

2. Intel SGXの基礎

Ao Sakurai

2023年度セキュリティキャンプ全国大会
L5 - TEEの活用と攻撃実践ゼミ

本セクションの目標



- TEEの最も有力な実装例であるIntel SGXの基礎知識を一通り学習する。
- 実世界におけるSGXの利用例をいくつか眺めていく。

Intel SGX

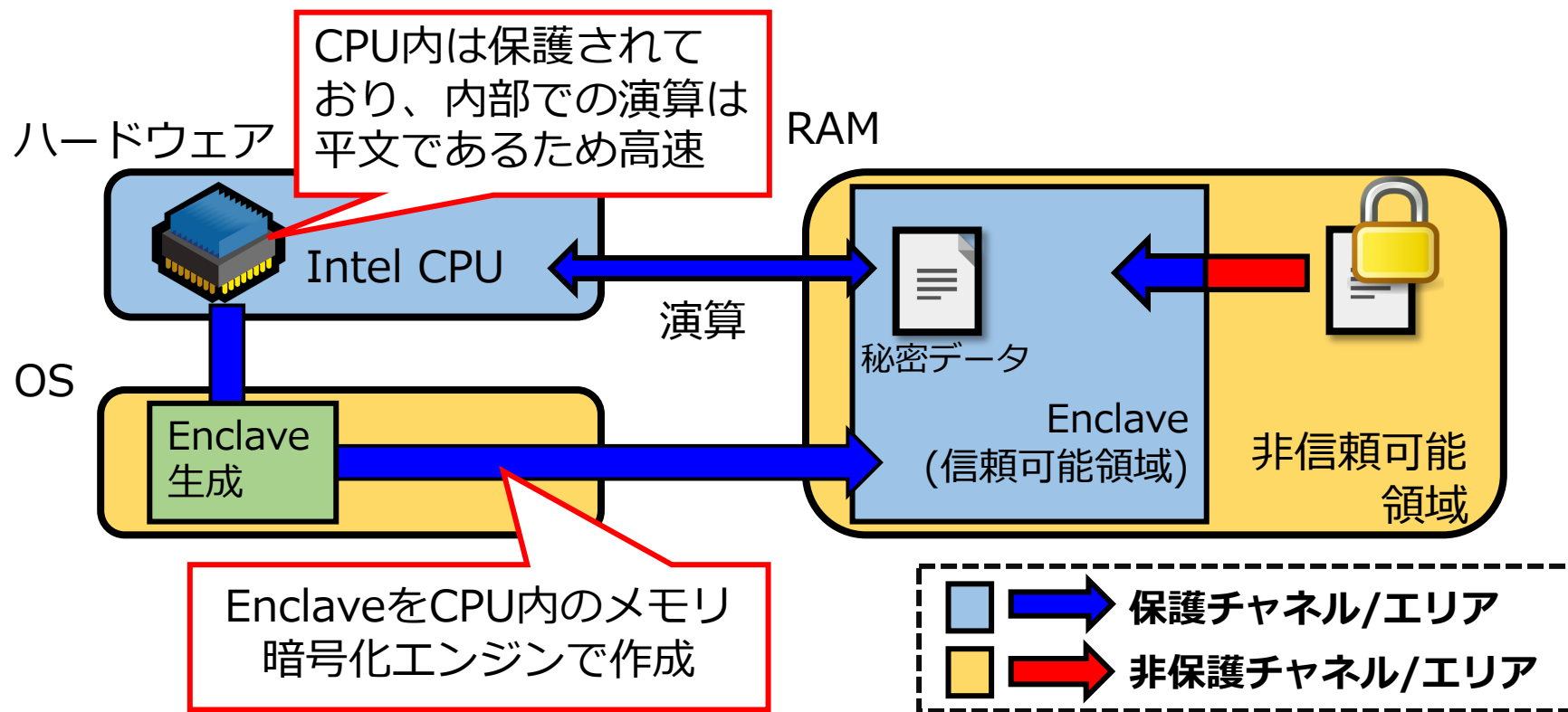


- SGXについて徹底的に仕様を学習したいのであれば、以下の論文が非常に有名である：
 - <https://eprint.iacr.org/2016/086.pdf>
- ただし、これを読み解くには多大な労力が必要であり、かつこの内容全てを解説しているとキャンプ期間がそれだけで終わってしまうため、必要な分だけ説明を行う
 - 何なら講師自身も全部は読んでいない

Intel SGX (Software Guard eXtensions)



- **SGX**: CPU内の専用ユニット (MEE)によってメモリ上に**保護領域 (Enclave)**を生成し、**OS**や**VMM**からすらも**データを保護**しながら**プログラムを実行**
 - 第6~10世代Coreシリーズ、第2 (一部) , 3世代Xeonシリーズで対応



Enclave (エンクレーブ)

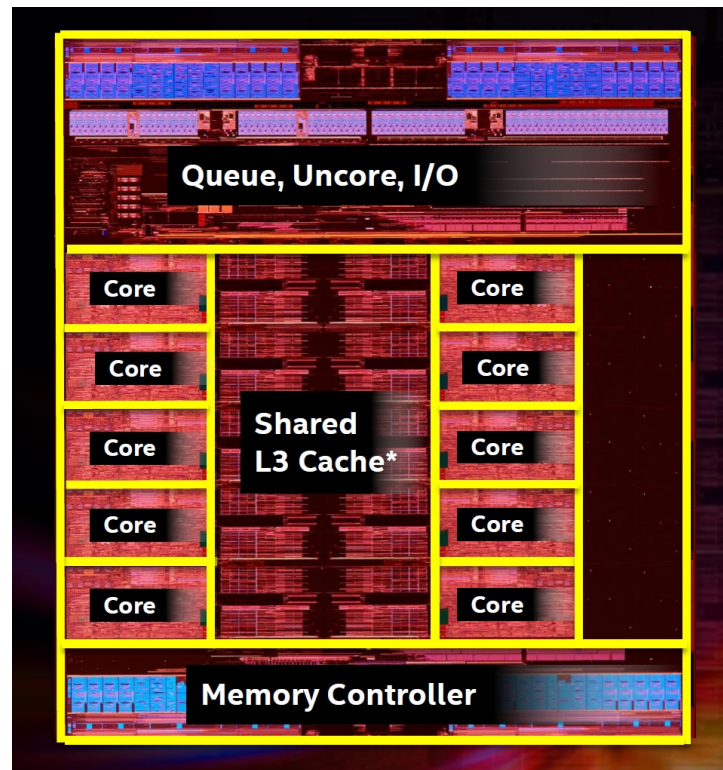


- 直訳で「**飛び地**」の意味であり、ここ（厳密には後述の**EPC**）に秘密情報を格納する事で**データ安全性・完全性**を保証する
- **アンコア (Uncore)** : CPUパッケージの内、**コア (Core)**の外に存在する周辺回路
- **メモリ暗号化エンジン (MEE)** : Enclaveの暗号化や完全性の保証を行うUncore上のユニット。Memory Encryption Engine. 主に以下の技術の複雑な組み合わせで構成されている[3]
 - 微調整された128bit AES/CTR暗号
 - マークルツリー
 - Carter-Wegman式メッセージ認証符号

