IoT のための PKI によるシステム構築方法の提案

飯田 正樹[†1]　亀谷 聡[†1]　永見 健一[†1]　遠藤 貴裕[†1]　古瀬 正浩[†1]

概要：モノのインターネット（IoT，Internet of Things）で利用される多種大量のデバイスを，様々なセキュリティ侵害から防御することは困難である．IoT に関わるデバイスおよびサービスの開発者には，デバイスへのセキュリティ侵害を前提としたシステム構築が求められる．本稿では，TEE（Trusted Execution Environment）を利用した実行環境分離技術により，IoT に最適な形で PKI（Public Key Infrastructure）を適用し，IoT のデバイスとサービスとを安全に接続するシステム構築方法を提案する．提案方式により，大量のデバイスに対しても管理コストを抑えつつ，デバイスへのセキュリティ侵害を前提としたシステム構築が可能となる．

キーワード：モノのインターネット（IoT），公開鍵基盤（PKI），TEE（Trusted Execution Environment）

The Proposal of a PKI-based System Design for IoT

Masaki Iida[†1]　Satoshi Kamegai[†1]　Kenichi Nagami[†1]
Takahiro Endo[†1]　Masahiro Furuse[†1]

Keywords: IoT (Internet of Things), PKI (Public Key Infrastructure), TEE (Trusted Execution Environment)

1. はじめに

1.1 IoT のセキュリティ脅威と対策

近年，IoT システムのセキュリティ対策が課題となっている．2015 年には自動車に対するリモート攻撃についての研究成果が発表され，攻撃者がネットワーク越しに自由に自動車を操作できたことから問題となった[1]．家電製品についても，デバイスとサービスとの間で相互認証や暗号化が不十分であったり，多くのデバイスが脆弱なパスワードのまま利用されていたり，実装と管理の両面で様々な不備が指摘されている[2]．2020 年には，インターネットに接続されるデバイスが 500 億台に達すると予測されている[3]．攻撃手法が高度化かつ巧妙化していくなか，これらの大量かつ管理不十分な IoT デバイスについて，セキュリティ侵害を未然に防ぐことは困難である．IoT 時代では，大量のデバイスに対しても管理コストを抑えつつ，セキュリティ侵害を前提としたセキュリティ対策が必要不可欠である．

我々は，IoT システムのセキュリティ対策として，PKI に基づく相互認証や暗号化が有効であると考えている．デバイスに固有となる秘密鍵を生成し，それに紐付く公開鍵証明書（以下，デバイス証明書）を発行できれば，パスワードに頼らない，公開鍵暗号方式によるデバイス認証が可能となる．また，デジタル署名技術を用いることで，デバイス認証に加え，デバイスとサービスとの間で交換される情報の完全性も保証できる．一方で，PKI で利用される RSA 暗号等の公開鍵暗号方式は，非力な IoT デバイスにとっては計算コストが高く，必ずしも適していない．しかしながら，最近では，RSA 暗号と同等の安全性を保ちつつ，鍵長を短く抑えることが可能な楕円曲線暗号（ECC，Elliptic Curve Cryptography）を利用した方式も提案され，IoT デバイスでも無理なく扱える状況が整ってきている．

1.2 IoT に PKI を適用するための課題

IoT で想定されるデバイスに対して PKI を適用するには，以下の課題を解決する必要があると考える．

- **デバイス証明書の信頼性確保**
  IoT デバイスに対してデバイス証明書を発行し，相互認証や暗号化を実現したとしても，使用している秘密鍵の安全性を保証できなければ意味がない．前述したとおり，IoT デバイスに対するセキュリティ侵害を未然に防ぐことは困難であるが，少なくとも秘密鍵の安全性は保証する必要がある．
- **証明書発行の低コスト化とスケーラビリティ確保**
  大量のデバイスに対してデバイス証明書を発行するためには，デバイス証明書に記載される識別情報の審査，もしくは審査に必要な正しい識別情報の準備を自動化し，人手による作業をできる限り減らす必要がある．また，デバイス数の増減に応じて，容易にスケールアウト／スケールインが行える必要がある．

1.3 本研究の貢献

本研究では，TEE および ARM® TrustZone®テクノロジに着目し，これらを信頼の起点とした，IoT に最適な PKI を

†1 株式会社インテック
INTEC Inc.

