	RSA公開鍵の指数のサイズ。	はい
RESERVED	40	88	予約領域。オール0でなければならない	はい
MODULUS	128	384	[6]曰く、モジュラスではなく「モジュール」の公開鍵がそのまま入っている。3072bit。	いいえ
EXPONENT	512	4	RSAの指数。値は $2^{16}+1$で固定。	いいえ
SIGNATURE	516	384	TDSIGSTRUCT内の署名対象に対する署名。	いいえ

TDSIGSTRUCT

- 前ページの続き：

メンバ	オフセット（バイト）	サイズ（バイト）	説明	署名対象か
POLICY	900	2	どの測定値を、TDSIGSTRUCT内のリファレンス測定値と比較するかを指定するビット列。 Bit0: SIGSTRUCTタイプ。0で「リーフ（単一）SIGSTRUCT」、1で「チェーンSIGSTRUCT」らしい。詳細不明。 Bit1: MRTD。恐らく1で比較有効化。以下同様。 Bit2: RTMR0 Bit3: RTMR1 Bit4: RTMR2 Bit5: RTMR3 Bit15-6: 予約領域。オール0	はい
ISVSVN	902	2	そのTDイメージに割り当てたSVN。	はい
PRODID	904	16	そのTDイメージに割り当てたプロダクトID	はい

TDSIGSTRUCT

- 前ページの続き：

メンバ	オフセット（バイト）	サイズ（バイト）	説明	署名対象か
RESERVED	920	8	予約領域。オール0でなければならない	はい
MRTDREF	928	8	MRTDのリファレンス測定値	はい
RTMRREF	976	48*4	4つのMRTDのリファレンス測定値からなる配列	はい
MRSIGNER	1168	48	そのTDに対するMRSIGNER値	はい
RESERVED	1216	64	予約領域。オール0でなければならない	はい

TDINFO_EXTENSIONとTDSIGSTRUCTについて

- ただし2026/6現在、TDINFO_EXTENSIONやTDSIGSTRUCTについては、IntelのGitHub等からその**実装上の実体を発見できていない**

- よって、この二者に関しては今後実装されていくものであると推測される

LAの実施とTD Quoteの生成

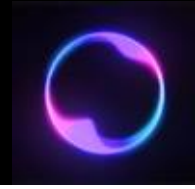
LAの実施とTD Quoteの生成

- TD Reportを生成したら、次はTD ReportをTDQEに転送し、TDQEはLA（Local Attestation）を実施する

- TDQEは、ENCLU[EVERIFYREPORT2]命令を用いる事でTD ReportをLA（同じマシンから来ているかを確認）する

- その後、アンシーリングした**AKで署名**する事で**TD Quoteを生成**する

LAの実施とTD Quoteの生成



- より実環境上での実践的な操作の話をすれば、TDゲスト内からQuoteの生成を要請する方法としては、以下の3つの方法を利用する事ができる[7]：
  - ゲスト内に露出している /dev/tdx_guest に所定のコマンドでioctlにてアクセスし要請する
  - ConfigFSを経由して要請する
  - vsock経由で要請する

- ちなみに、SGXのDCAP-RAにおけるQuoteとTD Quoteとでは、大きく違うのはQuote Body（SGXにおけるReport Body相当）部分のみである

TD Quoteの構造

- まずは高レベル観点でのTD Quoteの構造を以下に示す[8]：

メンバ	説明
Quoteヘッダ	Quoteヘッダ。SGXのQuoteと共通フォーマット。 詳細は後述の表で説明
Quoteボディ （TD Quote Body Descriptor）	TD Reportから必要な情報を抽出して構築した、主に測定値関係の本体部分。SGXのQuoteとの最も大きな差分。
署名関連情報データ長	後ろに続く署名関連情報のデータ長。
署名関連情報	署名関連情報。詳細は後述の表で説明

TD Quoteの構造

- この内、まずQuoteヘッダの内訳は以下の通り[8]：

メンバ	説明
バージョン	Quote構造体のバージョン。
AKタイプ	TDQEにより使用されるAttestationキーのタイプ。これが2であれば、デフォルトのNIST P-256 ECDSAキーペアを使用している事を示している。
TEEタイプ	TEEタイプ。0x00であればSGX用の、0x81であればTDX用のQuoteである事を表す。
QESVN	TDXにおいては予約領域扱いでオール0。SGXにおいては、そのQuote構造体の生成に使用されたQE3のISVSVN。
PCESVN	TDXにおいては予約領域扱いでオール0。SGXにおいては、そのQuote構造体の生成に関与したPCEのISVSVN。
QEベンダID	TDQEのベンダID。IntelのTDQEである場合は、939A7233F79C4CA9940A0DB3957F0607というIDが割り振られている。
ユーザデータ	ユーザ定義のデータ。Intel公式のTDQEの場合、暗号化済みのPPIDとPCK Certを紐づけるためのQEIDという値が先頭16バイトに格納されている。

