Computer Security Symposium 2020
26 - 29 October 2020

RISC-V TEEを強化するためのSecure CoProcessorとそれを活用するソフトウェア

須崎 有康[1,2]　中嶋 健太[1]　大居 司[1]　永田 貴彦[1]　菊池 正史[1]　塚本 明[2]　宮澤 慎一[1,3]
磯部 光平[1,3]　伊藤 大輔[1]　木村 貞弘[1,4]　安達 浩次[1,4]　高橋 睦史[1,4]

概要：近年 TEE (Trusted Execution Environment) の活用が多くなっているが，厳密な鍵や証明書の管理を行うためには信頼の基点 (Root of Trust) となるハードウェアが必要である．我々はオープンアーキテクチャの RISC-V を使い，Root of Trust を実現する 32bit RISC-V Secure CoProcessor，および TEE を実行する 64bit RISC-V を組みわせた Trusted-RV を開発した．このハードウェア詳細を述べると共に，TEE を活用するソフトウェアについて述べる．ソフトウェアとしては (1) TA(Trusted Application) プログラミングのための GlobalPlatform Internal API，(2) TA の Install/Update/Delete を行う TEEP(Trusted Execution Environment Provisioning) プロトコル，(3) ハードウェアおよび TA バイナリの認証を行う Remote Attestation を提供し，機密処理を安全に実行する．

キーワード：RISC-V，TEE (Trusted Execution Environment), セキュアコプロセッサ, 信頼の基点，リモートアテステーション, TEEP (Trusted Execution Environment Provisioning)

Secure CoProcessor and Support Software for RISC-V TEE

Kuniyasu Suzaki[1,2]　Nakajima Kenta[1]　Tsukasa Oi[1]　Takahiko Nagata[1]　Masashi Kikuchi[1]
Akira Tsukamoto[2]　Shinichi Miyazawa[1,3]　Kohei Isobe[1,3]　Daisuke Itoh[1]　Sadahiro Kimura[1,4]
Adachi Koji[1,4]　Chikafumi Takahashi[1,4]

Abstract: TEE (Trusted Execution Environment) becomes popular, but secure key/certificate management depends on a "Root of Trust" hardware. We developed a Trusted-RV which has 64bit RISC-V with 32bit RISC-V Secure CoProcessor. The 64bit RISC-V works as a main CPU, which has a TEE mechanism. The 32bit RISC-V Secure CoProcessor works as a Root of Trust to support a critical processing on the 64bit RISC-V. This paper describes the detail of Trusted-RV and TEE supporting software: 1) GlobalPlatform Internal API for TA (Trusted Application) programming, 2) TEEP(Trusted Execution Environment Provisioning) protocol to install/Update/Delete a TA, and 3) Remote Attestation to authenticate the platform and TA binary.

Keywords: RISC-V, TEE (Trusted Execution Environment), Secure CoProcessor, Root of Trust, Remote Attestation, TEEP (Trusted Execution Environment Provisioning)

NSITEXE,Inc
1 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
Technology Research Association of Secure IoT Edge Application Based on RISC-V Open Architecture (TRASIO)
2 国立研究開発法人産業技術総合研究所
National Institute of Advanced Industrial Science and Technology (AIST)
3 セコム株式会社
SECOM CO.,LTD.
4 株式会社エヌエスアイテクス

1. はじめに

近年の OS は多量なデバイスに対するドライバや高機能なアプリケーションに対する拡張機能などで大規模で複雑になっている．システムソフトウェアのバグの発生は，そのサイズや複雑さに依存すること [1], [2], [9], [22] が分かっており，バグから起因する脆弱性から逃れるこ