提案する．この提案方式により発行されるデバイス証明書を応用することで，IoT のデバイスとサービスとを安全に接続する IoT システムの構築が可能となる．また，提案方式によるデバイス証明書の信頼性と発行コスト，スケーラビリティについての評価を行い，その有用性を示す．

2. 関連研究

デバイス証明書の発行を自動化する方法として，プライベート環境等では SCEP（Simple Certificate Enrollment Protocol）が広く利用されている．SCEP では，デバイスと認証局との間で秘密のチャレンジパスワードを事前に共有しておき，認証局側で，デバイスから識別情報と一緒に提出されたチャレンジパスワードが正しいことを確認できれば，デバイス証明書を自動で発行する．しかしながら，悪意を持ったデバイスが，正しいチャレンジパスワードと一緒に詐称した識別情報を提出することが可能であるため[4]，認証局により信頼できる情報源（企業のディレクトリサービスなど）のデバイス識別情報を使った審査を追加で実施する方法が提案されている[5]．この場合，予め審査に必要な正しいデバイス識別情報を準備する必要がある．

また，PC のマザーボード等に実装される TPM（Trusted Platform Module）に対してデバイス証明書を発行する方法が TCG（Trusted Computing Group）により提案されている[6]．TPM は，ハードウェアレベルの耐タンパー性を持った特殊なセキュリティチップであり，秘密鍵の厳格な管理が行われるため安全性が非常に高い．しかしながら，専用のハードウェアを追加で必要とするため実装コストが高く，現状では一部の PC やタブレット等での利用に限られる．IoT デバイスの多くで利用可能な状況であるとは言い難い．（ただし最近では，ファームウェアベースの TPM も提案されている[7].）

本研究では，デバイスに対して専用のハードウェアを追加で必要とせず，IoT システムに適したデバイス証明書を低コストで発行する方法を提案する．

3. 要素技術

TEE および TrustZone テクノロジは，ハードウェアによる実行環境分離技術であり，スマートフォンでの指紋認証や決済システム，また DRM（Digital Rights Management）等で活用されている．本章では，提案方式を実現するために必要な，これらの要素技術について説明する．

3.1 TEE (Trusted Execution Environment)

TEE とは，通常の実行環境（REE，Rich Execution Environment）から分離された，アプリケーションのための安全な実行環境であり，その標準仕様である．実行環境の分離はハードウェアによるサポートによって実現され，TEE で動作するアプリケーション（TA, Trusted Application）およびデータについて，その機密性と完全性を保証する．

TEE 仕様は，IC カード仕様の標準化等を推進する GlobalPlatform により策定が進められている．図 1 に TEE のアーキテクチャを示す．また，以下に TEE を構成する主要な仕様について述べる．

- **TEE Client API[8]**
  REE で動作するアプリケーションから，TEE で動作するアプリケーション（TA）に対する接続方法を定める．
- **TEE Internal API[9]**
  TEE で動作するアプリケーションに対して，暗号処理やセキュアストレージ等の各種機能の使用方法を定める．

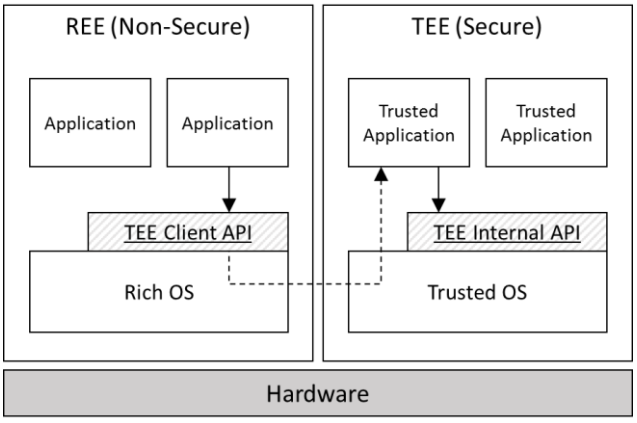

図 1　TEE のアーキテクチャ

TEE は，ソフトウェア攻撃（Hack Attack）と，一部の低コストのハードウェア攻撃（Shack Attack）を防御することを目的としている．電子顕微鏡を使用してチップの内部を解析するような，高コストのハードウェア攻撃（Lab Attack）を防御することはできない．ハードウェア攻撃を防御するためには，SE（Secure Element）を利用する方法がある．SE とは，クレジットカードで使用される IC チップのように，ハードウェアレベルの耐タンパー性を備えた特殊なハードウェアの総称である．表 1 に TEE と周辺技術との比較を示す．TEE は，SE に比べて安全性では劣るが，実装コストで優位である．また，TEE はメインプロセッサで実行されるため，REE での処理と比較しても性能劣化がなく，ソフトウェアにより様々な機能が追加できる自由度の高さを備える．