TD Quoteの構造

- 次に、TD Quote Body部分は以下の通り[8]：

メンバ	サイズ（バイト）	型	説明
TEE_TCB_SVN	16	バイト列	そのTDのTCBのCPUSVN。
MRSEAM	48	SHA384	TDX Moduleの測定値。
MRSIGNERSEAM	48	SHA384	TDX Moduleの署名（者）に対する測定値。ただ、[5]にはオール0固定と書かれており、現状では使用されていないのかも知れない。
SEAMATTRIBUTES	8	バイト列	恐らく、TEE_TCB_INFO由来の、SEAMに関するATTRIBUTES構造体。TDX 1.0ではオール0固定。
TDATTRIBUTES	8	バイト列	恐らく、TEE_INFO（TDINFO_BASE）由来の、TDに関するATTRIBUTES構造体。
XFAM	8	バイト列	Extended Features Available Mask。HVからTD初期化時に入力されるもので、TDX Connect含む様々な機能の有効化状況等が同梱されている[6]。
MRTD	48	SHA384	そのTDのMRTD。

TD Quoteの構造

- 前ページの続き：

メンバ	サイズ（バイト）	型	説明
MRCONFIGID	48	バイト列	TD所有者以外による設定についての、SW定義のID。例えばランタイムコンポネントやOS設定に関するもの等。
MROWNER	48	バイト列	TD所有者についてのSW定義のID。
MROWNERCONFIG	48	バイト列	TD所有者による設定についてのSW定義のID。例えば、TD所有者がデプロイしたワークロードに関する設定等。
RTMR0	48	SHA384	そのTDのRTMR[0]。
RTMR1	48	SHA384	そのTDのRTMR[1]。
RTMR2	48	SHA384	そのTDのRTMR[2]。
RTMR3	48	SHA384	そのTDのRTMR[3]。
REPORTDATA	64	バイト列	Report Data。

TD Quoteの構造

- ただしQuote v5という新しい規格のQuoteでは、このQuote Bodyは以下のようなTD Quote Body Descriptorの一部として存在する[8]：
  - Descriptorも位置としてはQuoteヘッダの次になるのは変わらない

メンバ	サイズ（バイト）	型	説明
Quote Body Type	2	uint	Quote Bodyのタイプを指定する。アーキテクチャ的には以下の3つをサポートする： - 1. (将来的にSGXに対応) - 2. TDX 1.0用 - 3. TDX 1.5用
size	4	uint	Quote Bodyフィールドのサイズ
Quote Body	可変	バイト列	Quote Body本体。Quote Body Typeに従い以下が入る： - 1. （将来的にSGXに対応） - 2. TDX 1.0用のTD Report - 3. TDX 1.5用のTD Report

TD Quoteの構造

- また、Quote v5ではQuote Bodyの末尾に以下の2つのエントリが加わる[8]：
  - ちなみに、逆に古い方のTD QuoteはQuote v4と呼ぶ

メンバ	サイズ（バイト）	型	説明
TEE_TCB_SVN_2	16	バイト列	そのTDのTCBのCPUSVN。この値は、TD-Preserving機能により新しいバージョンのTDX Moduleがロードされると、TEE_TCB_SVNと値が異なる可能性がある
MRSERVICETD	48	SHA384	MigTDのMRTD。

TD Quoteの構造

- 署名関連情報の長さに関しては自明であるため、最後にTDXにおける署名関連情報の構成を以下に示す[8]：

メンバ	説明
Quote署名	QuoteヘッダとEnclaveのREPORTに対する、Attestation秘密鍵によるECDSA署名。
Attestation公開鍵	Quote署名に使用したAttestation秘密鍵に対応するAttestation公開鍵。
TDQE REPORT	そのQuote構造体の生成に使用されたTDQEのTDQE REPORT。プロビジョニング時にシーリングされストアされたものをアンシーリングして格納する。
AK Cert	そのQuote構造体の生成に使用されたTDQEのTDQE REPORTに対する、PCK秘密鍵によるAK Cert。プロビジョニング時にシーリングされストアされたものをアンシーリングして格納する。
TDQE認証データ	TDQEが指定する追加情報。Intel公式のTDQEにおいては、ダミーのバイナリ列をプレースホルダとして使用している。
TDQE証明情報	QuoteやTDQE（Attestationキー）の信頼性を証明するための付属情報。証明情報タイプと証明情報サイズ、そして証明情報本体から構成される。デフォルトでは証明情報タイプは5となり、その場合証明情報本体はPCK Certチェーンとなる。

