# 8. Remote Attestation①

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、全体を通しての大まかな流れについて説明する

- TDXのRAはSGXのRAとほぼ同一であるため、徹底的な解説については以下のリンク先を参照（必要十分な詳細の掘り下げは次スライドにて実施）：
  - https://www.acompany.tech/privacytechlab/intel-sgx-dcap-ra
  - https://qliphoth.io/media/2025-6-remote-attestation.pdf
  - 以降、本スライドで引用明記のない部分の論拠となる文献・コードは1つ目のリンク先の記事を参照

# TDX Attestationの概要

# TDX Attestationの概要

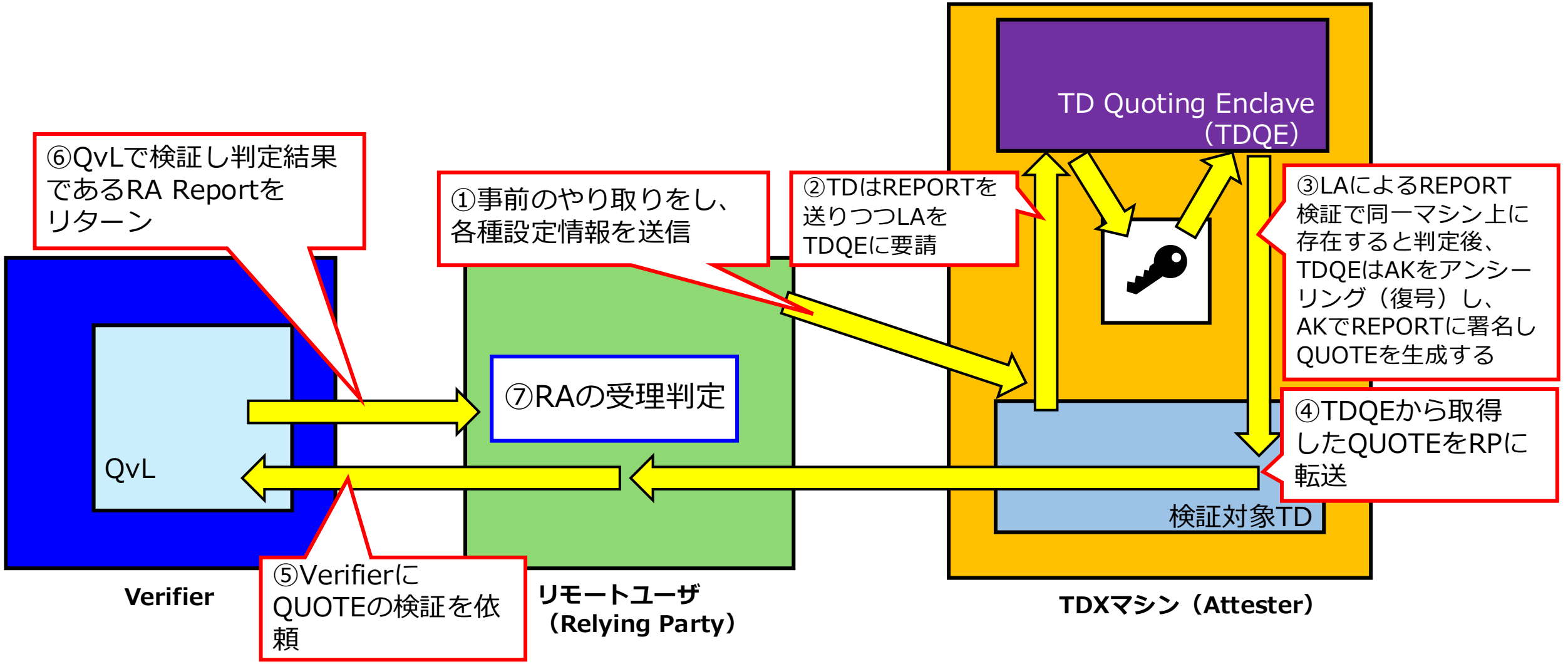
- 前回スライドで説明した基本的なRAの流れはあくまでも汎用性を重視した説明なので、細かい所ではTDX RAはそれと差分がある