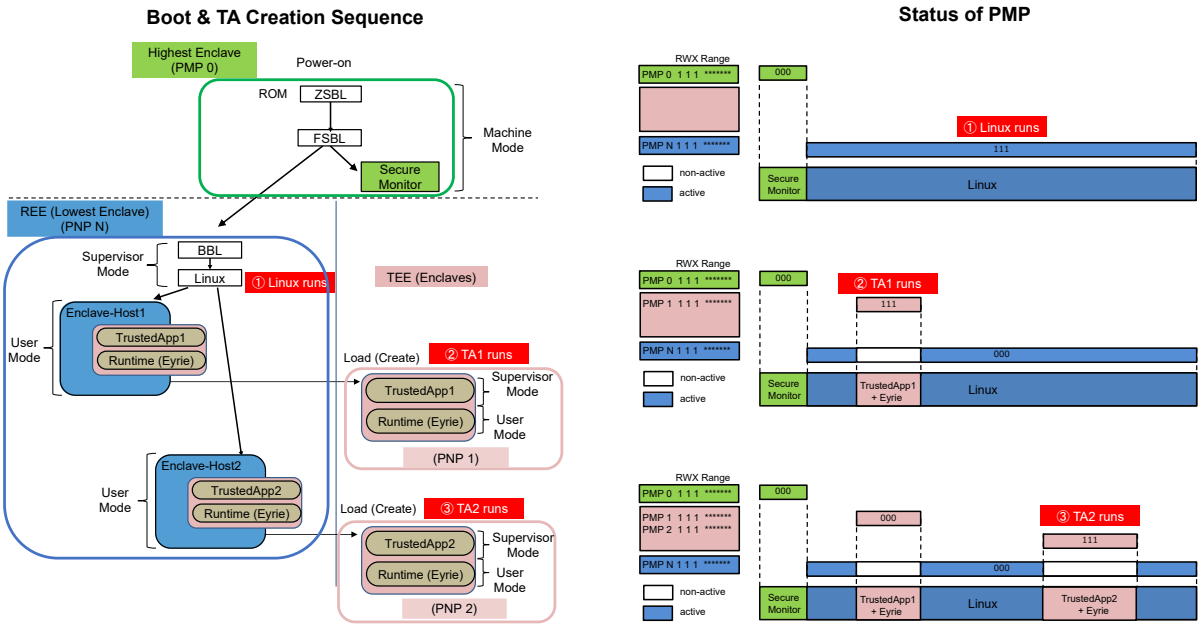

図 1 Keystone を想定したブートシーケンスおよび Trusted Application の実行．左図は必要となるソフトウェアコンポーネントを示し，右図では PMP の動作を示す．

とができない．この問題を回避するために近年の CPU では OS とは隔離した実行環境を提供する Trusted Execution Environment (TEE) の機能を有する．例えば，Intel CPU では SGX (Software Guard Extensions)，AMD では SEV (Secure Encrypted Virtualization)，Arm CPU では TrustZone が提供されている．

TEE の機能のある CPU では，通常の OS は Rich Execution Environment または Normal World と呼ぶ実行環境で動作し，必要に応じて隔離実行が必要な TA (Trusted Application) を TEE(Secure World あるいは Enclave とも呼ばれる) で処理する．TEE はそのアーキテクチャにより TEE 内に Trusted OS を走らせるもの (例えば Arm TrustZone) やユーザのアプリケーションの一部として走らせるもの (例えば Intel SGX) など色々な実装形態がある．個々の TEE については解説論文 [27] を参考にされたい．

残念ながら TEE では基本的に隔離実行を提供するのみであり，そこで使われる暗号鍵の管理には別の機能が要求される．個々のデバイス内で提供する場合は『信頼の基点 (Root of Trust)』と呼ばれるハードウェア機能がその役割を果たす [10], [12]．例えば Intel SGX に対する Intel ME(Management Engine) [7], [8] であり，AMD SEV に対する PSP(Platform Security Processor) [4] である．これらは商用であり，実装詳細が明らかでない．しかし，信頼の基点はセキュリティの根幹の関わっており，脆弱性検証の対象と考える．

我々は信頼の基点の実装を検証可能にするために，オープンアーキテクチャである 64bit RISC-V の TEE に必要な信頼の基点として 32bit RISC-V をベースとした Secure Coprocessor を作成した．この二つを一体化したプラットフォームを Trusted-RV と名付け，TEE を活用できる下記の機能を開発している．

(1) TA プログラミングのための GlobalPlatform Internal API 実行環境
(2) TA の Install/Update/Delete を行う TEEP(Trusted Execution Environment Provisioning) プロトコル処理できるソフトウェアコンポーネント
(3) プラットフォームハードウェアおよび TA バイナリの認証を行う Remote Attestation

本論文では Trusted-RV の詳細を述べると共に，Trusted-RV の TEE を活用するソフトウェアについて述べる．

2. バックグラウンド

本章では本論を理解する上で必要となる RISC-V，RISC-V TEE，信頼の基点，TA のプログラミング，TEEP，リモートアテステーションの技術の概要を説明する．