TD Quoteの検証

TD Quoteの検証

- TD Quoteを受け取ったら、前回スライドの通りVerifierはIntel PCSからコラテラルを取得しつつ、Intelルート～Quoteまでのトラストチェーンの検証を初めとした各種検証を行う

- 流れは以前説明の通りで、細かい部分は筆者の自著記事[9]に譲るが、コラテラルには以下のようなものがある：
  - PCK Certチェーン（これのみQuote生成時にフェッチしQuoteに同梱）
  - TCB Infoとその発行者証明書チェーン（Quote検証時にVerifierがフェッチ、以下同様）
    - 発行者証明書チェーンのトラストアンカーはPCK CertチェーンとIntelルートCA、以下同様
  - TDQE同一性情報とその発行者証明書チェーン
  - PCKCRLとその発行者証明書チェーン
  - ルートCA CRL

TD Quoteの検証

- その後、**TCB Info**をベースにそのTEEのセキュリティ状態を評価する

- TCB Infoは、マシンの**SVN**とそれに対応するマシンの**セキュリティ状態**（TCBステータス、アドバイザリID等）の**リスト**の形を取る**コラテラル**である

- これと**PCK Cert内のTCBレベル情報**を照らし合わせる事で、そのTEE・マシンの**セキュリティ状態を評価**できる
  - PCK Certの検証に成功した時点で、**そのPCK Certはそのマシンに正しく紐づいたデータとなる**点に注意

TCBレベルのチェック

- 例えば、TCBコンポネントSVNとPCESVNが**全て1**（1, 1, ..., 1）、**全て3**、**全て5**、**全て7**のTCBレベルエントリがTCB Infoコラテラルに含まれているとする

- PCK Certから抽出したTCBレベルのSVNが**全て6**である場合は、6以下かつその中で最大である、**全て5**の**TCBレベルを抽出**して**比較処理に使用**する

TCBレベルのチェック

- PCK Cert内のFMSPC値（CPUモデルを表す値）でTCB Info コラテラルをフェッチしているため、**上記条件を満たすTCBレベル**エントリは**必ず含まれている**必要がある
  - 含まれていない場合は直ちにエラーとしてリターンする
  - FMSPC：Family-Model-Stepping-Platform-CustomSKU

- 条件を満たすTCBレベルエントリを抽出したら、それに紐づいている**TCBステータス**を**取り出し**、それを検証対象マシンの検証結果（**Quote検証結果ステータス**）とする
  - このQuote検証結果ステータスは、Intelの用意している検証ロジック（QvL）のAPIを呼び出した戻り値として取得できる

Quote検証結果ステータス一覧

- TDXでは以下のようなQuote検証結果ステータスが存在する[8]：

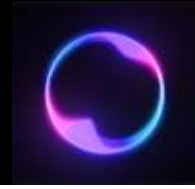
ステータス	説明	致命的か
OK	完全に信頼可能な状態。原則としてRAを受理して良い。ただし、使用しているコラテラルが期限切れを起こしている場合はそのフラグも検証結果として別個返ってくるため、期限切れが許容できない場合にはRAを棄却する事もできる。その他、後述の補足情報の中身で許容できない要素がある場合に棄却したりも可能。	-
INVALID_SIGNATURE	Quoteの署名が正しくない状態。Quoteの改竄が疑われるため、絶対に受理してはならない。	はい
REVOKED	PCK等が失効している状態。過去に侵害がありTCB Recoveryが為されていない事を意味するため、絶対に受理してはならない	はい
UNSPECIFIED	特定不能なエラー。こちらも受理しない方が無難。	はい

Quote検証結果ステータス一覧

- 前ページの続き：

ステータス	説明	致命的か
SW_HARDENING_NEEDED	TCBは全て最新化されているが、TD内で動作するソフトウェア（ユーザプログラム等）でも追加で脆弱性対策を意識した実装が必要である事を意味する。後述の補足情報内に含まれる脆弱性リストも見て、危険性や対策容易性も鑑みて許容できるか判断する。	いいえ
CONFIG_NEEDED	TCBは最新化されているが、安全性を高めるために何らかのマシン設定変更が必要である事を意味する。SGXにおける過去のEPID-RAでは例えばハイパースレッドを有効化しているとこのステータスになっていたが、現行のSGX/TDX DCAP-RAでは有効化していても出ないため、何が影響するのかは不明。どうしてもこのステータスを消せないようであれば、一定のリスクを承知の上で許容できる。	いいえ
CONFIG_AND_SW_HARDENING_NEEDED	上記SW_HARDENING_NEEDEDとCONFIG_NEEDEDが同時に発生している状態。	いいえ