- まず、ARを生成する上で前回スライド後半で説明した通り一度**LA**（Local Attestation）を実施する**多段階処理**となっている
  - LAのための身元証明書を**TD Report**（あるいは単にReport）、RAのための身元証明書を**TD Quote**（あるいは単にQuote）と呼ぶ
  - この**TD Quoteがいわゆる AR相当の存在**である

- AK相当の鍵が二段階存在する。具体的には**AK**本体と、それを認証するプラットフォーム由来の**PCK**という鍵の2つである
  - ベンダ（Intel）による認証はPCKによって行われる

# TDX Attestationの概要

- 以下にTDXにおけるRAの概要図を示す：



# TDX Attestationの概要

- 前図の内、手順①～④がQuote（AR）の作成、⑤、⑥がQuoteの検証、そして⑦が検証結果を受けてのRA受理判断である
  - AKに対する証明書チェーンの形成はRA実行よりも前に実施されるため、前図からは省略している

- このように、一部手間が増えている部分はあれど、TDXのRAも基本的なRAの流れに沿っている事が分かる

# ステップ1. プラットフォーム登録

# プラットフォーム登録

- TDXは、SGX時代から提供されている**DCAP**（DataCenter Attestation Primitives）というツール・ライブラリ群が提供する、**DCAP-RA**と呼ばれる方式に則って行われる
  - つまりSGXとTDXで基本的に共通のRA方式を使うという事である
- SGXにおいては、 DCAP-RAの前身のRA方式として、現在では既に廃止されているEPID-RAというものが初期では提供されていた
- この時代は、プロセッサ製造時にプロセッサ内のeFuseに**RPK**というHW-RoT鍵を焼き付け、そしてそれはIntel側の厳重な施設である**iKGF**でも保持管理されていた
  - RPK: Root Provisioning Key
  - iKGF: Intel Key Generation Facility

# プラットフォーム登録

- RPKは**PK**という鍵の導出に使用され、このPKはEPID-RAにおけるAKをそのSGXマシンに**プロビジョニング**（デプロイ）するための認証に使用された
  - PK: Provisioning Key

- この認証は、IntelがRPKから相手の状態に合わせて導出したPKに、相手から来たPKの情報が合致している場合にのみAKを返すという仕組みであった

- つまり、CPUとIntelのみがRPKを知っている状態であったため、言わば「**製造時にベンダがHW-RoTを控えておく**」方式であると言える

# プラットフォーム登録

- 一方、TDXでも使用されている現役の**DCAP-RA**の場合、この**PK**は後述の**PCKの導出**に主に**使用される**
  - PCKは、前述した「二段階のAK」の内上位の方で、下位のAKの認証を行う
- そして、Scalable-SGXやTDXにおいては、少なくともマルチソケットプラットフォームにおいては、このRPKをPK関連の導出には**直接は使用しない**
  - Scalable-SGX：第3世代Xeon-SP以降に搭載されている新しいSGX
- 代わりに、CPUパッケージ間で交渉し確立する**Platform Key**という鍵をPKの導出に用いる[1]
  - Platform Keyは導出後はCPU固有の鍵によって暗号化されストアされると書かれている[1]

# プラットフォーム登録

- ではScalable-SGXやTDXであっても、CPUを1つしか差していない（シングルパッケージ）場合は従来通りRPKを使うかというと、確かにIntel公式文書[1]上はそう読み取れる

- しかし、Intelが公開している現行のセットアップ手順[2]を見ると、シングル・マルチ問わずに**Platform Manifest**というものをIntelの登録サービス（IRS）に送信している
  - IRS: Intel Registration Service

- Platform Manifestは、**RSEK**という鍵で暗号化されたPlatform Keyを内包し、**RSAK**という鍵で署名されている
  - **RSEK**：Registration Service Encryption Key
  - **RSAK**：Registration Service Authentication Key