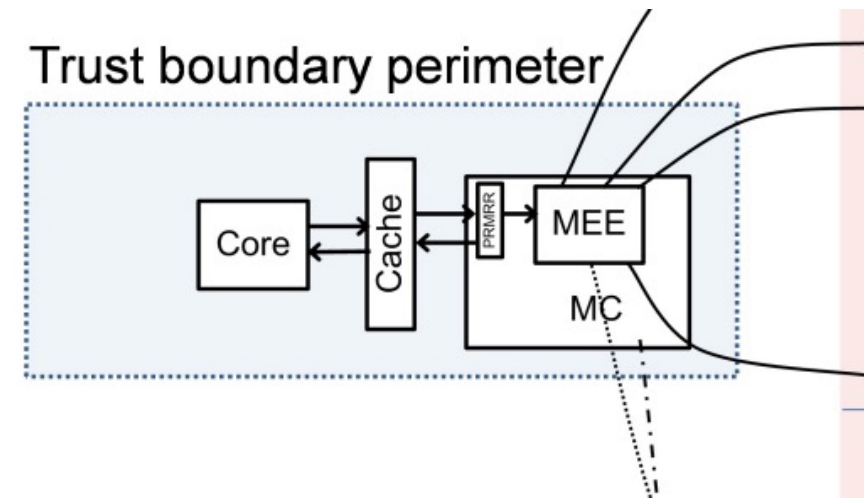
画像で見るメモリ暗号化エンジン



- 左図はCore i7 6950XのCPUダイの画像。左図下部のメモリコントローラ（MC）の中に、右図のようにMEEが含まれている



引用：参考文献[2]



引用：参考文献[3]

メモリ暗号化エンジンの完全性保証



- MEEは、Enclaveの内容に少しでも**改竄**が発生した場合、**完全性の保証**のためにそのマシンを**シャットダウン**する
 - 後述の通り、この仕様が逆に**DoS攻撃**に悪用される可能性がある

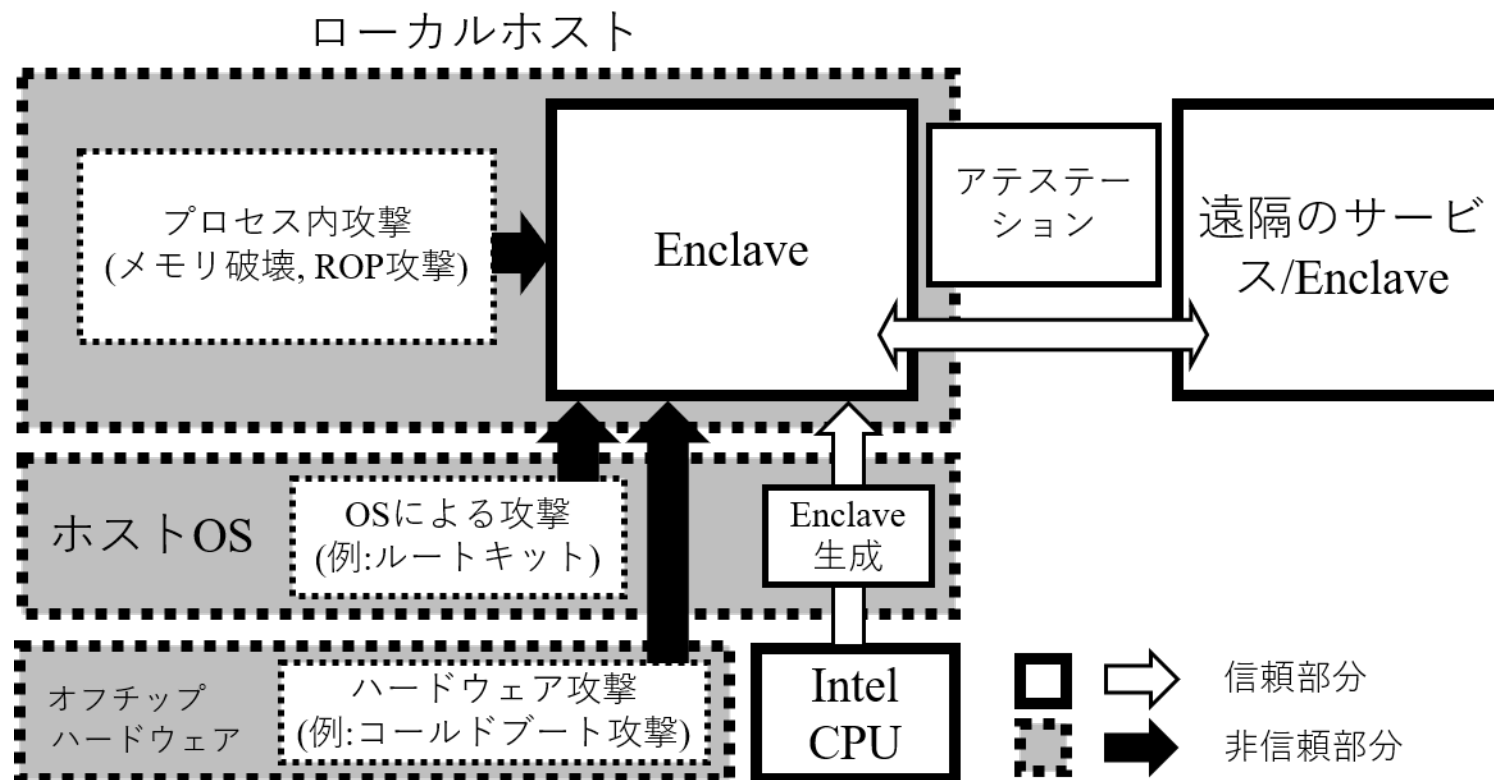


- **第3世代Xeonスケーラブルプロセッサ**では、MEEではなく**Intel TME-MK**を用いて**メモリ暗号化**が行われている
 - **TME-MK**: Total Memory Encryption – Multi-key
 - CPU1ソケットあたり最大**512GBのEnclave (EPC) サイズ**を確保できる
- **Enclaveサイズが大幅に増加**したのは**大きなメリット**ではあるが、MEEの持っていた**メモリ完全性検証能力**を**持たない**ため、EPCの**再生攻撃**に対して**耐性が無い**と議論になっている[14]
 - 再生攻撃の概念については攻撃編セクションで解説

SGXが想定する脅威モデル（1/4）



- 以下の図は、Graphene-SGX[4]の論文において記載されているSGXの脅威モデル図を和訳したものである：





■プロセス内攻撃

- 言い換えれば、**ユーザモード (非特権)** で動作する、**同一マシン上のプロセスによる攻撃**
 - メモリ破壊、ROP攻撃等
- Enclaveへの進入は**専用のEnclaveエントリポイント**に対する**専用のCPU命令 (EENTER ; 後述)** でしか行えないため、Enclave外のプロセスでメモリ破壊を行っても無意味
- …なのだが、**Enclave内のコードに脆弱性があるとこれに似た攻撃が実現してしまう**
 - 例 : Dark-ROP、SGX-Bleed (ゼミ後半で説明)
 - シーリング再生攻撃も広義にはこれに含まれる



■ OS・ハイパーバイザ・BIOSによる攻撃

- 言い換えれば、**root権限を有するような特権攻撃者**による攻撃
 - 例：ルートキット
- 特権を持っていようが、前述の**Enclave進入メカニズム**や**Enclave暗号化の牙城は崩せない**
 - この挑戦的な脅威モデルの想定は**SGXの大きな売りの1つ**
- 反面、この強力な脅威モデルにつけこまれ、コードやSGX自体の脆弱性への攻撃に、**普通ありえないような特権攻撃が使用されがち**
 - 例：Plundervolt、LVI、SGX-Step（ゼミ後半で説明）
 - 完全に攻撃の研究を成立させる上での口実