Quote検証結果ステータス一覧



- 前ページの続き：

ステータス	説明	致命的か
OUT_OF_DATE	マシンや導入されたTDXが古くなっており、一定の安全性を維持できない事を意味する。TCB Recoveryで解決できる場合もあるが、そのマシン（CPU）でのアップデートがIntelにより打ち切られていたり、Intelによりアップデートが配布されていても、マザーボードベンダがそれをインストール可能な形にしていない可能性もあったりするため、これが出るとなかなか対策は難しい。基本的には拒否を強く推奨。	いいえ
OUT_OF_DATE_CONFIG_NEEDED	上記OUT_OF_DATEとCONFIG_NEEDEDが同時に発生している状態。	いいえ
TD_RELAUNCH_ADVISED	マシン側のTDX関連コンポネントは最新化されているが、対象のTDが最新化前に起動されているため、TDを再起動する事が推奨されている、というものである。実例を確認できていないが、再起動すれば直ると思われる。このRAステータスも拒否するに越した事は無い。	いいえ
TD_RELAUNCH_ADVISED_CONFIG_NEEDED	上記TD_RELAUNCH_ADVISEDとCONFIG_NEEDEDが同時に発生している状態。	いいえ

補足情報

- Quote検証結果ステータスと同時に、Quote検証結果**補足情報**（**Supplemental Data**）というものも返される

- これは、そのAttester TEE・マシンに関する様々なセキュリティ関連の情報が同梱されているため、RAの受理判断を行う上で参考にする事ができる

補足情報一覧

- 補足情報の一覧は以下の通り：

メンバ	説明
version	補足情報のバージョン。現在では下記のメジャー/マイナーバージョンで表現する仕様となっているため、後方互換性のために用意されているメンバである
major_version	補足情報のメジャーバージョン
minor_version	補足情報のマイナーバージョン
earliest_issue_date	全てのコラテラルの発行日時の内、一番古い日時がここに格納される
latest_issue_date	全てのコラテラルの発行日時の内、一番新しい日時がここに格納される
earliest_expiration_date	全てのコラテラルの期限切れ日時の内、一番早い日時がここに格納される
tcb_level_date_tag	TDX脆弱性（セキュリティアドバイザリ）に対する軽減策の内、このメンバが示す日時までにリリースされたもの全てが検証対象マシンに適用されている事を示す。
pck_crl_num	PCK CRLのCRL番号（各CRLに対し一意に割り当てられるID）

補足情報一覧

- 前ページの続き：

メンバ	説明
root_ca_crl_num	ルートCA CRLのCRL番号
tcb_eval_ref_num	検証に使用されたコラテラルのバージョンを簡易的に把握するための値[8]。上記のCRL番号を用いても同様の把握ができるため、どちらか任意の方法を選択する事ができる
root_key_id	IntelルートCA証明書の公開鍵に対するSHA384ハッシュ値
pck_ppid	検証対象プラットフォームのPPID。プラットフォーム同一性の確認に利用できる
tcb_cpusvn	検証対象マシン（に紐づくPCK Cert内）のCPUSVN
tcb_pce_isvsvn	検証対象マシン（に紐づくPCK Cert内）のPCEのISVSVN（PCESVN）
pce_id	検証対象マシンのPCEID
tee_type	検証対象マシンのTEEタイプ
sgx_type	検証対象マシンのSGXタイプ。TDXにおいては未使用

補足情報一覧

- 前ページの続き：

メンバ	説明
platform_instance_id	マルチソケットプラットフォームに対し一意に割り当てられたID。PPIDに似ているが、こちらはプロビジョニングに対するIDではなく、プラットフォーム自体に対するIDとして機能する、マルチソケット環境専用の値のようである[11]
dynamic_platform	原文[8]直訳：「SGX Registration BackendへのPackage Addコールにより、追加パッケージでプラットフォームを拡張できるかどうかを示す」。 プロビジョニング後にさらにCPUパッケージを追加できるマシンであるかのフラグであると推測される
cached_keys	原文[8]直訳：「SGX Registration Backendによってプラットフォームルート鍵がキャッシュされるかどうかを示す」。 これは補足で「Direct Registration vs Indirect Registration」とあるので、マルチソケットプラットフォームの登録方法についてのフラグであると推測される[12]
smt_enabled	ハイパースレッドが有効化されているかのフラグ。ハイパースレッドはTEEへの攻撃に悪用される事が非常に多いため、基本的に無効化されている事が望まれる