# プラットフォーム登録

- この事実からは、以下の2通りの可能性が考えられる
  - シングルの場合でもPlatform Keyを単一CPUで生成しPlatform Manifestに入れている
  - RPK（またはそこから導出される鍵）をPlatform Manifestに入れている

- いずれにせよ、もはやIntelはCPU製造時には、**HW-RoT鍵を管理**（iKGFへのIntel側自身による登録）は**していない**という事になる

# プラットフォーム登録

- Platform Keyの場合、初使用時等にマシンで動的に生成されるものであるためIntelは知りようがない

- 仮説としてもしシングル時にRPKを送信している場合にも、送信しているという事自体が「IntelがRPKを知らない」事を意味しているため、同じくIntelの製造時管理から外れている事になる

- よって、TDX含む現行のDCAP-RAは、言わば「**実利用開始時にHW-RoT鍵を登録するタイプ**」であると言える

# プラットフォーム登録

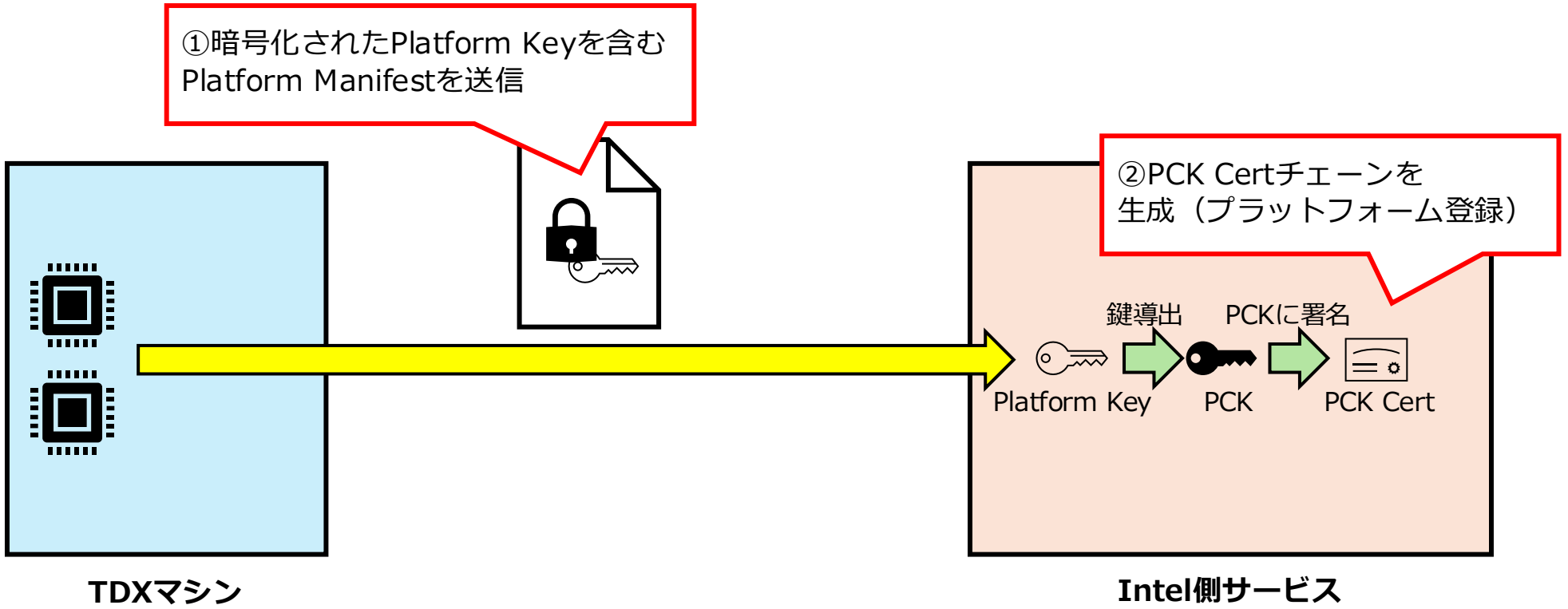
- いずれにせよ、HW-RoT鍵が登録されたら、Intel側でもPCKを導出できるようになるため、PCKに対する証明書（**PCK Cert**）を発行する