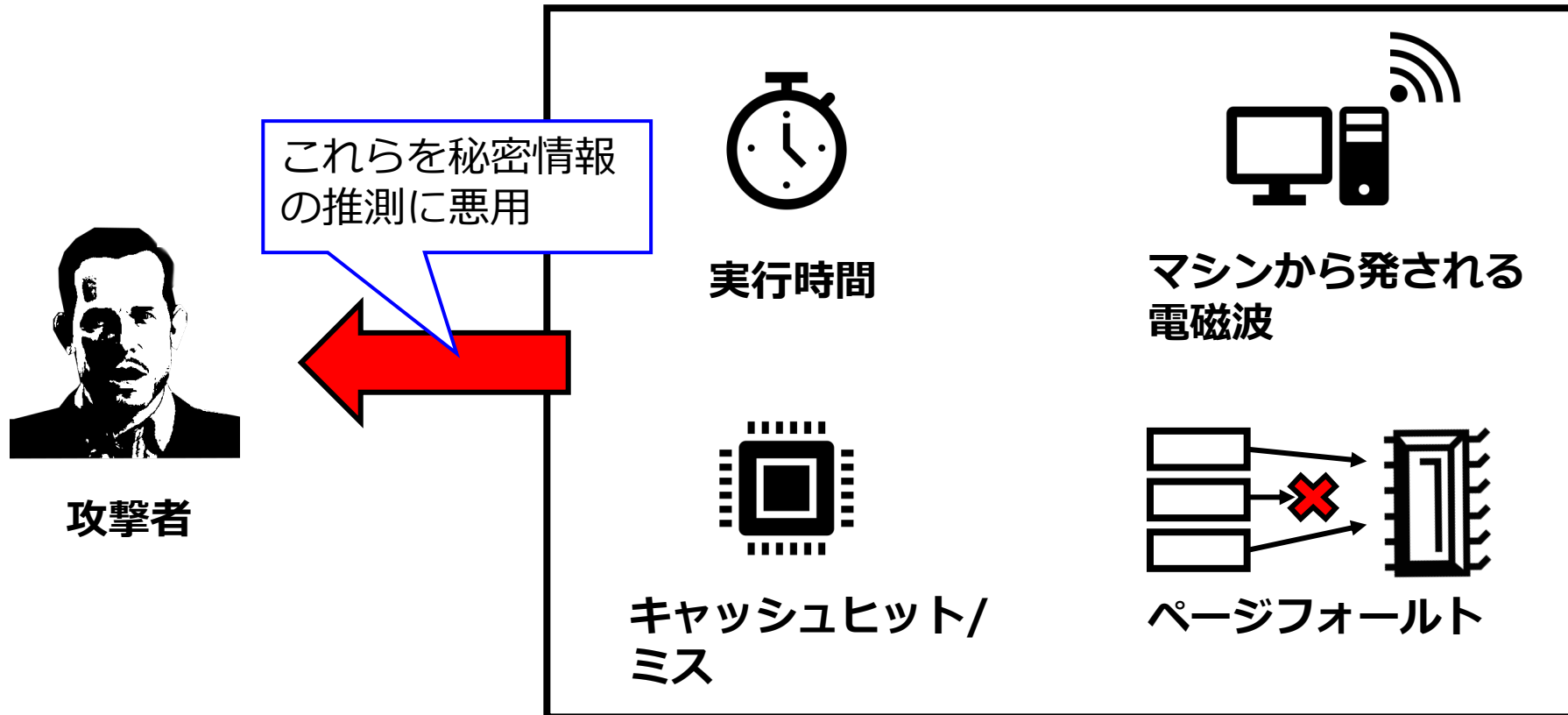
■ CPU以外のハードウェアへの攻撃

- 要するに**DRAMに対する物理的な攻撃**
 - 例：**コールドブート攻撃**（攻撃対象のマシンの電源が落ちた後、すぐにDRAMを取り外し冷却して別マシンに挿して起動すると、電荷が残留しているので攻撃対象マシンでのメモリ内容を盗聴できる）
- Enclaveは暗号化されているので、暗号文が得られるのみ
 - 基本的に**保護領域が平文**なTrustZoneやKeyStoneに対する**SGXの強み**
- ただし、MEEの完全性保証を悪用した**物理寄りのDoS攻撃**は成立し、また**CPU自体にバグがあると物理寄りの攻撃で秘密が漏洩する**
 - 例：SGX-Bomb（ロウハンマー攻撃）、Plundervolt（電圧降下攻撃）
 - 上記もゼミ後半で説明

SGXが対策できない攻撃 (1/3)



- **サイドチャンネル攻撃**対策を開発者の実装に丸投げしているのは有名
 - サイドチャンネル攻撃：秘密情報を直接解読するのではなく、その周辺環境から得られる情報を用いて**秘密情報を推測**する攻撃



SGXが対策できない攻撃 (2/3)



- お察しの通り、**SGXに対するサイドチャンネル攻撃は無数に存在する**
 - 例：Foreshadow、Controlled-Channel Attacks、SgxPectre、Branch Shadowing Attack、MDS (RIDL、Fallout、ZombieLoad)
- サイドチャンネル攻撃への**対策技術**も同様に多く存在する
 - 例：T-SGX、SGX-LAPD、DR.SGX、Zigzagger
- これらについてもゼミ後半で解説

SGXが対策できない攻撃 (3/3)

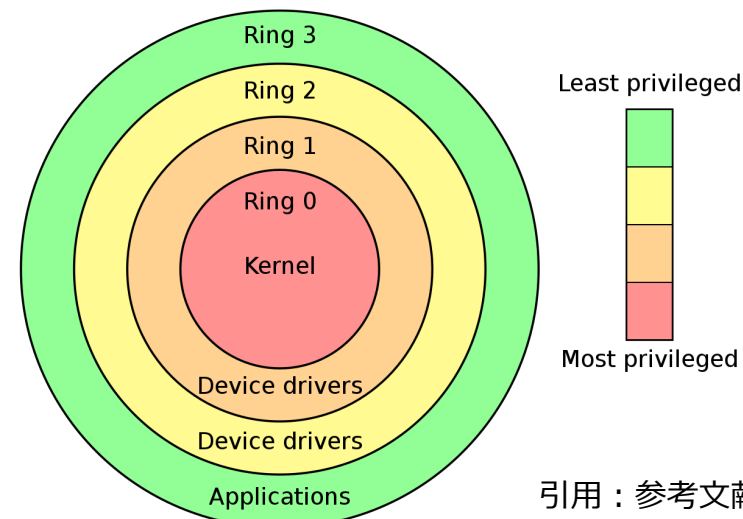


- SGXは**データの保護**は提供するが、コード安全性の保証は行わない
 - コードの保護も行わないし、バグ対策もしてくれない
- Enclaveコードに**実装上のミス（脆弱性）がある**事により実現する攻撃もやはり多い
 - 例：Dark-ROP、SGX-Bleed、Controlled-Channel Attacks
- **Rust-SGX**を用いると大幅にこのリスクは軽減できる
 - Rustでは危険なコーディングはコンパイル時に拒絶されるため
 - 今回のゼミはオーソドックスなC++でやります

Enclaveプログラムの実行権限



- Enclave内のコードは、**Ring-3権限（ユーザモード）**でのみ実行可能である
 - root権限のような特権モードでのEnclaveコード実行は不可
- Enclave内のコードが攻撃に必要なデータをEnclaveに隠しつつ攻撃を行う**マルウェア的な動作をさせないための措置**