補足情報一覧

- 前ページの続き：

メンバ	説明
sa_list	検証対象マシンが抱えるTDX・SGX関連の脆弱性に対するアドバイザリIDの一覧。各IDはカンマ区切りされており、リストはヌル終端されている。例えば、Downfall（GDS）脆弱性を抱えている場合は、INTEL-SA-00828というエントリが入っている。
qe_iden_earliest_issue_date	コード実装[13]でのみ確認可能。TDQE同一性情報に含まれるデータの内、一番古い発行日時がここに含まれ
qe_iden_latest_issue_date	コード実装[13]でのみ確認可能。TDQE同一性情報に含まれるデータの内、一番新しい発行日時がここに含まれる
qe_iden_earliest_expiration_date	コード実装[13]でのみ確認可能。TDQE同一性情報に含まれるデータの内、一番早い期限切れ日時がここに含まれると思われる
qe_iden_tcb_level_date_tag	コード実装[13]でのみ確認可能。TDQEについてのtcb_level_data_tagである

補足情報一覧

- 前ページの続き：

メンバ	説明
qe_iden_tcb_eval_ref_num	コード実装[13]でのみ確認可能。TDQEについてのtcb_eval_ref_numである
qe_iden_status	コード実装[13]でのみ確認可能。TDQEについてのTCB検証結果ステータス
platform_tcb_level_date_tag	コード実装[13]でのみ確認可能。プラットフォームについての tcb_level_date_tagとあるが、具体的に何を意味するのかは不明

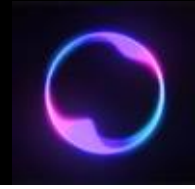
RA受理判断

- これらのQuote検証結果ステータスや補足情報の使い方は実に様々であり、致命的でない検証結果ステータスの場合は特に悩ましい

- 厳しく判定するほどセキュリティレベルは上がるが、これは受理対象のTEEマシンの数を絞る事と表裏一体である
  - これは**ユーザビリティ上の問題に直結**する

- ただし、OKであるようなマシンを見つける方が難しいSGXと違い、TDXマシンは比較的OKとなる実例もよく見かける
  - その意味では**SGX時代よりはマシ**になっていると言える

RA受理判断



- いずれにせよ、Relying Party（RP）はこれらをVerifierから受け取ったら、例えば以下のようにして判断に利用する事ができる：
  - Quote検証結果ステータスがOKでないと認めない。
  - Quote検証結果ステータスがSW_HARDENING_NEEDEDであるが、補足情報の脆弱性リスト（sa_list）を見るにあまり重大な脆弱性は無いので、リスクを承知の上で受理する。
  - Quote検証結果ステータスがOKであっても、補足情報のsmt_enabledを見るにハイパースレッドが有効化されており、ハイパースレッドは各種攻撃を助長するリスクがあるため、ゼロデイ脆弱性対策の意味で棄却する。

RA受理判断

- これらのどれが正解であるという結論はなく、これらのどれがその場その場で最適かは分からず、勿論上記の例の限りとは全く限らない
  - あくまでもRPの裁量に委ねられているため、ここはRAの中でも決定性ではない難しい部分であると言える

- いずれにせよ、RPはRAの受理判断を行い、棄却の場合にはその時点でAttesterとのやり取りを終了し、受理する場合にはReport Dataに紐付けられた安全な暗号通信路を用いて後続のやり取りを続行する