- PCKは、前回スライドのSEV-SNPにおけるVCEKと同様、ベンダ（Intel）による**中間CA秘密鍵**で署名され、さらに中間CA公開鍵は**IntelルートCA秘密鍵**で署名される
  - これを以てしてIntelへのPCKの登録、つまりはプラットフォーム登録が完了する

- この3段階からなる証明書を**PCK Certチェーン**と呼ぶ

# プラットフォーム登録

- このプラットフォーム登録の概要図を以下に示す：
  - 紙面の都合上、Platform Key→PKの導出に関しては図中では省き、直接PCKを導出しているように見えるようにしている



# ステップ2. プロビジョニング

# 二段階のAK

- プロビジョニングと言えば、Quote（AR）に署名する**AK**を**Attesterの狭義のTCBにデプロイする処理**である

- ステップ1によりPCKはデプロイされたが、実はこの時点では**まだプロビジョニングはされたとは言えない**

- 何故ならば、前述の通りDCAP-RAには**AK相当が二段階存在**するためである

# 二段階のAK

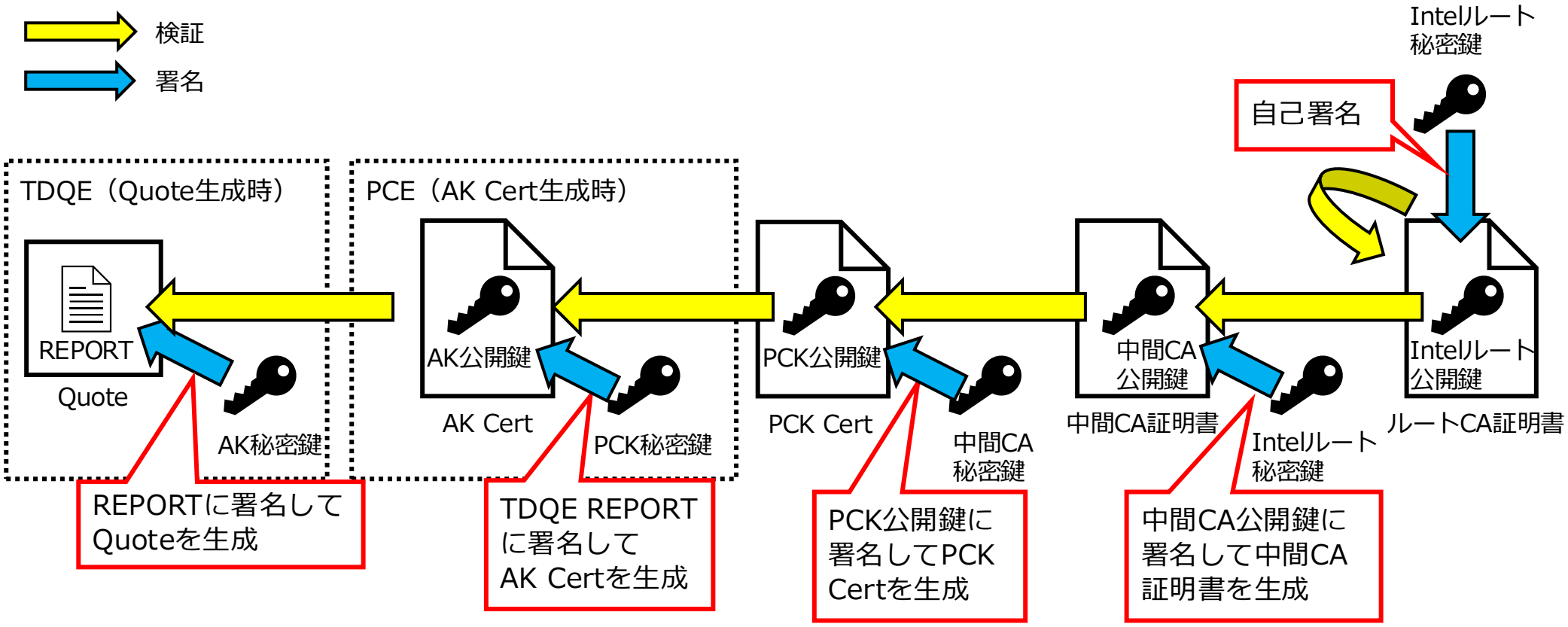
- IntelのDCAP-RAは、ややこしい仕様として**PCK秘密鍵がまた別のAKをさらに認証する**