リングプロテクションの概念図

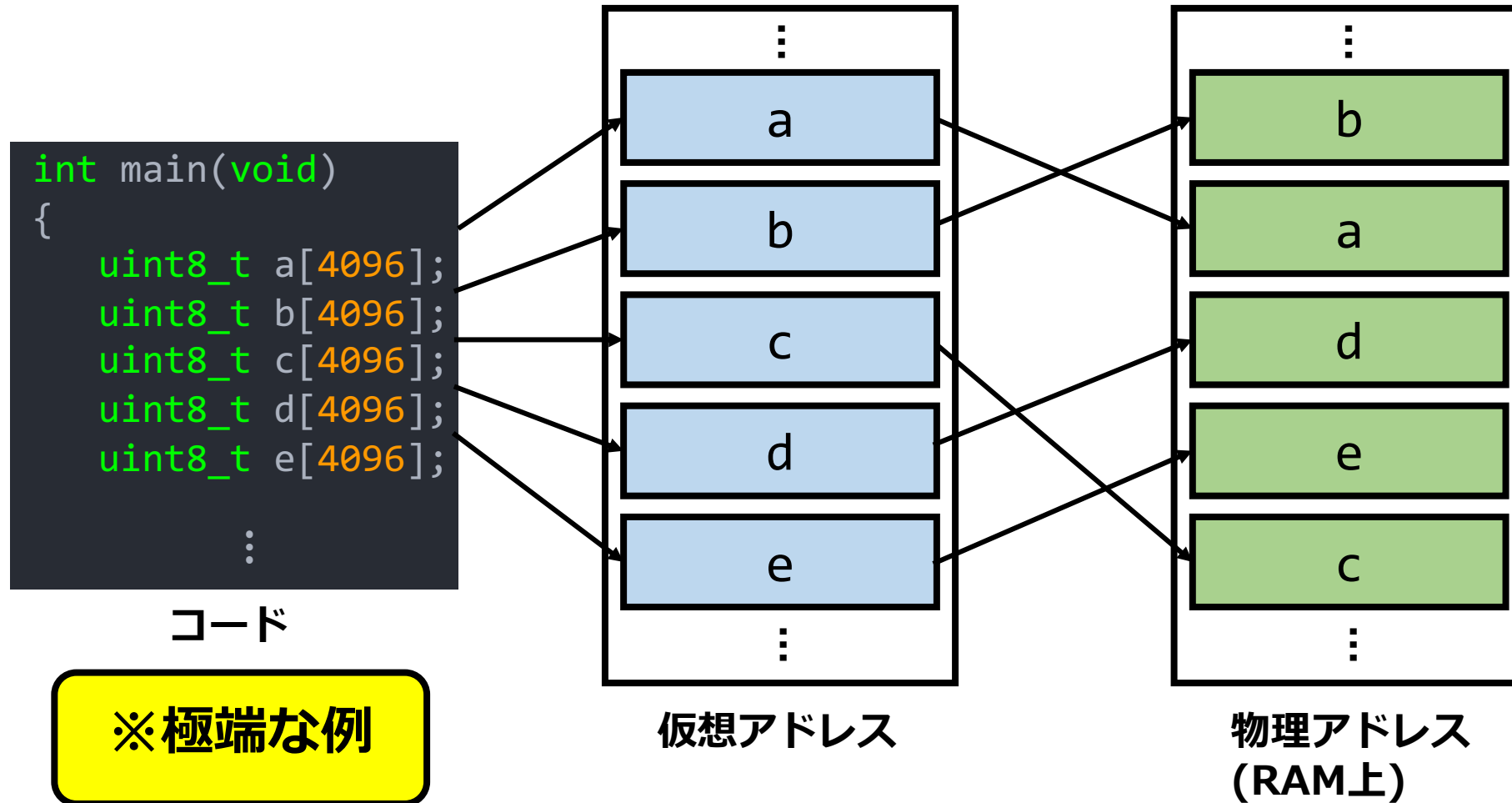


- よく**保護領域**の事を「**Enclave**」と呼ぶが、これは比較的概念的な呼称で、秘密情報を保持する**特別なメモリ領域**を**EPC** (**Enclave Page Cache**) と呼ぶ
- EPCは**物理アドレス空間上**のページであり、SGXの安全な秘密保持向けにCPUによって確保されている**プロセッサ予約メモリ** (**PRM**) という**物理アドレス空間**内にページ単位で生成される
 - PRM: Processor Reserved Memory
- **仮想アドレス空間**におけるEPCに対応する領域を**ELRANGE**という

(補足) ページング



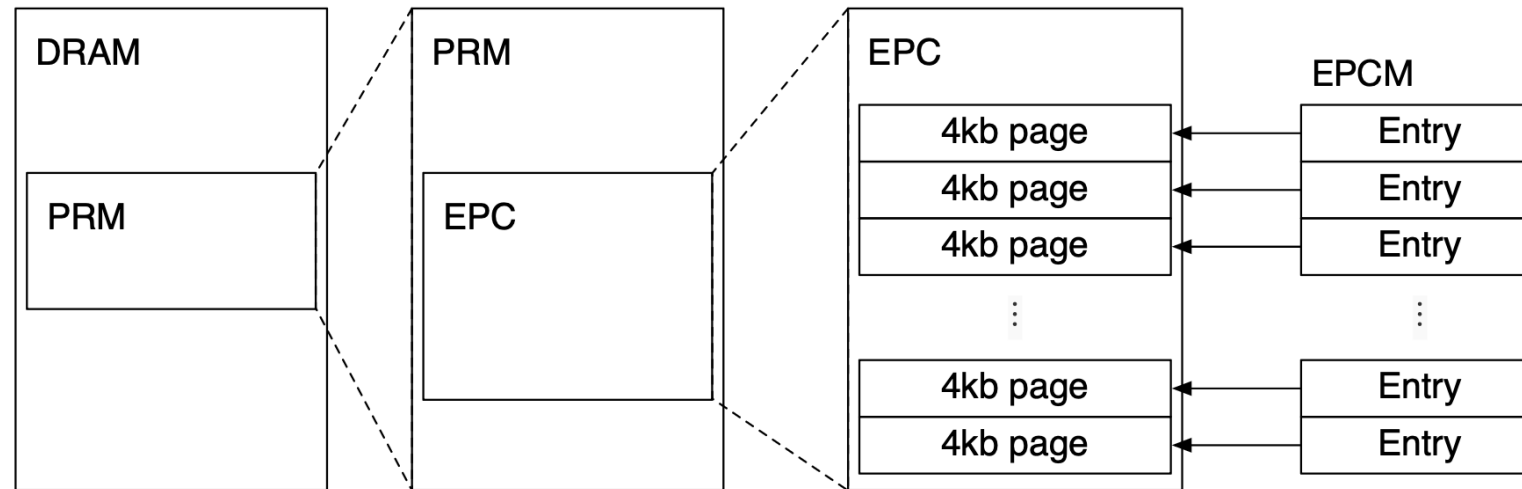
- 仮想記憶を実装するためのアルゴリズム



秘密情報を保持するメモリ領域 (2/3)