表 1　TEE と周辺技術との比較（文献[10]を参考に作成）

技術種別	安全性	自由度	コスト	特長
ソフトウェアベース	低	高	低	
ハードウェアベース（TEE）	中	中～高	中	● ソフトウェアレベルの脅威から保護
ハードウェアベース（SE）	高	低	高	● ソフトウェアレベルの脅威から保護 ● ハードウェアレベルの脅威から保護

3.2 ARM TrustZone テクノロジ

TEE を実現する実装技術として，ARM アーキテクチャの一部のプロセッサに搭載される TrustZone テクノロジが知られている．TrustZone テクノロジでは，プロセッサにセキュアモニタモードを新たに追加することで，実行環境をノーマルワールドとセキュアワールドとに分離することを実現している．ノーマルワールドでは，我々が普段利用している Linux 等の汎用的な OS（Rich OS）を稼動させる．一方のセキュアワールドでは，生体認証など，特にセキュリティが要求される処理を行う信頼された OS（Trusted OS）を稼動させる．

ワールド間のコンテキストを切り替えるには，プロセッサに対してセキュアモニタコール（SMC，Secure Monitor Call）命令を与える．プロセッサが SMC 命令によりセキュアモニタモードに移行すると，セキュア構成レジスタ（SCR, Secure Configuration Register）の NS ビットを変更することで，セキュアワールド（SCR.NS=0）とノーマルワールド（SCR.NS=1）とを切り替える．メインメモリや周辺システムは，システムバスを通じてそれを認識することで，システム全体に渡る実行環境の分離が実現される．

通常，セキュアワールドの安全性を保証するために，デバイスの電源が投入されると，セキュアブートによる起動プロセスが実行される．最初に，書き換え不能な ROM ベースのブートローダにより，セキュアワールドのブートローダの完全性が検証されて起動する．セキュアワールドのブートローダは，Trusted OS の完全性を検証してから起動させる．このように，ソフトウェアの完全性を検証しながら順次起動していくことで，不正なファームウェアや OS の起動を排除している．Trusted OS は，次にノーマルワールドのブートローダを起動し，最後にノーマルワールドの Rich OS が起動されることで，システムの起動が完了する．

3.3 TAM (Trusted Application Management)

本稿では，TEE で動作するアプリケーション（TA）を管理し，TEE に対して安全に配備する仕組みを TAM と呼ぶ．TAM は，デバイスの出荷時に埋め込まれ，TEE からのみアクセスすることが可能なデバイスの固有鍵，もしくはそこから導出される派生鍵を利用することで，TA を暗号化して TEE に安全に配備する．

現在，TAM については，複数の事業者により独自のサービスとして提供されている[11][12]．また，GlobalPlatform において標準化が進められている[13]．

4. 提案方式

4.1 アイデア

我々は，IoT に PKI を適用するための課題について，以下に述べる方法で解決できると考えている．

- **デバイス証明書の信頼性確保**

  TEE で実行されるアプリケーションは，REE を汚染するマルウェアや，悪意を持つ利用者による管理者権限悪用の影響を受けることがない．したがって，デバイス証明書の発行対象となる公開鍵暗号の鍵ペアおよび証明書署名要求を TEE で生成し，生成された秘密鍵が TEE の外部に漏えいしないよう制御することで，デバイスへのセキュリティ侵害の有無によらず，デバイス証明書の信頼性を確保することができる．

- **証明書発行の低コスト化とスケーラビリティ確保**

  TEE は前述の通り，REE からの不正な干渉を受けることがない．したがって，TEE で動作するように設定された手続き（TA，Trusted Application）は，常に意図した通りに動作することを期待できる．認証局により設定された証明書申請手続きが TEE で実行されるのであれば，認証局によって信頼できる方法でデバイス識別情報が取得され，証明書署名要求が生成されることを期待できる．つまり認証局では，デバイスから受け取った証明書署名要求が，TEE で生成されたものであり，その内容が途中経路で改ざんされていないことを確認できれば，記載されたデバイス識別情報の正しさを審査する必要がなくなる．