- このAKは実際にAR（Quote）の署名を行う鍵であり、名前もそのまま**AK（Attestation Key）**である

- よって、このAKを生成してPCKで認証して初めてDCAP-RAにおけるプロビジョニングが完了したと言える

# DCAP-RAにおけるトラストチェーン

- TDX（DCAP-RA）におけるトラストチェーンの概要図を以下に示す：



# 二段階のAK

- 二段階のAKという複雑な事をしている理由の1つとしては、TDXも用いている**DCAP-RA**は、**SGX初期時代から存在**するものであり、**TDXはこれをほぼそのまま流用している**からというものがある

- 前図における**TDQE**や**PCE**は、いずれもSGX Enclave（SGXのTEEインスタンス）である
  - **PCE**：Provisioning Certification Enclave
  - これらはより専門的にはArchitectural Enclave（AE）と呼ばれ、SGXの狭義のTCBとして機能する

# 二段階のAK

- PCEは、DCAP-RAにおける二段階のAKの内上位側である**PCK**を**PKから導出**し、（下位側の）**AK公開鍵に署名する事で認証**する役割を持つ
  - **PCK**: Provisioning Certification Key
  - より正確にはAK公開鍵をReport Dataに含む、次に説明するTDQEのReportに対して署名する
- 一方でTDQEは実際に**AKでTD Reportに署名**し**TD Quoteを生成**する役割を持つ
- いずれもマイクロコードで実装するには複雑すぎるという事で、代わりにSGX Enclaveという形で実装された[3]

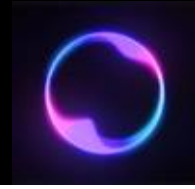
# AKが二段階構成である理由

- ところで、直感的には例えばPCEのみを用意してPCKでQuoteに署名する、あるいはPCKを挟まずTDQEにてAKで署名し、そのAKに対してIntelが証明書チェーンを発行すれば良いように思える
  - そして現代の状況からするとこの直感の方が正しい

- これにはEPID-RAの事情が絡んでおり、当時最初期はIntelが用意したQEでQuoteを署名し、その検証もIntelのサービスである**IAS**が実施するEPID-RAしか存在しなかった
  - **IAS**: Intel Attestation Service

- その結果、Intelへの信頼の重きが大きすぎるので、自前で検証する択も用意しろ、という意見が発生した[4]

# AKが二段階構成である理由



- TEEである以上**Intelは信頼する前提**であるため、今となっては変な意見にも見えるが、当時は今よりも余程コミュニティも未成熟で、様々なブレがあった事は念頭に置くべきである

- そこで登場したのがDCAP-RAで、IntelはHW-RoT鍵から導出され、かつIntelによる署名のついているPCEのみが導出可能である、**PCKに対して認証を行う以上の役割**を**Intelが持たないよう**にした

- かつ、EPID-RAとは異なり、Quoteの生成を行う**QEもユーザが作り**、**独自の署名方式**で**独自に署名**できるようにした
  - 勿論Intel提供のQEを用いる事も可能である

# AKが二段階構成である理由

- この構成は以下の両方のニーズを満たせる事を意味する：
  - プラットフォームに対して最低限Intelのお墨付きを与える仕事はIntelがしなければならない
  - Quoteに署名するQEはユーザが自由に作り、かつ好きな署名方式で署名できるようにしたい