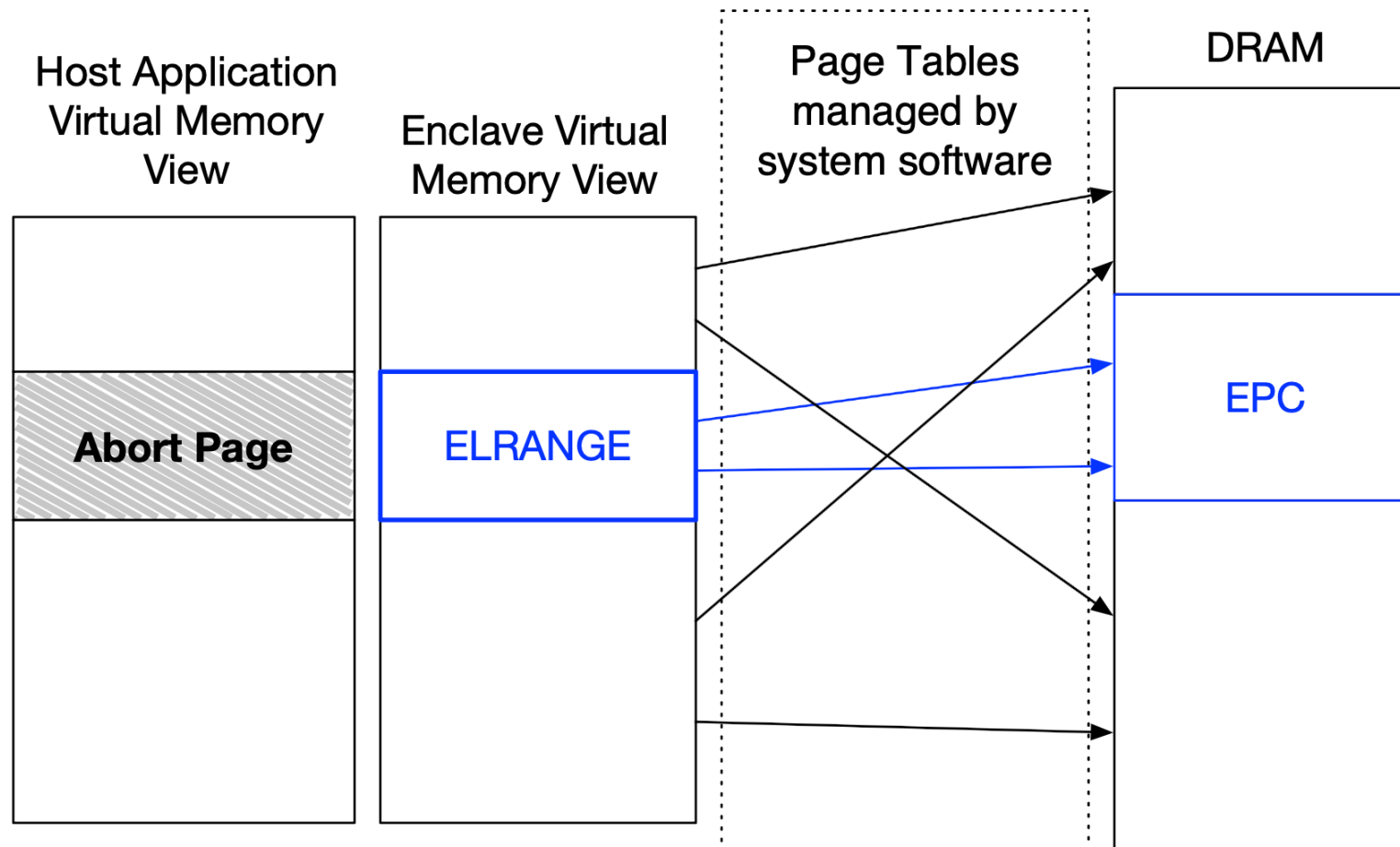
- PRMとEPCの関係図 (参考文献[6]より引用)



秘密情報を保持するメモリ領域 (3/3)



- EPCとELRANGEの関係図 (参考文献[6]より引用)



SGXを支える様々なデータ構造 (1/2)



- SGXは、EPCやその内部に生成されるデータ構造、及び特定の処理で使用されるその他のデータ構造により支えられている

データ構造名	機能
EPC (Enclave Page Cache)	メモリ上の保護領域。ユーザが自由に使える EPCサイズ上限は96MB
SECS (SGX Enclave Control Structure)	Enclave同一性に関するメタデータを保持 EPC内に存在
TCS (Thread Control Structure)	Enclave実行スレッドを管理。EPC内に存在
SSA (State Save Area)	AEX(*)時にEnclaveの状態を保持。EPC内に存在
PAGEINFO (Page Information)	EPCのアドレス情報やSECSを保持。EPC外に存在
SECINFO (Security Information)	ページタイプ等EPCのメタデータを保持。EPC外に存在
PCMD (Paging Crypto MetaData)	ページアウトしたEPCページの暗号的メタデータを保持。 特定のPAGEINFO内などに組み込まれる

(*)AEX : 非同期Enclave脱出。割り込みやフォールトに伴い、一時的に実行をEnclave外に移す処理

Enclaveを構成するデータ構造 (2/2)



(前ページの表の続き)

(*)LE : Launch Enclave。詳細は後述

データ構造名	機能
VA (Version Array)	退避したEPCページのバージョンを管理する配列。 EPC内に存在
EPCM (Enclave Page Cache Map)	CPUが特定のEnclaveに対応するEPCを追跡するための情報を保持。アーキテクチャ的な実体は不明
SIGSTRUCT (Enclave Signature Structure)	Enclaveのハッシュ値や署名者情報を保持。 Enclave初期化時等に使用
EINITTOKEN (EINIT Token Structure)	Enclaveの初期化 (EINIT) が許可された事を証明する トークン。LE(*)によって生成され、EINIT時に使用
REPORT (Report)	Enclave検証用の公開情報。アテステーションで使用
TARGETINFO (Report Target Info)	EREPORT命令において使用されるデータ構造。 レポート提出先Enclaveの同一性情報を含む
KEYREQUEST (Key Request)	EGETKEY命令にて、要求する鍵の種類や 追加パラメータを指定するデータ構造

Enclaveにまつわる特別なCPU命令 (1/4)



- Enclaveに関連する操作は、全て**特別なCPU命令**によって行われる
- この特別なCPU命令には2つ存在し、**カーネルレベルの命令がENCLS、ユーザレベルの命令がENCLU**である
 - ENCLSのSはスーパーバイザ (=OS)、ENCLUのUはユーザの意
- ENCLSやENCLUは、それぞれ引数に応じて様々な処理を執り行う
 - 厳密にはENCLSやENCLUはオペコードであり、EAX (アキュムレータ) の内容 (リーフ関数名) に応じて対応する処理を実行する
 - ENCLS[EADD]のように表現する事がある

Enclaveにまつわる特別なCPU命令 (2/4)



- ENCLSによって実行される各種処理（リーフ関数）は以下の通り：

リーフ関数名	機能
EADD	Enclave初期化前のロード処理時にEPCページの追加
EBLOCK	EPCページをブロック状態（使用禁止状態）にする。EWB前などに使用
ECREATE	Enclaveを生成する。具体的には、未使用のEPCを新EnclaveのSECSに変換する
EDBGGRD	デバッグモードのEnclaveのEPCページを平文状態で読み取る
EDBGWR	デバッグモードのEnclaveのEPCページに平文でデータを書き込む
EEXTEND	EADDに伴い、その時点でのMRENCLAVEを取得（更新）する
EINIT	Enclaveを初期化する。ECREATE後に使用
ELDB	EWBで退避されたEPCをブロック状態としてロードする
ELDU	EWBで退避されたEPCを非ブロック状態（使用可能状態）として読み込む
EPA	前述のデータ構造VA（Version Array）を追加する
EREMOVE	EPCからページを削除する（そのページをECREATE前の状態にする）
ETRACK	EBLOCK後、論理プロセッサが関連するTLBをフラッシュしたかを追跡開始する
EWB	EPCページを暗号化された状態で非PRM（Enclave外）メモリに退避させる