  これまでは，認証局がデバイスの属性を保証するために，デバイスから提出される識別情報の正しさを何らかの方法により審査する必要があった．その審査のために，例えば企業であれば，デバイスの管理者が予め正しいデバイス識別情報のリストを作成してディレクトリサービスに登録しておく等の負担を強いられた．認証局によるデバイス識別情報の審査が不要となれば，こうした負担を解消することが可能となる．

4.2 デバイス証明書の発行方法

図 2 に，提案方式によるデバイス証明書の発行に必要な構成要素を示す．以下に，その構成要素それぞれの詳細を説明する．

- **TA**
  デバイス証明書の申請手続きが設定された，デバイスの TEE で実行されるアプリケーション．
- **認証局**
  デバイス証明書を発行する．また，デバイス証明書の正しい申請手続きを定め，TA を構成する．
- **TAM**
  TA をデバイスの TEE に安全に配備するため，TEE からアクセスできるデバイスの固有鍵，もしくはその派生鍵により TA を暗号化する．
- **デバイス（REE）**
  デバイスのノーマルワールド（通常の実行環境）．ネットワークに接続されるため，リモート攻撃やマルウェア感染などが想定される．また，デバイスの利用者による管理者権限の悪用も想定される．
- **デバイス（TEE）**
  デバイスのセキュアワールド（信頼された実行環境）．ネットワークには直接接続されず，デバイスの REE からはハードウェアレベルで分離される．

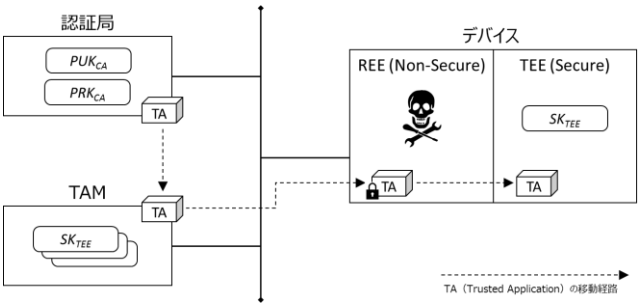

図 2　提案方式の構成

図 3 にデバイス証明書発行の流れを示す．提案方式は，“TA 配付フェーズ”と“デバイス証明書発行フェーズ”の 2 つのフェーズから構成される．以下に，図 3 に記載された手順番号にしたがって，各フェーズの詳細を説明する．

TA 配付フェーズ

[1] 認証局は，証明書署名要求を認証するための CSR 認証鍵 SK_{CSR} を乱数により生成する．

[2] 認証局はデバイス証明書の正しい申請手続きを定め，手順[1]で生成した CSR 認証鍵 SK_{CSR}, そして自身の公開鍵 PUK_{CA} と一緒に，TA として TAM に登録する．

[3] TAM は，デバイスからのリクエストに応じて，そのデバイスのデバイス固有鍵 SK_{TEE} により暗号化した TA’を配付する．

[4] デバイスの TEE は，REE から TA’を取得する．取得した TA’について，デバイス固有鍵 SK_{TEE} により復号および完全性の検証を行って TA を得る．

デバイス証明書発行フェーズ

[5] TEE は，デバイス証明書が必要となった時点で，TA を実行してデバイス証明書の申請手続きを開始する．

[6] 実行された TA は，認証局により定められた手続きにしたがい，デバイス証明書の発行対象となる公開鍵暗号の鍵ペア（公開鍵 PUK_{TEE} と秘密鍵 PRK_{TEE}）を生成する．

[7] TA は認証局により定められた手続きにしたがい，デバイス識別情報を取得する．デバイス識別情報は，ハードウェアレベルで変更ができない情報（OTP-ROM に記録された情報など）を利用する．例えば，シリアル番号, IMEI, MAC アドレスなどの情報を取得する．

[8] TA は，手順[6]および手順[7]で得た鍵ペアとデバイス識別情報を使い，証明書署名要求 *CSR* を生成する．

[9] TA は証明書署名要求 *CSR* について，TA に含まれる CSR 認証鍵 SK_{CSR} により, メッセージ認証符号 MAC_{CSR} を計算する．