2.1 RISC-V

RISC-V は UC Berkeley の Krste Asanović 教授が始めた命令セットアーキテクチャ (ISA: Instruction Set Architecture) が公開されている CPU である．現在その仕様は RISC-V Internatinal において管理され，特権命令，非特権命令の他にデバッグなどの仕様が公開されている [21]．RISC-V は 32/64/128 bit の 3 つのワード幅の仕様があり，Machine(M) モード，Supervisor(S) モード，User(U) モードを提供する．組込みシステムなどを考慮してモードの組み合わせは M のみ，M+U，M+S+U の構成を取ることができる．また，メモリの境界を区分けする PMP(Physical Memory Protection) の機能も有する．後述する Keystone

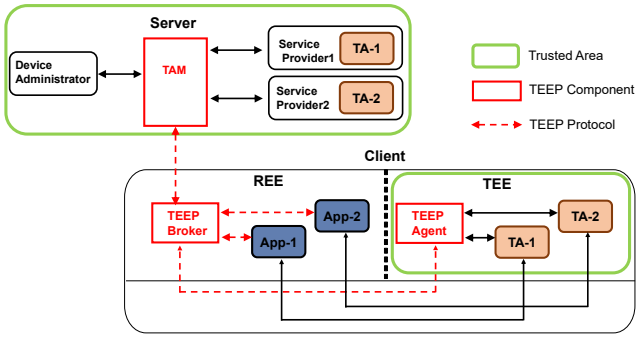

図 2　TEEP の構成図

ではこの PMP の機能を活用して TEE が実装されている．

2.2　RISC-V の TEE

RISC-V はそのオープン性を生かして TEE 技術開発が盛んである．論文発表済みのものとしては Sanctum [6], TIMBER-V [26], MI6 [3], Keystone [17] などがある．商用の TEE としては組込み用に HexFive 社の MultiZone [11] がある．また，RISC-V International には TEE Working Group がある．ここでは特定の TEE 実装を決めるわけではく，共通で使われる PMP などの仕様策定を行っている．我々の TEE 技術開発は技術公開が積極的な UC Berkeley の Keystone をベースとして進めている．

図 1 に Keystone を有効にした場合のブートシーケンスと TA の実行の様子 (図左) とその時の PMP の状態 (図右) を示す．Keystone は RISC-V International で規格化されている PMP (Physical Memory Protection) の機能を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を管理するメカニズムである．各レジスタはメモリ領域とそのパーミッションを保持する．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 レジスタは TEE のための Enclave に使われる．

起動後，Secure Monitor と Linux のみが実行している状態を図 1 右上段に示す．この状態ではほとんどのメモリを OS(Linux) が保持している．図中でパーミッションが有効なっている PMP の領域 (図中ではすべて 111) のみがアクセス可能で，他の領域にはアクセスできない．他の PMP のパーミッションは無効 (図中ではすべて 000) として扱われる．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 1 右中段に 1 つの Enclave1 が生成されたのちに，その Enclave1 がアクティブな状態を示す．図 1 右下段では Enclave2 が生成され，アクティブな状態を示す．それぞれの Enclave は処理が完了すればメモリが解放されて，Linux に戻される．

2.3　信頼の基点

信頼の基点 (Root of Trust) は鍵などの機密情報保存，およびその処理をするためのハードウェアである．このような機能が必要なことは論文 [12] でも求められている．GlobalPlatform では TEE で求められる信頼の基点について定義 [10] している．

信頼の基点の実装はサイドチャネル攻撃に対する耐タンパであることが望ましいが，物理攻撃がない環境（例：クラウド環境なので物理的侵入はなし）ならば，この条件を外すことができる．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，機密情報保存の他に鍵生成，乱数生成などの機能を有する．

TEE 環境に対してこのような信頼の基点を使う例は多くある．例えば Intel SGX では Intel ME (Management Engine) [7], [8] に一部の処理を依存している．また，AMD の SEVC では PSP (Platform Security Processor) [4] でも同様である．RISC-V においては Rambus 社の CryptoManager や Silex Insight 社の Secure Root of Trust が信頼の基点を提供する．Secure Root of Trust については Andes 社の AndesCore N22 に採用されていることが公開されている．

2.4　TA のプログラミング

TEE のプログラミングについては CPU の開発ベンダーやソフトウェアベンダーがそれぞれの SDK(Software Development Kit) を提供している．例えば，Intel SGX のためには Intel から SGX SDK [13] を提供し，Microsoft からは OpenEnclave SDK [18] が提供されている．RISC-V Keystone でも Keystone SDK [16] が提供されている．但し，これらはそれぞれの TEE に依存しており，相互にインターオペラビリティがない．