- このために、**プラットフォームの認証**だけはIntelがIntel製AEであるPCEのみが導出できる**PCKに署名する事で実現**し、そこからさらにQEでAKを導出させ**PCKで認証**する二段階構成とした
  - これにより、QEの自由度を持たせながらPCKでプラットフォームを認証できるようにした
  - DCAP-RAにおけるQEの内、SGX用のものを**QE3**、TDX用を**TDQE**と呼ぶ

# そもそもSGXのRAを使い回している理由

- それでもまだ疑問として残るのは、何故TDXがこのような（背景としては古い）SGXのRA基盤を流用しているのかという事である
  - TDX Moduleにさせるのは前回スライドの議論からともかく、それこそ専用のTDでQuoteを作るように何故しないのか？

- これは恐らく単純で、TDXを開発提供する上でスピードを重視し、流用できるものは流用して**時短**したかったためであると考えられる
  - スクラッチで実装すると大変だが、QEがTD Reportを認識できるようにすれば後はそのままDCAP-RAを流用可能で工数が大幅に下がる

- ちなみに、次回スライドにて軽く触れるが、「TDでQuoteを作る」方式も**仕様書上は存在**している

# プロビジョニング

- プロビジョニングに話を戻すと、要は**TDQE内でAKを生成**し、それを**PCEがPCKで認証**する、つまりはいわば**AK Certを生成**すれば
  それで**プロビジョニングは完了**である

- これには前回スライドで説明した**LA**が使用される

# プロビジョニング

- AK Certを生成するには、まずTDQEのReportのReport Data部分にAK公開鍵のハッシュを入れるようにする
  - そうして生成されたReportは、前回スライドの通りReport KeyでMACが付与されている

- 次に、このTDQE ReportをPCEに送信し、PCEはこれを**LA**後、**PCKで署名**する事で**AK Certを生成**する

- 生成されたTDQE Report、AK Cert、AKは、RA本番で使う時のためにSGXの**シーリング**（Sealing）と呼ばれる機能で**暗号化して安全に保管**する
  - シーリングしたデータの復号の事を**アンシーリング**と呼ぶ

# ステップ3. Quoteの生成

# Quoteの生成

- これにより、Quote署名に使うAKと、それをPCKにより認証した証明書であるAK Certが用意できた

- 後はこのAKで署名を行えば、IntelルートCAをトラストアンカーとした盤石なトラストチェーンにバインドされた身元証明書、つまりはQuoteを生成する事ができる

- しかし、TDQEはあくまでもSGXコンポネントであるため、TDXの要素である**TD Reportを取得**するのには**一定のハードルが存在**する

# Quoteの生成

- SGX EnclaveはVMXモードにすら関与しない、通常の**非特権ユーザ空間で動く**ため、**TDXのSEAMモードには全く関与せず**、さらに言えば**SEAM VMX rootに干渉できるわけがない**
  - TDXの測定値等を格納するTDCSを管理するのはSEAM VMX rootで動作するTDX Moduleである

- もし仮にTDQEがTDCSから測定値を直接持ってこられたら、それはTDXからしたら野良のユーザプロセスであるTDQEがSEAM VMX rootに土足で踏み入っている事になる
  - これでは**TDXのセキュリティ原則が崩壊**してしまう

# Quoteの生成

- よって、代わりに**TD自身**が**TDCALL**で**TDX ModuleにTD Reportの生成を要求**する事で、TDXのセキュリティ境界を守りつつReportの生成を実現する
  - この時、必要に応じてReport Dataに同梱させたいデータも渡し同梱させる

- ちなみにこのTD Reportの生成は、TDCALLにより呼び出される**TDX Module内の実装**と、TDX Moduleからさらに呼び出される**SEAMOPS命令**の**SEAMREPORT**または**SEAMDB_REPORT**リーフ関数が実施する
  - つまりは、TD Reportの生成はTDX Module及びCPUマイクロコードが行っているという事である