[10] TA は，手順[8]および手順[9]により得られた証明書署名要求 *CSR* とメッセージ認証符号 MAC_{CSR} をまとめ，REE に渡す．

[11] デバイスの REE は，証明書署名要求 *CSR* とメッセージ認証符号 MAC_{CSR} を認証局に送付する．

[12] 認証局は, 受け取った証明書署名要求 *CSR* について，同じく受け取ったメッセージ認証符号 MAC_{CSR} と，手順[1]で生成した CSR 認証鍵 SK_{CSR} により，認証および完全性の検証を行う．

[13] 認証局は，認証および完全性を検証した証明書署名要求 *CSR* を信頼し，デバイス証明書 *CRT* を発行する．

[14] デバイスの TEE で実行されている TA は，REE からデバイス証明書 *CRT* を取得する．取得した *CRT* について，TA に含まれる認証局の公開鍵 PUK_{CA} で署名の検証を行い，正しい証明書と確認できればインストールする．

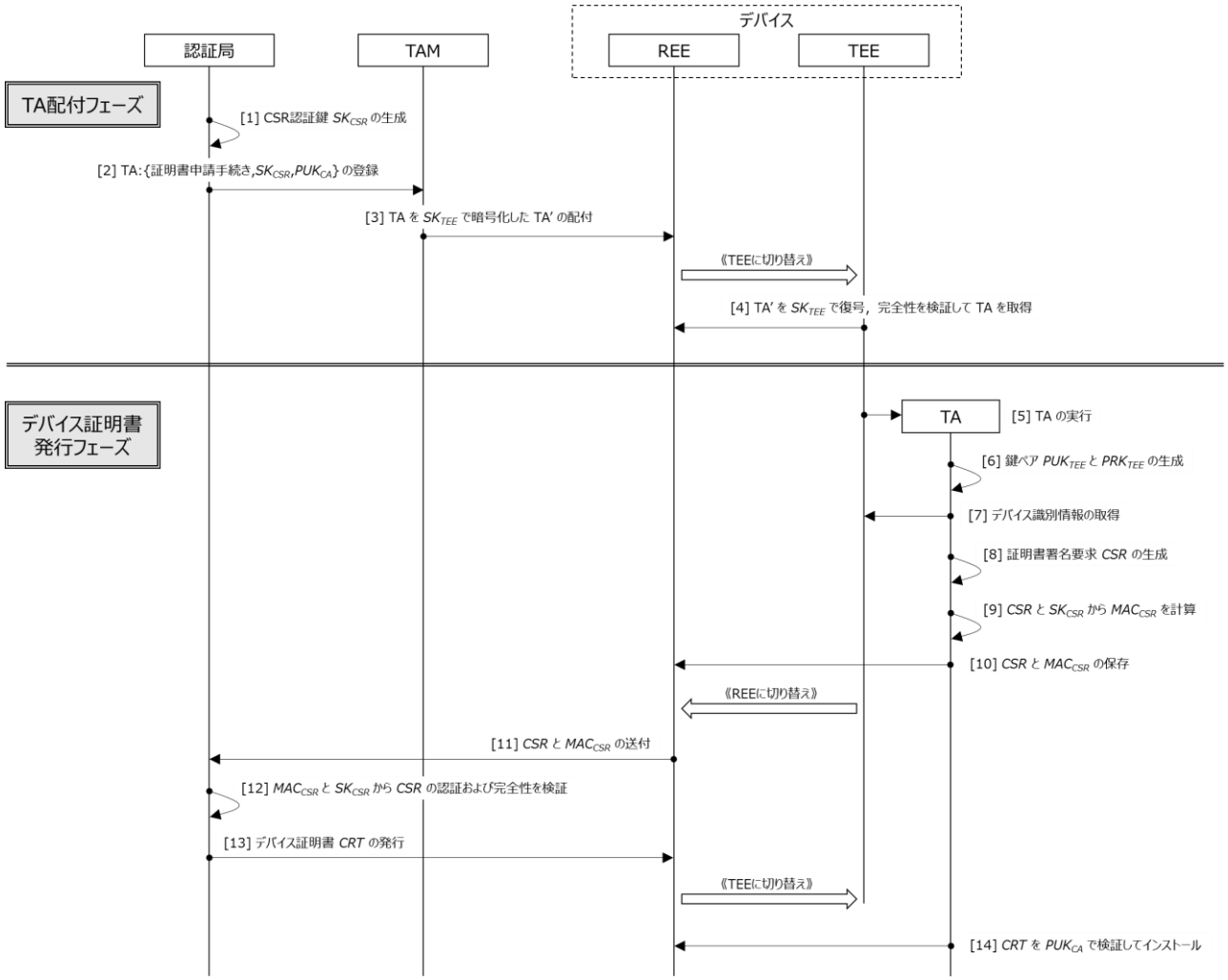

図 3　デバイス証明書発行の流れ

4.3 秘密鍵のアクセス制御

発行されたデバイス証明書の安全性を確保するために，TEE で生成した公開鍵暗号の鍵ペア（公開鍵 PUK_{TEE} と秘密鍵 PRK_{TEE}）について，秘密鍵 PRK_{TEE} は TEE の外部に漏えいしないように制御する必要がある．そのために，デバイス証明書 *CRT* が適切にインストールされた後，TA は秘密鍵を利用するための各種 API を提供する“証明書／鍵管理モジュール”として機能する．

図 4 に，TA による証明書／鍵管理モジュールの構成を示す. TA は, 生成した公開鍵暗号の鍵ペア（PUK_{TEE}, PRK_{TEE}）と認証局より発行されたデバイス証明書 *CRT* を，TEE Internal API により提供されるセキュアストレージで管理する．セキュアストレージは TA 単位で空間が区切られ，他の TA からはアクセスすることができない．そのため，クライアントアプリケーションが署名や復号等の秘密鍵 PRK_{TEE} を使用する処理を行う場合，必ず TA に用意された証明書／鍵利用 API を経由しなければならない（図 4 の(a)と(b)）.