IC カードのセキュリティ仕様を定義している GlobalPlatform では，TEE についても幾つかの仕様を出している．その一つが GP TEE Internal API である．現在，多くのスマートフォンで使われており，TA のプログラミングとしては最も普及している．GP TEE Internal API は 130 の API を含み，それらは 29 のカテゴリーに分けられている．この API 名は具体的な暗号アルゴリズム名を含まず，処理の概要のみを表す名前になっている．暗号アルゴリズム名は引数として分離することで暗号が拡張されても対応できるようになっている．

2.5　TEEP

TEEP (Trusted Execution Environment Provisioning) [23] とは TEE 内の TA の install/update/delete を行うためのプロトコルであり，IETF でその仕様策定が進められている．

図 2 に TEEP の構成を示す．TEEP を実現するために，

表 1　実装された GlobalPlatform TEE Internal API

Category	Architecture	API name
Random Number	Dependent	TEE_GenerateRandom
Time	Dependent	TEE_GetREETime, TEE_GetSystemTime
Secure Storage	Dependent	TEE_CreatePersistentObject, TEE_OpenPersistentObject, TEE_ReadObjectData, TEE_WriteObjectData, TEE_CloseObject
Transient Object	Dependent Independent	TEE_GenerateKey, TEE_AllocateTransientObject, TEE_FreeTransientObject, TEE_InitRefAttribute, TEE_InitValueAttribute, TEE_SetOperationKey
Crypto Common	Independent	TEE_AllocateOperation, TEE_FreeOperation
Authenticated Encryption	Independent	TEE_AEInit, TEE_AEUpdateAAD, TEE_AEUpdate, TEE_AEEncryptFinal, TEE_AEDecryptFinal
Symmetric Cipher	Independent	TEE_CipherInit, TEE_CipherUpdate, TEE_CipherDoFinal
Asymmetric Cipher	Independent	TEE_AsymmetricSignDigest, TEE_AsymmetricVerifyDigest
Message Digest	Independent	TEE_DigestUpdate, TEE_DigestDoFinal

ソフトウェアコンポーネントとして TAM (Trusted Application Manager)，TEEP Broker，TEEP Agent で構成される．TAM はサーバ上にあり，クライアントのアプリケーションから必要な TA は TEEP Broker を通して要求される．TAM がその要求に対して，既に TA がインストールされているか等を TEEP Agent に確認する．問題がなければ TAM から TA がダウンロードされてインストールされる．注意しなければならないのは，TA の起動は各アーキテクチャに依存するために TEEP では TA の起動までを定義していない．TEEP の仕様では install/update/delete のみの規定である．

2.6　リモートアテステーション

先に述べたように基本的に TEE は隔離実行を提供するのみであり，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．リモートアテステーションではプラットフォーム内の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および TA の真正性判定をリモートの信頼できるサーバで行う仕組みである．

Intel SGX や AMD SEV では先に述べた Intel ME や AMD PSP などの信頼の基点を基にリモートアテステーションを提供している．現状でこららのリモートアテステーションはそれぞれのベンダーが提供するサーバを使うことを前提としている．これに対して，リモートアテステーションをオープンにする研究・活動として OPERA (OPEn Remote Attestation) [5] や Data Center Attestation Primitives(DCAP) がある．また，Intel ME や AMD PSP の詳細は公開されておらず，そのセキュリティがどのように保証されれいるかが不明である．

リモートアテステーションに関するプロトコルとして IETF で RATS (Remote ATtestation ProcedureS) [20] が議論されている．リモートアテステーションは既に TPM や FIDO で使われていた仕様があり，こららを統合するフォーマットとしての議論が進んでいる．また，先に述べた TEEP でも RATS をベースに TAM がリモートアテステーションをすることが想定されている．