Enclaveにまつわる特別なCPU命令 (3/4)



- ENCLUによって実行される各種処理（リーフ関数）は以下の通り：

リーフ関数名	機能
EENTER	Enclaveに進入する。後述のECALLに対応
EEXIT	Enclaveから脱出する。Enclaveコードからのreturnに対応
EGETKEY	SGXで使用する各種鍵（例：レポートキー）を導出する
EReport	そのEnclaveのメタデータであるREPORT構造体を取得
EResume	AEXによりEnclaveから一時的に脱出した際、Enclaveに再進入する時に使用する

Enclaveにまつわる特別なCPU命令 (4/4)

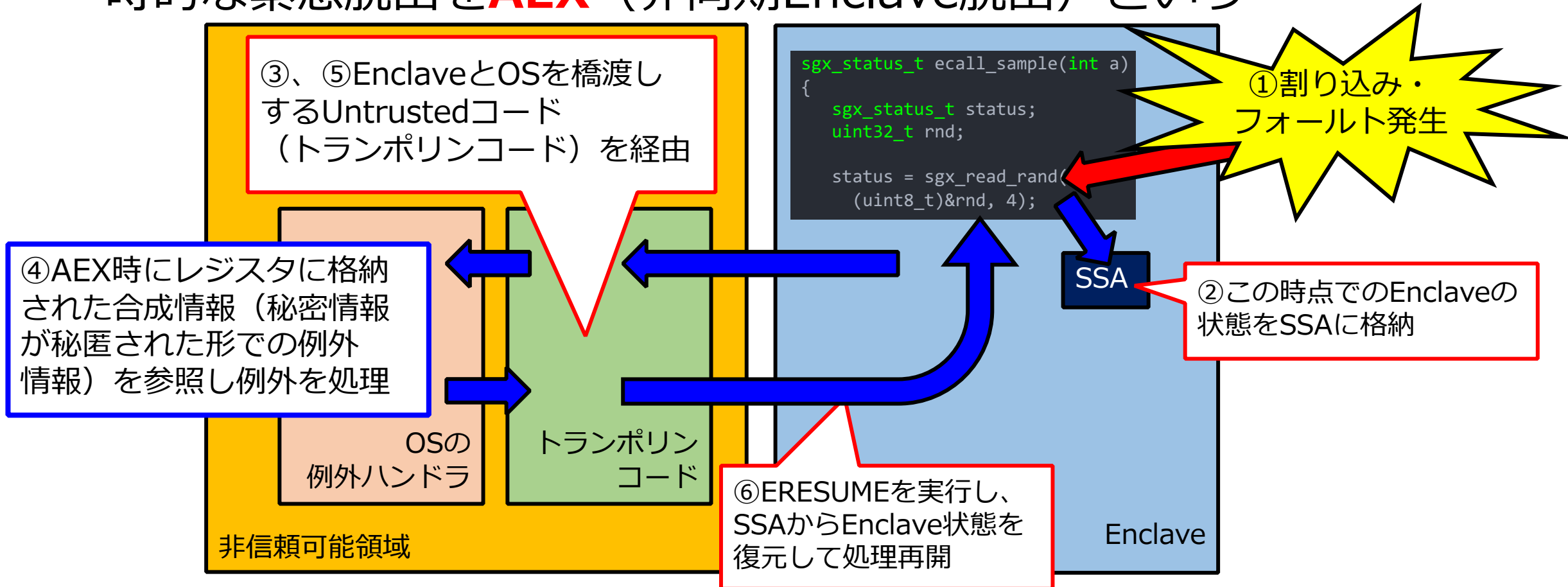


- これらのCPU命令を直接意識する事はあまりないが、ENCLUのリーフ関数はSGXプログラミングで呼ぶAPIに密接に関わる
- ELDUとEWBは、キャッシュ階層からデータを漏洩させる攻撃の議論の際に頻出
 - 例：Foreshadow、ÆPIC Leak
- ERESUMEはAEXの議論が必要な場合に頻出
 - 例：SGX-Step

AEX (Asynchronous Enclave Exit)



- Enclaveからのリターン、即ちEEXITによる**通常のEnclave脱出**（同期脱出）ではなく、**割り込みやフォールト**が発生した際の**一時的な緊急脱出をAEX**（非同期Enclave脱出）という



Enclaveの同一性に関するハッシュ値 (1/2)



- Enclaveの**動作定義**やEnclaveへの**署名者**の同一性を検証するために使用される、2つのハッシュ値 (Measurement Register; 測定記録値) が存在する
 - 後のセクションで説明するが、Enclave製作者はそのEnclaveの動作定義イメージに対して署名してSGXマシンにデプロイし、実行時にその署名を検証する
- Enclaveの**動作定義** (コード) に対するハッシュ値を**MRENCLAVE** (Measurement Register of ENCLAVE)、**署名者**に対するハッシュ値を**MRSIGNER** (Measurement Register of SIGNER) と呼ぶ

Enclaveの同一性に関するハッシュ値 (2/2)



- **MRENCLAVE** : **Intelの人間**によれば、以下の要素に依存する[10]
256bit SHA-2ハッシュ値
 - 署名前のEnclaveイメージ (≒Enclaveのソースコード)
 - Enclave設定用XMLファイル (Enclave.config.xml) の一部エントリ
 - SDK、PSWのバージョン
 - Enclave署名ツールであるsgx_sign (←**真偽未検証**)
 - Intelは**Enclaveへの署名鍵もMRENCLAVEに影響する**と言っているが、**ほぼ嘘で確定**
- **MRSIGNER** : **Enclave署名鍵** (3072bit RSA鍵) の**RSAモジュラス**に対する256bit SHA-2ハッシュ値

Enclaveの2つのモード



- Enclaveには、**Production Mode Enclave**（製品版Enclave、本番Enclave等と呼ぶ）と**Debug Mode Enclave**（デバッグ版Enclave）が存在する
- 製品版Enclaveは、**内部の秘密情報が完璧に保護される**が、その実行には原則的に**Intelへのライセンス申請が必要**となる
<https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/request-license.html>
- デバッグ版Enclaveは**特に申請等はいらない**が、デバッグのため**EDBG RD命令**や**EDBG WR命令**にて**平文でEnclaveを読み書き出来る**



- SGXの大部分は**ハードウェア**と**マイクロコード**で実装されているが、**特定の重要な機能**に関しては**専用のEnclave**を用いて実装している
 - マイクロコード：特にCISCプロセッサにおいてCPU命令を実行するためにプロセッサ内で使用される、**より単純な命令**
 - ある**分子**（複雑なCPU命令）を構成する**原子**（マイクロコード）のイメージに近い
- この専用のEnclaveを**Architectural Enclave (AE)** という



- **Provisioning Enclave (PvE)**

マシン初使用時やTCB Recoveryの際に、そのマシンのCPUやSGX関連コンポーネントが正当であることを確認後、Attestationキーをマシンにデプロイする「**プロビジョニング**」処理で使用。本ゼミのProvisioningセクションで詳述。

- **Quoting Enclave (QE)**

リモート・アテステーション時に、**REPORT構造体**に対してAttestationキーで署名し**QUOTE構造体**を生成する。本ゼミのAttestationセクションで詳述。



- **Launch Enclave (LE)**

特に**製品版Enclave**において、そのEnclaveがIntelによって起動を許可されているか (**ライセンス**) を確認し、問題なければ **EINITTOKEN構造体**を生成する。