SGX不使用のTDX Attestation

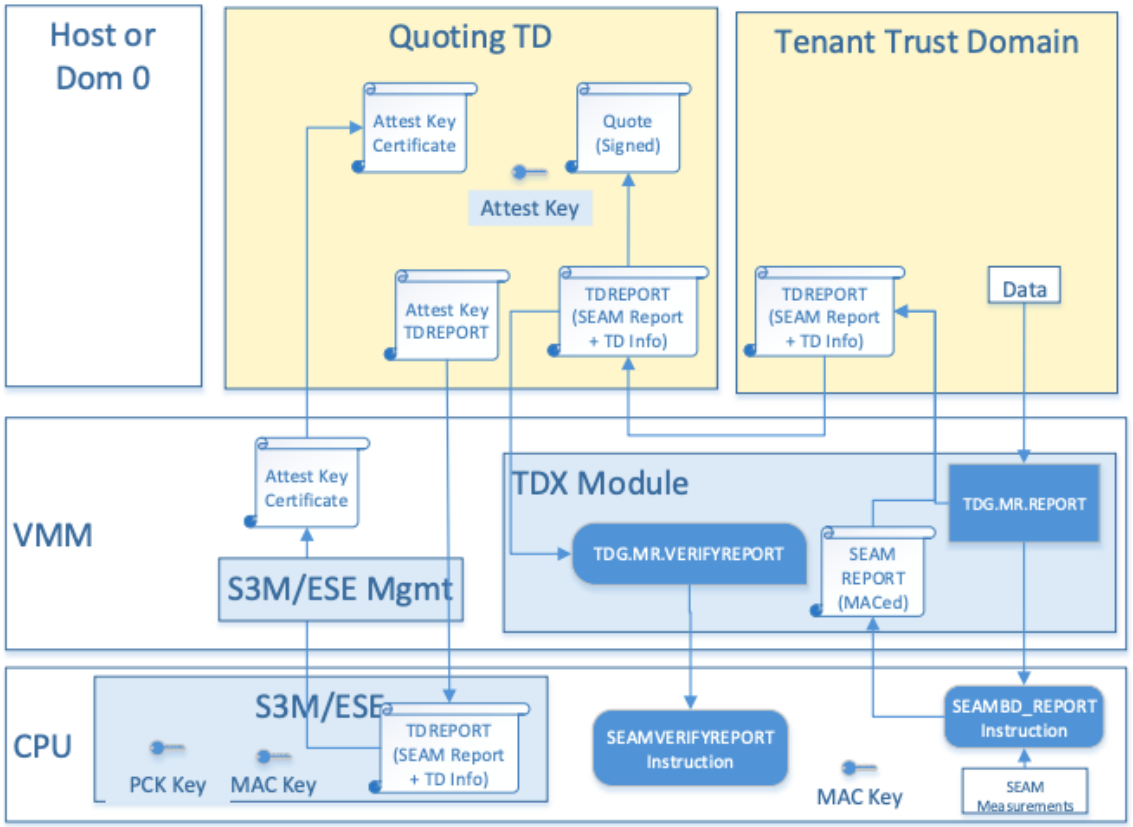
SGX不使用のTDX Attestation

- ここまで、TDXのAttestationにはSGX EnclaveであるPCEやTDQEが不可欠である事を説明してきた

- しかし、Intel公式文書[3]を見るに、**SGXに一切依存しない方式の TDX Attestationの構想が存在**するようである

SGX不使用のTDX Attestation

- 参考文献[3]の図12.5からそのまま引用する形で、以下にこの方式のAttestationの概要図を掲載する：



SGX不使用のTDX Attestation

- 一見複雑に見えるかも知れないが、これまでに説明したSGXベースのRAとの差分は大きく分けて2つだけであるように見える

- 1つ目に、TDQEが**Quoting TD**と呼ばれる専用のTDになっている
  - 参考文献[3]には、次に説明する**セキュリティエンジン**によってQuoting TD自体は認証されると記述されている

- 直接の記述はないが、恐らくは**サービスTDの一種**なのではないかと筆者は憶測している

SGX不使用のTDX Attestation

- TDになった事以外には、細かい処理を除けば基本的にTDQEとの違いはほとんど無いように見える

- ただしTDではSGX命令を発行する事はできないため、**ENCLU[EVERIFYREPORT2]**命令の代わりに、**TDCALL[TDG.MR.VERIFYREPORT]**を使うように変更されている
  - これは内部でSEAMOPS[SEAMVERIFYREPORT]をさらに呼び出す

SGX不使用のTDX Attestation

- 2つ目に、**PCE**の役割がCPU内部の**セキュリティエンジンに移管**されている

- このセキュリティエンジンは、具体的にはESEまたはS3MというものとしてCPU内に実装されるようである[14]
  - ESE：Enhanced Security Engine
  - S3M：Server Startup Secure Service Module

- このように、TDQEとPCEに依存していた部分を完全にSGX非依存にしただけであり、仕組みとしては**これまで説明したものと大差はない**
  - ただし、このQuoting TDを用いた方式の実装は筆者が探す限りは一切見当たらないため、将来的に実装される機能であるのだと予想される

SGX不使用のTDX Attestation

- SGX非依存にするメリットとして、そのマシンに差すDIMMモジュール（メインメモリ）の数を節約できる点が予想できる

- Scalable-SGXは**全DRAMチャネルにDIMMモジュールを差さないと有効化できない**という制約が存在する
  - 筆者が確認した限り、**TDX有効化の方にはこの制約は発生しない**
  - つまりはSGXを使うが故に大量のメモリを搭載する必要があった

- TDX完結になればこのような制約も無くなるので、LLM競争でメモリが暴騰している中、お財布に優しくなると想定される

おまけ：TDKEYREQUEST機能

TDKEYREQUEST

- Intel公式文書[6]によれば、TD内でそのTDやTCBの測定値を材料にした鍵の導出を行えるとされている

- これは、**TDCALL[TDG.MR.KEY.GET]**により取得する事が可能で、**TDKEYREQUEST構造体**でどのような鍵を導出したいかを指定できる

TDKEYREQUEST

- これは、SGXで言えばENCLU[EGETKEY]による**シーリング鍵の導出に相当する機能**と言える
  - Intel公式文書[6]を見るにMigTD周りを念頭に実装されたようにも見えるが、その他の用途に用いる事も可能なはずである