デバイス証明書は使用する用途の違いにより，一つのデバイスに対して複数発行され得る. 証明書／鍵利用 API は，デバイス証明書の発行を要求したクライアントアプリケーションに対してのみ，そのデバイス証明書に紐付けられた秘密鍵に関連する各種 API を提供する．

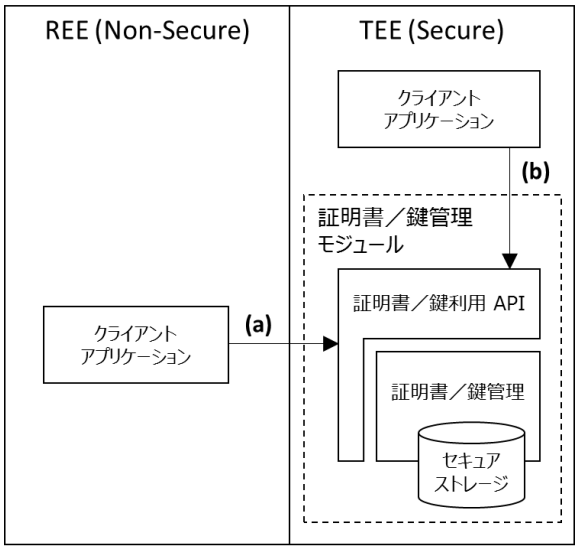

図 4　証明書／鍵管理モジュールの構成

5. 提案方式を用いた応用例

本章では，提案方式を用いた一つの応用例として，位置情報サービスの信頼性向上を目的とした，位置情報検証システムを検討する．

位置情報サービスとは，スマートフォンの位置情報を利用したゲームアプリケーションや，歩行者や自動車の位置情報を利用したナビゲーションシステムなどを指す．これらの位置情報サービスは，今や生活に欠かせないものになっている．しかしその一方で，利用者のデバイスで測位された位置情報に基づく仕組みのため，悪意を持つ一部の利用者により，不正な利益を得る目的で位置情報を詐称されることが問題となっている．そのため，信頼できる位置情報サービスを提供するためには，位置情報の詐称を防ぐ仕組みが必要となる．

位置情報の詐称を防ぐには，位置情報が改ざんされていないことを位置情報サービスが検証できればよい．この検証には，提案方式により発行されたデバイス証明書，そしてデジタル署名技術が利用できる．

図 5 に位置情報検証システムの構成例を示す．この例では，デバイスの GNSS（Global Navigation Satellite System）受信機により測位された位置情報を，デバイスの REE ではなく TEE で直接取得している．TEE では，取得した位置情報に対して，証明書／鍵管理モジュールが提供する証明書／鍵利用 API により署名を行ってから REE に渡す．