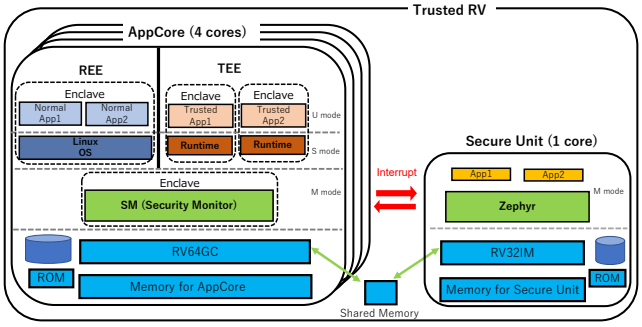

図 3　Trusted RV (AppCore と Secure Unit) のハード・ソフト構成

3.　Trusted RV の実装

64bit RISC-V のセキュリティを 32bit RISC-V で保護する Secure CoProcessor を開発した．64bit RISC-V と Secure CoProcessor が一体となったプラットフォームを Trusted RV と呼ぶ．Trusted-RV では 64bit RISC-V を AppCore と呼び，Secure CoProcessor を Secure Unit と呼ぶ．

図 3 に AppCore と Secure Unit のハード・ソフト構成を示す．AppCore では M/S/U モードを提供し，通常の Linux と Keystone が動作する．Secure Unit は Machine モードのみを提供し，OS としては Zephyr が動作する．AppCore と Secure Unit の間には共有メモリを置き，その間でのみデータ交換が行われる．データ交換はそれぞれ単方向の割り込みによって通知される．

現在，プロトタイプは XILINX FGPA VC707 で試作しており，Secure Unit は耐タンパであると想定する．Seucre Unit は小規模は ROM とセキュアストレージを有し，更新

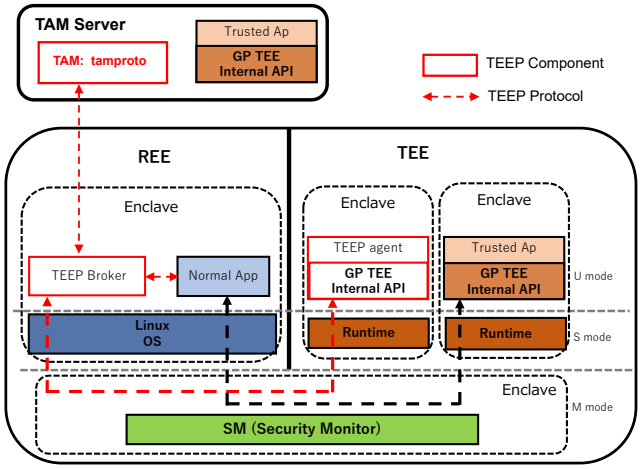

図 4　RISC-V Keystone 上での TEEP 実装

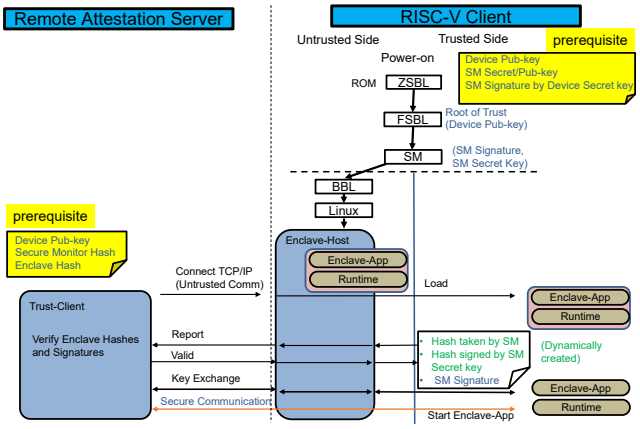

図 5　Keystone のリモートアテステーション

可能鍵をセキュアストレージに更新不可能鍵を ROM に保存する．ROM に保存される更新不可能鍵は Seucre Unit 外に出ることはない．セキュアストレージおよび ROM にアクセスできるは Zephyr 上のアプリケーションのみである．このアプリケーションは基本的に AppCore からの割り込みリクエストで起動され，その結果を共有メモリに保存した後に AppCore 側に対する割り込みを発生することで完了通知を行うことを基本とする．

Secure Unit では鍵管理の他に鍵生成，乱数生成などの機能が望まれる．このためには真正乱数を生成するハードウェアの乱数生成器，あるいは外部から信頼できる真正乱数を取得できる仕組みが必要となる．また，証明書の検証をするために正確な時間も必要となる．これらの機能はまだ実装されておらず，現在は TRV にその機能があるものとしてソフトウェア開発を行っている．

4. GP TEE Internal API の実装

GP TEE Internal API を Keystone に実装するにあたり，Keystone が提供する SDK を活用した．これは SDK が runtime である eyrie との API を提供するためである．実装はプログラミングに必要最低点と思われるものに絞った．具他的には 130 API のうち 28 API，29 のカテゴリーのうち，9 カテゴリーである．こららの一覧を表 1 に示す．