- この鍵は、Intel公式文書[6]の§5.5.8.3.2を見る限り、SGXのシーリング鍵と同じく**そのHWに固有な鍵**である

TDKEYREQUEST

- TDKEYREQUESTの中にはTDKEYPOLICYが含まれ、どの測定値を導出材料とするかを指定する事ができる
  - 詳細は不明だが恐らく複数選択可

- 現在Intel公式文書[6]で列挙されている測定値としては、MRTD、MROWNER、MRCONFIGID、MROWNERCONFIG、MRSIGROOT、MRSIGNER、MRPRODID、RTMR[0]～[3]があるようである

- ただしこの機能はマシンによって対応しているかが分かれると2026/6現在のIntel公式文書[6]に書かれているため、安定して使用可能になるのはもう少し先の事になると思われる
  - ちなみに、前までは未実装の機能であった

TDKEYREQUEST

- 2026/8におけるTDX Module 1.5.34での実験の限りでは、TDCALL[TDG.MR.KEY.GET]が内部で呼び出すSEAMOPSが未搭載であるために利用不可能である事は確認済み
  - 第5世代Xeon-SPマシンで確認

- 今後マイクロコードアップデートで追加されるのか、あるいはより新しい世代のCPUでしか対応しないのかは不明

実際にTDX RAを実装する？

実際にTDX RAを実装する

- それでは、実際にこれまでに説明した**全項目**（ReportやQuote等の各メンバ等全て含む）**を意識**しながら**TDXのRAを実装**しましょう！

初学者に実装させるには
RAはあまりにも非人道的

Humane-RAFW-TDX

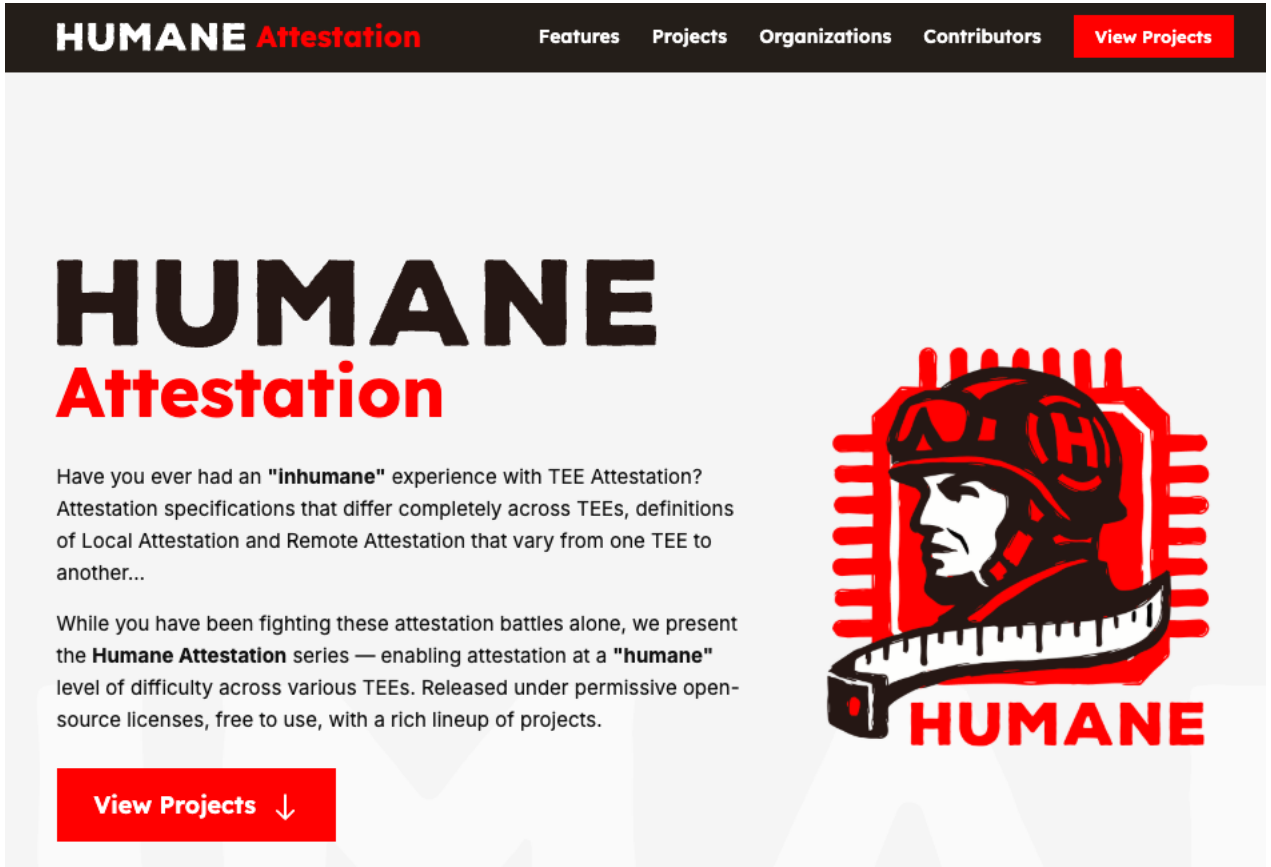
- 昔のSGXのRAであるEPID-RAよりは遥かに公式サンプルコードがマシとは言え、ここまで説明したものを全て意識してのRAの実装は**非常に高度**と言える