ライセンスの判断には、Enclave製作者の署名情報に紐づく **MRSIGNER**値を用いる。

お察しの通り**LEはIntelの利権のためにのみ存在**しており、否定的な意見[6][10]も多かったためか、**現在ではDeprecated (非推奨)** となっている。



- **Reference Launch Enclave (ref-LE)**
LEのようなIntelの利権にまみれた実装が顰蹙を買ったため、他の承認者による起動許可が出来るようになった**Flexible Launch Control (FLC)** で使用されるバージョンのLE。
基本的な機能はLEと同様だが、MRENCLAVEベースでも許可可能。
- **Platform Service Enclave (PSE)**
モニタックカウンタや**信頼可能時間ソース**といった、信頼性保証のための機能を提供するAE。
が、**Linux-SGXではv2.7辺りから突然使用不可になっている**。
iclsClientというものをLinuxで使えなくなったのが理由らしい。



- 他にも、Provisioning Certification Enclave (PCE) や Quote Verification Enclave (QvE) 等があるが、これらは **ECDSA Attestation**の文脈で使用される
 - 本ゼミでは**EPID Attestation**を採用し解説するため、これらについての詳細に関しては割愛する

LEを取り巻く現在の状況



- LEについてはあまりにも批評が多かったためか、Linux向けSGXドライバの内、**DCAP**（データセンターマシン向け）**ドライバ**と**in-kernelドライバ**では、もはや**FLCすら不要**になっている[15]
 - **in-kernelドライバ**：Linuxカーネル内に元々内包されているドライバ。LinuxのSGXドライバは、Linuxカーネル5.11以降からこれになっている
- よって、上記の環境であれば**Enclaveの起動**に関しては**自動的に許可**されるようになっている
 - 特殊な鍵へのアクセス権を要求するEnclave（一部AEなど）については別

Intel SGXを用いた実世界での製品



- SGXを使用している中で、この世で最も全世界に普及しているであろう製品がCyberLink社製の「**PowerDVD**」
- **Ultra HD Blu-rayのDRM（デジタル著作権管理）処理**にSGXのEnclaveを用いている
 - このDRMアルゴリズムは**AACS 2.0**として策定されており、**その具体的な仕様は不明**
- …**なはずだったのだが、SGX Fail**という論文により攻撃され、**AACS 2.0の仕様が暴露**されてしまった
 - SGX Failについては本ゼミの「SGX Fail」セクションで説明

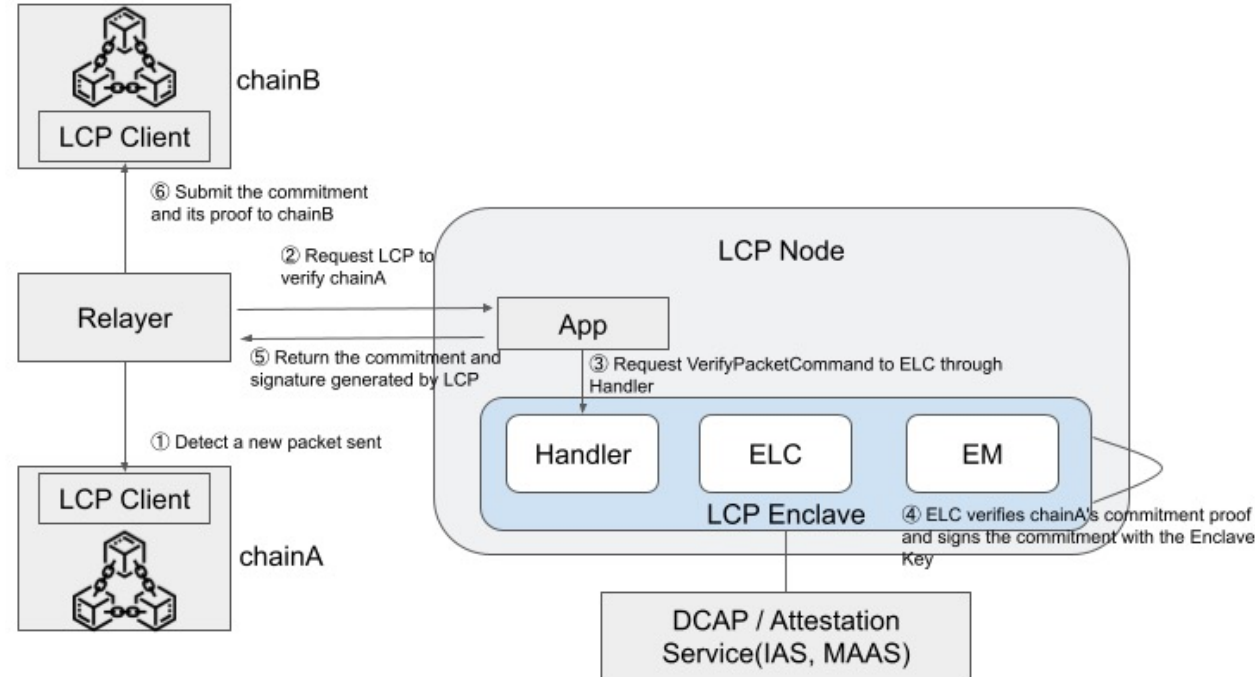


- SGXの**動作に対する完全性保証** (マリシャスモデルへの対応) に着目しているタイプの製品
 - 本ゼミ講師がSGX周りで技術的に支援しているプロダクト
 - Remote AttestationにおけるMRENCLAVEの検証が要
- ブロックチェーンにおいて、**異種チェーン同士**で通信する (**IBC**) 際の**検証処理**を、**LCPがSGX Enclaveを用いて肩代わり**する
- これにより、チェーン当事者が**検証用ロジック** (Light Client) を**実装する手間を省き**、検証に伴う**手数料も大幅に抑える**事が出来る
 - 手数料は検証の実行ステップごとに加算されるが、LCPに委託すればLCPとの検証結果の送受信だけで済み、とても安上がりになる

LCP (Light Client Proxy) (2/2)