本実装では他の TEE でも活用できるようにするために，GP TEE Internal API をアーキテクチャ非依存/依存に分けた．アーキテクチャ非依存の API は CPU に依存しない純粋のライブラリとして実装した．また，アーキテクチャ依存の API は CPU 固有の実装になるが，できるだけポータビリティをもった実装とした．この分離のお陰で，Intel SGX にもポーティングすることができた．

また，Keystone SDK を活用するとそれに付随する EDL(Enclave Definition Language) を適用することになる．EDLL は TEE と REE 間の通信部分のコードを自動生成するもので，ポインタの悪用やバッファの使い方に間違いがないことを検証する．この自動生成されコードによりセキュリティは向上するが検証のためのオーバーヘッドがある．特に細粒度の通信を行う場合には注意が必要である．この詳細については [19] で筆者らが発表しているので参照されたい．

5. TEEP の実装

図 4 に TEEP を RISC-V Keystone に対応させた実装を示す．この図は図 2 と図 3 と統合した形になっている．図 4 に示すように TEE 内では我々が開発した GP TEE Internal API を使い，TEE Agent が実装されている．TAM についてはその実装を node.js で行い，tamproto [14] としてオープンソースで公開している．この開発は IETF TEEP の Hackathon を中心に行い，その状況は IETF 108 [15] で報告している．また，Keystone 上での TEEP 実装詳細については [24] で筆者らが発表しているので参照されたい．

6. Remote Attestation の実装

RISC-V Keystone では基本的に Keystone が提供した機能をそのまま使えるようにした．ここで Secure CoProcessor と関係する部分はデバイス固有鍵である．現状でまだ実装が完了していないが，デバイス固有鍵は Secure CoProcessor 内で保存し，必要に応じて AppCore 側である 64bit RISC-V に渡される．

図 5 に Keystone のリモートアテステーションの手順を示す．ここ手順で重要なものは AppCore および Remote Attestation サーバの事前準備の部分 (黄色で強調部分) である．クライアント側ではプラットフォーム固有の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 のハッシュ値が設定されいている．

図 5 では AppCore で Enclave-Host が実行されており，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 をロードを開始する前にリモートアテステーションを行う．このリモートアテステーションのためのセキュアモニタと TA コードの認証を行うためのレポートを作成する．セキュアモニタのレポートは，セキュアモニタのハッシュ値，セキュアモニタ公開鍵，およびそれに対して行ったデバイス固有秘密鍵による署名である．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を使って検証する．この検証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することで，リモートアテステーションが完了する．

完了後，図 5 下に示したように Enclave-App と Attestation-App がディフィーヘルマンで鍵交換してセキュアチャネルを作ることで，本来意図した TA の実行結果を受け取ることができる．

7. セキュリティ解析

現在，Truxted-RV は XILINX 社の FPGA である VC-707 上で試作されている．このため，Secure CoProcessor のサイドチャネル攻撃耐性は想定されていない．耐タンパ性はチップ作成とともの改めで考慮する予定である．

GP TEE Internal API を実装するために Keystone SDK を使って，EDL によるポインタおよびバッファ利用が防いでないかの検証が行われている．しかし，このようはツールが使っても不十分であることが CCS19 の論文 [25] で報告されている．今回の実装ではそのセキュリティ詳細までの確認は行えなかったため，今後の課題とする．

8. おわりに

本論文では RISC-V TEE である Keystone を強化するための Secure CoProcessor の開発を中心に，それに関係するソフトウェアである GlobalPlatform TEE Internal API，TEEP プロトコル，Remote Attestation の実装について述べた．まだ，開発途中ではあるが，それぞれは一応の機能は有しており，今後は実応用に耐えうる性能，およびセキュリティ評価を行っていく予定である．

本論文で述べた Trusted-RV およびそれに関連するソフトウェアは NEDO の委託事業を受けた技術研究組合 TRASIO で開発を進めている．

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「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．

参考文献

[1] Alhazmi, O. H. and Malaiya, Y. K.: Quantitative vulnerability assessment of systems software, *Reliability and Maintainability Symposium*, (RAMS), pp. 615–620 (2005).