- よって、E2Eでしっかりと TDXをRAするためのフレームワークである**Humane-RAFW-TDX**を講師が公開している
  - https://github.com/acompany-develop/Humane-RAFW-TDX

- キャンプ本番ではRAを行う事は大前提であるが、このフレームワークを用いる事で**大幅に手間を省く事が可能**である

Humane-RAFW-TDX

- ちなみに、TDX以外にも~~講師がSGX時代キレながら実装し始めた事が~~~~きっかけの~~人道的にAttestationをできる**Humane Attestation**シリーズの公式サイトがあるため、そちらも是非参照の事



まとめ

- TDXにおけるDCAP-RAについて、その比較的詳細な部分について説明した
- TDX完結のAttestationやTDKEYREQUESTといった、現段階では未実装の機能についても説明した
- TDXを簡単にRAするためのフレームワークであるHumane-RAFW-TDXを紹介した

参考文献

[1] Pau-Chen Cheng, Wojciech Ozga, Enriquillo Valdez, Salman Ahmed, Zhongshu Gu, Hani Jamjoom, Hubertus Franke, and James Bottomley. 2024. Intel TDX Demystified: A Top-Down Approach. ACM Comput. Surv. 56, 9, Article 238 (September 2024), 33 pages. https://doi.org/10.1145/3652597

[2] “Intel® Trust Domain Extensions - SEAM Loader (SEAMLDR) Interface Specification”, Intel, https://cdrdv2-public.intel.com/739045/intel-tdx-seamldr-interface-specification.pdf

[3] “Intel® Trust Domain Extensions (Intel® TDX) Module Base Architecture Specification”, Intel, https://cdrdv2.intel.com/v1/dl/getContent/733575

[4] “Intel® TDX Virtual Firmware Design Guide”, Intel, https://cdrdv2.intel.com/v1/dl/getContent/733585

[5] “Intel® Trust Domain CPU Architectural Extensions”, Intel, https://cdrdv2-public.intel.com/733582/intel-tdx-cpu-architectural-specification.pdf

[6] “Intel® Trust Domain Extensions (Intel® TDX) Module Architecture Application Binary Interface (ABI) Reference Specification”, Intel, https://cdrdv2.intel.com/v1/dl/getContent/733579

[7] “tdx_attest.c”, GitHub, https://github.com/intel/confidential-computing.tee.dcap/blob/DCAP_1.25/QuoteGeneration/quote_wrapper/tdx_attest/tdx_attest.c#L785

参考文献

[8] “Intel® Trust Domain Extensions Data Center Attestation Primitives (Intel® TDX DCAP): Quote Generation Library and Quote Verification Library”, Intel, https://download.01.org/intel-sgx/latest/dcap-latest/linux/docs/Intel_TDX_DCAP_Quoting_Library_API.pdf
[9] “【技術】Intel SGX - DCAP-RA解体新書”, 自著記事, https://www.acompany.tech/privacytechlab/intel-sgx-dcap-ra
[10] “Intel® Software Guard Extensions (Intel® SGX) Data Center Attestation Primitives: ECDSA Quote Library API”, Intel, https://download.01.org/intel-sgx/sgx-dcap/1.25/linux/docs/Intel_SGX_ECDSA_QuoteLibReference_DCAP_API.pdf
[11] “Intel SGX(R) PCK Certificate and CRL Profile Specification, Intel”, https://api.trustedservices.intel.com/documents/Intel_SGX_PCK_Certificate_CRL_Spec-1.5.pdf
[12] “Remote Attestation for Multi-Package Platforms using Intel®SGX Datacenter Attestation Primitives (DCAP)”, Intel, https://download.01.org/intel-sgx/sgx-dcap/1.20/linux/docs/Intel_SGX_DCAP_Multipackage_SW.pdf
[13] “sgx_qve_header.h”, Intel, https://github.com/intel/confidential-computing.tee.dcap/blob/DCAP_1.25/ae/QvE/Include/sgx_qve_header.h
[14] “Confidential compute architecture for silicon initialization for ip protection and assurance”, 特許文書, https://patents.google.com/patent/WO2023230834A1/en