位置情報サービスでは，事前に登録されたデバイスのシリアル番号等により，署名の検証に必要なデバイスの公開鍵を認証局から得ることができる．仮に位置情報が詐称されていたとしても，位置情報に付与された署名を検証することで，位置情報の改ざんの有無を判断することが可能となる．

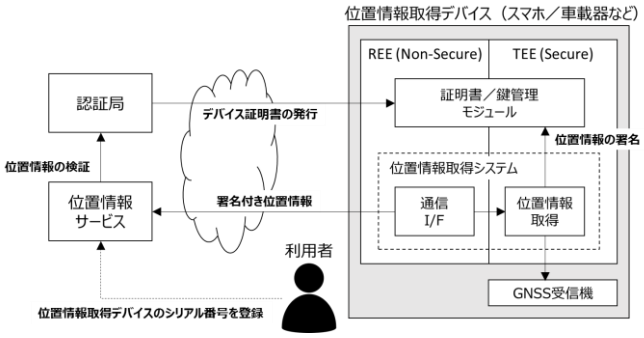

図 5　位置情報検証システムの構成

6. 提案方式の評価

本章では，デバイス証明書の信頼性と発行コスト，スケーラビリティについての評価を行い，その有用性を示す．

以下，第 4.2 節の手順番号を用いて説明する．

6.1 デバイス証明書の信頼性確保

デバイスの REE に対して，リモート攻撃やマルウェアの感染，もしくは悪意ある利用者による管理者権限の悪用が行われた場合においても，提案方式により発行されたデバイス証明書の信頼性が確保されることを説明する．

ここでは，デバイスの TEE が REE からの干渉を受けないこと，認証局および TAM は安全かつ信頼できることを前提とする．

- **TA の安全性**

  提案方式の手順[4]において，TEE は取得した TA’について，デバイス固有鍵 SK_{TEE} による復号および完全性の検証を行った上で TA を得ている．REE はデバイス固有鍵 SK_{TEE} にアクセスすることができないため，手順[3]で受け取った TA’について，内容の参照や改ざんはできない．

- **デバイス秘密鍵 PRK_{TEE} の安全性**

  提案方式の手順[6]において，デバイスごとに異なる鍵ペアを生成しているため，複数のデバイスで同一の秘密鍵 PRK_{TEE} が使用されることはない．また，秘密鍵 PRK_{TEE} へのアクセスは証明書／鍵利用 API によって制限されるため，秘密鍵 PRK_{TEE} が TEE の外部に漏えいすることはない．

- **デバイス識別情報の安全性**

  提案方式の手順[7]において，デバイス識別情報はハードウェアレベルで変更ができない情報を利用している．そのため，これらのデバイス識別情報が REE のカーネルやデバイスドライバによりソフトウェアレベルで変更されていたとしても，TEE で取得するデバイス識別情報が改ざんされることはない．

- **証明書署名要求 *CSR* の安全性**

  提案方式の手順[12]において，認証局は受け取った証明書署名要求 *CSR* について，メッセージ認証符号 MAC_{CSR} と CSR 認証鍵 SK_{CSR} により認証および完全性の検証が行われている．REE は CSR 認証鍵 SK_{CSR} にアクセスすることができないため，手順[11]で送付する *CSR* の改ざんはできない．

- **デバイス証明書 *CRT* の安全性**

  提案方式の手順[14]において，TA は認証局から受け取ったデバイス証明書 *CRT* について，認証局の公開鍵 PUK_{CA} により認証および完全性の検証が行われている．REE は認証局の秘密鍵 PRK_{CA} にアクセスすることができないため，手順[13]で受け取った *CRT* について，偽の CRT への差し替えや内容の改ざんはできない．

以上の理由により，TEE の安全性のレベルにおいて，デバイス証明書の信頼性は確保されていると考えられる．

6.2 証明書発行の低コスト化とスケーラビリティ確保

提案方式の手順[12]において，認証局はメッセージ認証符号 MAC_{CSR} を用いた証明書署名要求 *CSR* の検証により，*CSR* が CSR 認証鍵 SK_{CSR} を持つ TA により生成されたものであると確認することができる．また，TA はデバイス固有鍵 SK_{TEE} により，デバイスの TEE でのみ実行されることが保証されている．つまり，認証局が受け取った *CSR* は，デバイスの TEE において，認証局が意図した手続きにより生成されたものであると確認できる．したがって，*CSR* に記載されたデバイス識別情報の審査が不要となるから，認証局によるデバイス証明書発行に関する一連の処理は自動化することができる．