[2] Alhazmi, O. H., Malaiya, Y. K. and Ray, I.: Measuring, Analyzing and Predicting Security Vulnerabilities in Software Systems, *computers & security*, Vol. 26, No. 3, pp. 219–228 (2007).

[3] Bourgeat, T., Lebedev, I., Wright, A., Zhang, S. and Devadas, S.: MI6: Secure enclaves in a speculative out-of-order processor, *International Symposium on Microarchitecture (MICRO)* (2019).

[4] Buhren, R., Werling, C. and Seifert, J.-P.: Insecure Until Proven Updated: Analyzing AMD SEV's Remote Attestation, *Computer and Communications Security*, (CCS) (2019).

[5] Chen, G., Zhang, Y. and Lai, T.-H.: OPERA: Open Remote Attestation for Intel's Secure Enclaves, *Conference on Computer and Communications Security*, (CCS) (2019).

[6] Costan, V., Lebedev, I. and Devadas, S.: Sanctum: Minimal hardware extensions for strong software isolation, *USENIX Security Symposium*, (USENIX Sec) (2016).

[7] Costan, V., Lebedev, I. and Devadas, S.: 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 (2017).

[8] Costan, V., Lebedev, I. and Devadas, S.: Secure Processors Part II: Intel SGX Security Analysis and MIT Sanctum Architecture, *Foundations and Trends in Electronic Design Automation*, Vol. 11, No. 3 (2017).

[9] Dambra, S., Bilge, L. and Balzarotti, D.: SoK: Cyber Insurance–Technical Challenges and a System Security Roadmap, *IEEE Symposium on Security and Privacy*, (IEEE SSP) (2020).

[10] GlobalPlatform: Root of Trust Definitions and Requirements, Version 1.1, *GlobalPlatform Specification* (2018).

[11] HexFive: MultiZone Secure IoT Stack, https://github.com/hex-five/multizone-secure-iot-stack/ (2018).

[12] Hunt, G., Letey, G. and Nightingale, E.: The seven properties of highly secure devices, *Tech. Report MSR-TR-2017-16* (2017).

[13] Intel: Software Guard Extensions SDK, https://software.intel.com/en-us/sgx/sdk/ (2019).

[14] Isobe, K.: tamproto, https://github.com/ko-isobe/tamproto (2020).

[15] Isobe, K. and Tsukamoto, A.: ETF 108 TEEP Hackathon Report, https://www.ietf.org/proceedings/108/slides/slides-108-teep-hackathon-report-01 (2020).

[16] Keystone developers: Keystone SDK, https://github.com/keystone-enclave/keystone-sdk (2019).

[17] Lee, D., Kohlbrenner, D., Shinde, S., Asanović, K. and Song, D.: Keystone: an open framework for architecting trusted execution environments, *European Conference on Computer Systems*, (EuroSys) (2020).

[18] Microsoft: Open Enclave SDK, https://openenclave.io/sdk/ (2019).

[19] Nakajima, K. and Suzaki, K.: Portable Implementation of GlobalPlatform API for TEE, *RISC-V GlobalForum* (2020).

[20] RATS Working Goup: RATS (Remote ATtestation Pro-

cedureS), https://datatracker.ietf.org/wg/rats/ (2018).
[21] RISC-V International: RISC-V Specifications, https://riscv.org/technical/specifications/ (2019).
[22] Shin, Y. and Williams, L.: An empirical model to predict security vulnerabilities using code complexity metrics, *International Symposium on Empirical Software Engineering and Measurement*, (ESEM) (2008).
[23] TEEP Working Goup: TEEP (Trusted Execution Environment Provisioning), https://datatracker.ietf.org/wg/teep/ (2017).
[24] Tsukamoto, A. and Suzaki, K.: TEEP (Trusted Execution Environment Provisioning) on RISC-V, *RISC-V GlobalForum* (2020).
[25] Van Bulck, J., Oswald, D., Marin, E., Aldoseri, A., Garcia, F. D. and Piessens, F.: A Tale of Two Worlds: Assessing the Vulnerability of Enclave Shielding Runtimes, *Computer and Communications Security*, (CCS) (2019).
[26] Weiser, S., Werner, M., Brasser, F., Malenko, M., Mangard, S. and Sadeghi, A.-R.: TIMBER-V: Tag-Isolated Memory Bringing Fine-grained Enclaves to RISC-V., *Network and Distributed System Security Symposium*, (NDSS) (2019).
[27] 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, pp. 576 – 579 (2020).