# Quoteの生成

- このTD ReportのままではReport Keyによる同一マシン上由来かの確認（LA）しかできない
  - 直接TD Quoteを作ればいいのにと思うかも知れないが、これまた前述の通りSGXのQEを流用している以上、QE自体がLAを前提としているため仕方がない
- よって、TDQEがこのTD Reportを用いて**Attester TDをLA**し、**AKで署名する**事で**TD Quoteを生成**するのは前回スライドで説明した通り
- 実務視点で見れば、生成したTD ReportをTDゲストOS内からアクセス可能な/dev/tdx_guestを通す等の方法でホスト側に構えるTDQEに対しTD Reportへの署名（TD Quoteの生成）を要求できる
  - TD Reportの生成もこのデバイスを通す事で要求できる

# ステップ4. Quoteの検証

# Quote署名の検証

- TD Quoteを受け取ったら、AttesterはVerifierにそれを転送する事でQuoteの検証を要請する

- TD Quoteを受け取ったら、まずはVerifierは**Quoteの署名が正しいか**を、前述したRAの作法に従って**検証**する
  - それには、Quoteの署名に用いられているAKに対する証明書である**AK Cert**と、AK Certを生成したPCKに対する**PCK Certチェーン**も**必要**である

- AK CertはTDQEから受け取る事ができるが、**PCK Certチェーン**に関してはプラットフォーム登録を受けて完全にIntelが発行しているものですあるため、**Intelから受け取る必要がある**

# Quote署名の検証

- Intelは、Intel PCS（Provisioning Certification Service）というサービスを公開しており、Quoteの検証に用いる事ができる以下を含む様々な情報をここから取得する事ができる：
  - PCK Certチェーン
  - そのTEEについてのセキュリティ情報を内包するTCB Info
  - Intelにより失効済みのPCK Certの一覧であるPCK CRL（PCK Cert Revocation List）

- Intelは、これらのRAにおいて使用する、PCSからフェッチ可能な情報を**コラテラル**（付属情報）と呼んでいる

# Quote署名の検証

- ただし、毎回RAの度にリモートのIntel PCSからフェッチすると無駄が多くなり、かつIntelのサービスにも負荷がかかるためか、そのような毎回のフェッチは**利用規約で明確に禁止**されている

- 代わりに、コラテラルをキャッシュしておくキャッシュサービスである**PCCS**というものを自身の環境（例：ローカル上、自社ネットワーク内等）に用意し、そこにキャッシュさせてRAに利用すると良い
  - **PCCS**：Provisioning Certification Caching Service

- デフォルトのDCAP-RA実装では、PCK CertチェーンはTD Quoteの発行時にPCS/PCCSからフェッチしQuoteに同梱され、それ以外のコラテラルはVerifierがPCS/PCCSからフェッチする
  - これは、PCK Certチェーン以外はPCK Cert内の情報でクエリできる一方、PCK Certチェーンのみはマシン固有の情報を必要とする事によると思われる

# Quote署名の検証

- TD Quoteとコラテラルが揃ったら、Verifierの検証コードに**ハードコーディング**されている**IntelルートCA公開鍵**を用いて**コラテラルについている署名を検証**する
  - この検証コードとして、デフォルトでは**QvL**（Quote Verification Library）と呼ばれるものが使用される

- この時、ルートCA公開鍵による**PCK Certチェーンの検証**、PCK Cert（内のPCK公開鍵）による**AK Certの検証**、そしてQuoteに添付されているAK公開鍵による**TD Quoteの署名検証**を行う
  - Report DataにはAK公開鍵のハッシュ値しか入っていないため、AK公開鍵本体はQuoteに添付される

# Quote署名の検証

- この署名検証により、そのTD Quoteが真に**Intel**が把握し正当であると**お墨付きをつけているTDXマシンから来たものである**事を**確信**できる