また，第 4.2 節で示した一連の手順は，複数のデバイスを対象として並列に実行することが可能である．したがって，デバイス数の増減に応じて，認証局および TAM のサーバ台数を増減することで，スケールアウト／スケールインが可能であると言える．

以上の理由より，デバイス証明書の発行コストを低下させ，またスケーラビリティが確保されていると考えられる．

7. おわりに

本稿では，TEE による実行環境分離技術を利用することで，IoT に最適な PKI を構築し，そのなかで安全かつ低コストでデバイス証明書を発行する方法を提案した．提案方式により，IoT デバイスに対するセキュリティ侵害が発生したとしても，発行されるデバイス証明書の信頼性が確保できることを示した．また同時に，デバイス証明書の発行コストを低下させられることを示した．

現在，本提案方式の実装を進めており，今後はパフォーマンス評価等も進める．TEE は，REE でのアプリケーションの実行と同様に，メインプロセッサおよびメインメモリを使用した高速な処理が可能であり，パフォーマンスの面でもその有用性が期待できる．

また，本稿ではデバイス証明書の発行についてのみ論じたが，実際の運用では証明書の更新や失効についても考える必要がある．今後の研究では，こうした課題についても検討を進めていく予定である．

参考文献

[1] Dr. Charlie Miller, and Chris Valasek. “Remote Exploitation of an Unaltered Passenger Vehicle”, Black Hat USA 2015, 2015. http://illmatics.com/Remote%20Car%20Hacking.pdf, (参照 2016-08-08)

[2] Mario Ballano Barcena, and Candid Wueest. “Insecurity in the Internet of Things”, Symantec Security Response, 2015. https://www.symantec.com/content/dam/symantec/docs/white-papers/insecurity-in-the-internet-of-things-en.pdf, (参照 2016-08-08)

[3] Dave Evans. “The Internet of Things: How the Next Evolution of the Internet Is Changing Everything”, Cisco Internet Business Solutions Group, 2011. https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf, (参照 2016-08-08)

[4] Ted Shorter, and Wayne Harris. “The Use of the Simple Certificate Enrollment Protocol (SCEP) and Untrusted Devices”, Certified Security Solutions, Inc., 2012. https://www.css-security.com/media/1473/scep-and-untrusted-devices_css.pdf (参照 2016-08-08)

[5] Gary A. Galehouse, Wayne A. Harris, Edward R. Shorter, and Kevin M. Tambascio. “System and method for validating SCEP certificate enrollment requests”, United States Patent 8745378 B1, 2014.

[6] “TPM Main Specification Level 2 Version 1.2, Revision 116”, Trusted Computing Group. http://www.trustedcomputinggroup.org/tpm-main-specification/, (参照 2016-08-08)

[7] H. Raj, S. Saroiu, A. Wolman, R. Aigner, J. Cox, P. England, C. Fenner, K. Kinshumann, J. Loeser, D. Mattoon, M. Nystrom, D. Robinson, R. Spiger, S. Thom, and D. Wooten. “fTPM: A Firmware-based TPM 2.0 Implementation”, Microsoft Research, 2015. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/msr-tr-2015-84.pdf, (参照 2016-08-08)

[8] “TEE Client API Specification Version 1.0”, GlobalPlatform Device Technology, 2010.

[9] “TEE Internal Core API Specification Version 1.1”, GlobalPlatform Device Technology, 2014.

[10] “GlobalPlatform made simple guide: Trusted Execution Environment (TEE) Guide”, GlobalPlatform. http://www.globalplatform.org/mediaguidetee.asp, (参照 2016-08-08)

[11] “MyTAM”, Intercede. https://www.intercede.com/product/mytam/, (参照 2016-08-08)

[12] “Trusted Application Management”, Gemalto. http://www.gemalto.com/mobile/mcommerce/mfs/trusted-service-hub/trusted-application-management, (参照 2016-08-08)

[13] “TEE Management Framework Version 0.0.0.38”, GlobalPlatform Device Technology, 2016.