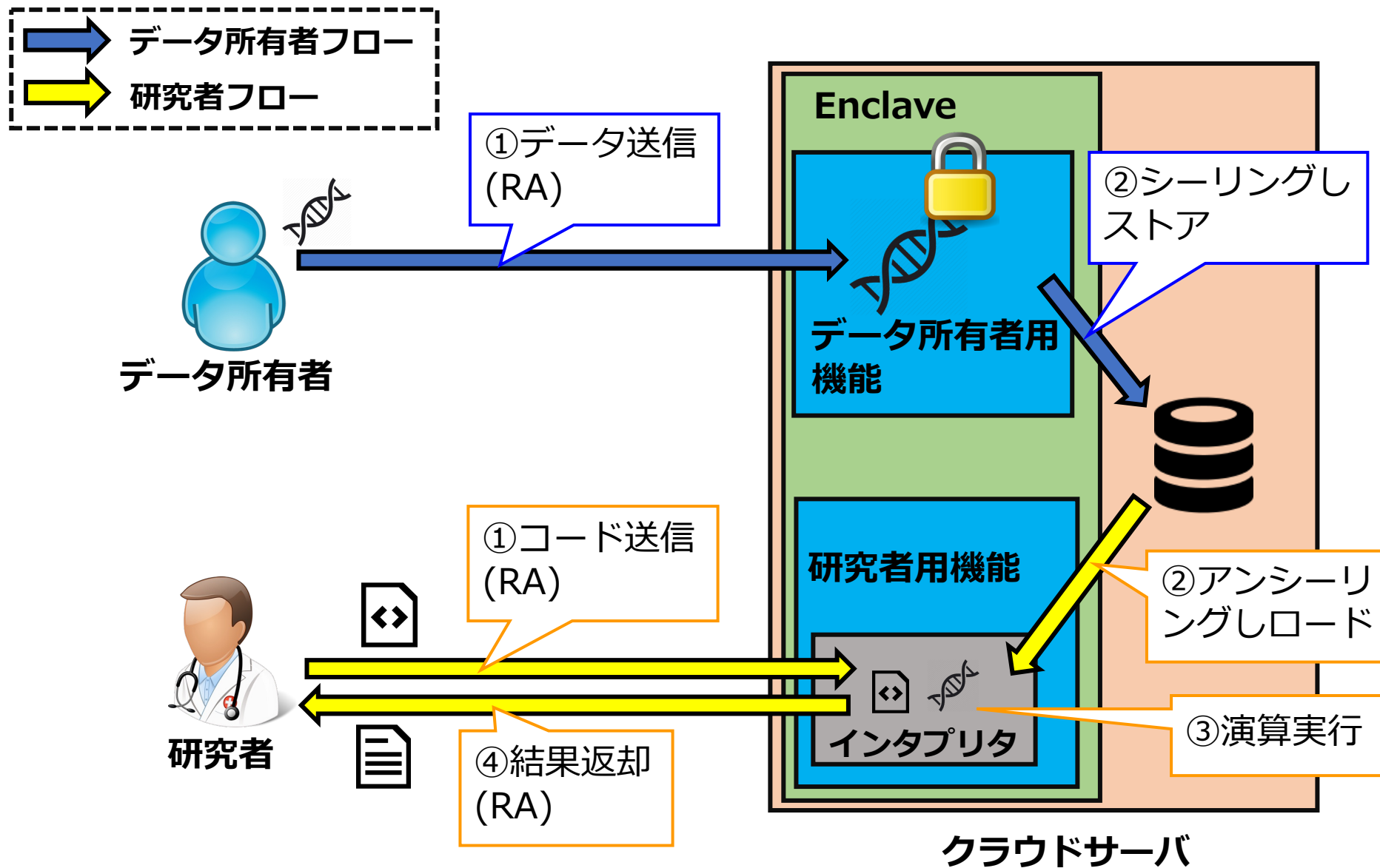
- Remote Attestation (本ゼミのAttestationのセクションで解説) でMRENCLAVEを検証する事で、Enclaveが意図している動作のものである事を確認し、それによる署名がされた検証結果であれば信頼可能であると見なせる





- 本ゼミ講師が未踏で開発したSGXベースのクラウドシステム
 - Bioinformatic Interpreter on SGX-based Secure Computing Cloudの略
 - 全く普及してはいないが、まあ私のゼミなので…
- Enclave内で**インタプリタを動作**させる事で、後に説明する**SGXSDKの魔の手**に**苛まれず**に、SGXの保護機能の恩恵に預かりながら秘密計算を含む様々な処理を行える
- BI-SGX紹介サイトのリンク：<https://bi-sgx.net/>

BI-SGX (2/3)





- 当時所属していた研究室の都合上、生命情報解析をメインに置いてはいるが、基本的に色々な処理を手軽にSGX上で行える

```
status = sgx_ra_proc_msg2(ra_ctx, eid,  
    sgx_ra_proc_msg2_trusted,  
    sgx_ra_get_msg3_trusted, msg2,  
    sizeof(sgx_ra_msg2_t) + msg2->sig_rl_size,  
    &msg3, &msg3_sz);
```

```
status = sgx_rijndael128GCM_decrypt(&sk_key,  
    (uint8_t*)data_cipher, cipherlen,  
    data_plain, p_iv, p_iv_len, NULL, 0,  
    &tag_t);
```

```
public sgx_status_t encrypt_store_status(  
    sgx_ra_context_t context,  
    size_t store_flag,  
    [in, out, size=12]uint8_t *p_iv,  
    [in, out, size=16]uint8_t *tag,  
    [in, out, size=10000]uint8_t *result_cipher,  
    [out]size_t *res_len);
```

⋮

SGX公式開発ライブラリ (40000行)

```
func main()  
    a = edist("dataset0")  
    println "result: ", a  
end
```

Qliphoth (4行)

**アウトプットプライバシー
を保護しつつ、劇的な
負担削減を実現**

本セクションのまとめ



- これを一気に覚えるのは相当しんどいはずなので、必要になったら見返すくらいの温度感で良い
- 何ならSGXプログラミングをするだけであればここまで知る必要はあまりない

参考文献 (1/2)



- [1] "Intel SGX Details? – Stack Exchange", 2023/6/4閲覧,
<https://security.stackexchange.com/questions/136993/intel-sgx-details>
- [2] "Intel Unleashes Broadwell-E Core i7-6950X, Core i7-6900K and Core i7-6800 Series Processors – Next-Gen Extreme Edition Processor For \$1700 US", 2023/6/4閲覧,
<https://wccftech.com/intel-broadwell-e-core-i7-6950x-launch/>
- [3] "A Memory Encryption Engine Suitable for General Purpose Processors", Shay Gueron,
<https://eprint.iacr.org/2016/204.pdf>
- [4] "Graphene-SGX: A Practical Library OS for Unmodified Applications on SGX", Chia-Che Tsai et al., <https://www.usenix.org/system/files/conference/atc17/atc17-tsai.pdf>
- [5] "リングプロテクション - Wikipedia", 2023/6/4閲覧,
<https://ja.wikipedia.org/wiki/%E3%83%AA%E3%83%B3%E3%82%B0%E3%83%97%E3%83%AD%E3%83%86%E3%82%AF%E3%82%B7%E3%83%A7%E3%83%B3>
- [6] "Intel SGX Explained", Victor Costan & Srinivas Devadas, <https://eprint.iacr.org/2016/086.pdf>
- [7] "Overview of Intel® Software Guard Extensions Instructions and Data Structure", Intel,
<https://www.intel.com/content/dam/develop/external/us/en/documents/overview-of-intel-sgx-instructions-and-datastructures.pdf>
- [8] "Exception Handling in Intel® Software Guard Extensions (Intel® SGX) Applications", Intel,
<https://cdrdv2-public.intel.com/671544/exception-handling-in-intel-sgx.pdf>

参考文献 (2/2)



- [9]“Communication between Architectural and Application Enclaves – SGX 101”, 2023/6/4閱覽, <https://sgx101.gitbook.io/sgx101/sgx-bootstrap/enclave/interaction-between-pse-and-application-enclaves>
- [10]“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
- [11]“Question about uniqueness of MRENCLAVE – Intel Community”, 2023/6/4閱覽, <https://community.intel.com/t5/Intel-Software-Guard-Extensions/Question-about-uniqueness-of-MRENCLAVE/m-p/1159777>
- [12]“Intel SGX入門 - SGX基礎知識編”, 自己引用, <https://qiita.com/Clifford/items/2f155f40a1c3eec288cf>
- [13]“Intel Powers New Azure Confidential Computing VMs”, 2023/7/27閱覽, <https://www.intel.com/content/www/us/en/newsroom/news/intel-powers-new-azure-confidential-computing-vm.html>
- [14]“Towards TEEs with Large Secure Memory and Integrity Protection Against HW Attacks”, Pierre-Louis Aublin et al., <https://systex22.github.io/papers/systex22-final15.pdf>
- [15]“HW_Release mode Intel SGX applications without whitelisted keypair over sgx_lc enabled processor”, 2023/7/27閱覽, <https://community.intel.com/t5/Intel-Software-Guard-Extensions/HW-Release-mode-Intel-SGX-applications-without-whitelisted/td-p/1394136>