- 同時に、PCK CRL等との照合も行う事で、そのマシンのPCKが過去に脆弱性を突かれて漏洩した等の理由で**Intelにより失効されていない事も確認**する

# Quoteの信頼可能性の判定

- TD Quoteの署名が正しい事が分かったら、今度はTD Quote内に含まれるTCBSVN（CPUSVN）という、AttesterのTDXマシンのCPUやTCBに対するSVN（**TCBSVN**）を取り出す

- その後、そのマシンで取り得る全てのTCBSVNと対応するセキュリティ情報を列挙している**TCB Infoコラテラル**を、この**取り出したTCBSVNで検索**し**当てはまるセキュリティ情報を取得**する

- そのセキュリティ情報を見てその**Attesterが信頼可能であるかの判定を下し**、その**判定結果**及び、もし何らかの脆弱性を抱えている場合にはその**脆弱性ID**（例：INTEL-SA-00657）を**抽出**できる

# ステップ5. RAの受理判断

# RAの受理判断

- 判定が完了したら、Verifierは判定結果、脆弱性ID等の補足情報、そしてTD QuoteをRelying Party（RP）に送信する
  - どのような判定結果が存在するかに関しては次回の記事で詳述する

- RPは、上記の結果や情報を受け取ったら、それを受けて**そのRAを受理するか判断**する
  - 例えばPCKが失効している、TD Quoteの署名が壊れているといった判定結果は直ちに棄却すべきである
  - 一方、例えばいくつかの脆弱性を抱えているだけといった場合には、その脆弱性の脅威を鑑みた上で受容する判断を下す事も可能である

# RAの受理判断

- その後、TD Quoteをパースし、必要に応じて**各種測定値**（次回記事で詳述）を**リファレンス測定値と比較**する
  - 少なくとも、TDXの各種動作定義に関する測定値（次回記事で説明するMRTD等）はチェックしておいた方が良い

- 最終的に、Verifierによる**判定結果が許容可能**なもので、かつ**測定値のチェックの結果も問題ない**場合には、RPは**RAを受理すると判断**する
  - その後はReport Dataに紐づいた安全な暗号通信路を通してAttesterとの本来やりたかったやり取りを続行できる

- 一方、Verifierの判定結果や測定値チェックの結果**許容できない**場合には、**RAを棄却**し**その場でセッションを終了**する

# RAの受理判断

- なお、判定結果に対するRP側判断やリファレンス測定値との比較は、それを指定するポリシをVerifierに渡し、それ込みで全てVerifierに実施してもらう事も可能である

- この辺りはRAのインフラ設計や、使用するVerifierの仕様に沿って決める事になる

# SGXのDCAP-RAとの相違点

# SGXのDCAP-RAとの相違点

- SGXとの相違点としては主に以下が挙げられる：
  - LAのReportの生成と検証に用いる命令
  - QEの種類（SGXはQE3、TDXはTDQE）
  - ReportやQuoteの中身（含まれている各種測定値等）

- よって、これらを脳内で置き換えてしまえば、冒頭で紹介したDCAP-RAの徹底解析記事を読む事でTDXのDCAP-RAもおおよそ完全にマスターできる
  - が、そもそもが極めて難解なので次回のスライドの後に読む事を推奨

# まとめ

- TDXにおけるDCAP-RAのおおまかな流れについて説明した

# 参考文献


[1]”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

[2]”Infrastructure Setup - Intel® TDX Enabling Guide”, 2026/3/19閲覧, https://cc-enabling.trustedservices.intel.com/intel-tdx-enabling-guide/02/infrastructure_setup/

[3]”Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution”, Jo Van Bulck et al., https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-van_bulck.pdf

[4]”Supporting Third Party Attestation for Intel® SGX with Intel® Data Center Attestation Primitives”, Vinnie Scarlata et al., https://cdrdv2-public.intel.com/671314/intel-sgx-support-for-third-party-attestation.pdf

[5]”Intel® Trust Domain Extensions White Paper”, Intel, https://www.intel.com/content/dam/develop/external/us/en/documents/tdx-whitepaper-final9-17.pdf

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