Cortex-M向けTrustZoneとZETA環境利用時の暗号プロトコル実現性の考察

田村 桜子[1,a)]　伊藤 真奈美[1]　永井 彰[1]　安田 幹[1]

概要：IoT 機器で広く利用されている Cortex-M シリーズにおいても，TrustZone を搭載した ARMv8-M が注目され利用可能になっている．しかし，Cortex-M 向けの TrustZone は Cortex-A とはその構造も大きく異なり，SDK 等利用環境も整っていない．また，現状の IoT システムでは，LPWA(Low Power Wide Area) 通信が広く使われているが，LPWA 環境下で公開鍵ベースのプロトコルが動作するか不明である．本稿では，ID ベース認証鍵交換プロトコルを対象とし，将来 IoT システムの構成として考えられる LPWA の環境における TrustZone を使った場合，分単位のリアルタイム性で十分なシステムであれば実用的に利用できることを示した．

キーワード：TrustZone，Cortex-M，LPWA, ZETA, ID ベース認証鍵交換

A Study of Cryptographic Protocol Feasibility when Using TrustZone for Cortex-M and ZETA Environment

Sakurako Tamura[1,a)]　Manami Ito[1]　Akira Nagai[1]　Kan Yasuda[1]

Abstract: Also in the Cortex-M series widely used in IoT devices, ARMv8-M equipped with TrustZone has attracted attention and become available. Wrinkle, TrustZone for Cortex-M is Cortex-A and its structure is also large. In the current IoT system, Low Power Wide Area (LPWA) communication is widely used, but a public key based protocol under LPWA environment In this paper, we discuss the feasibility of the protocol using TrustZone and ZETA, a new LPWA, for the ID-based authentication key exchange protocol.

Keywords: TrustZone，Cortex-M，LPWA, ZETA, Id-based key exchange

1. はじめに

現在，IoT システムにおいて利用される通信規格としては，LPWA(Low Power Wide Area) が注目を集めており，LPWA のモジュール出荷台数，接続数が年々急速に増加し，2017 年に 0.6 億台であった出荷台数は，2020 年には 4 億台を超えることが予測されている [1]．LPWA とは，低消費電力で広域データ通信が可能な通信規格であり，通信速度は低速なものの，数 km から数十 km もの通信が可能な通信規格である．LPWA では，通信量の制限等のため，共通鍵ベースの認証や暗号化機能が実装されている [2], [6]．

IoT システムを用いたサービスの多くは，センサデバイスからセンサデータ等を収集し，見える化を行う可視化サービスが主流である．今後はさらに，IoT クラウド経由で IoT 機器を制御を行うシステムや，自律的に IoT 機器間で通信を行うようなユースケースも出てくると考えられる．例えば，農業分野では，気温，湿度等の環境データをもとに作物への水やりを実施したり，スマートホームでは，遠隔からエアコンをつけたりすることも可能になる．しかし，このような機能が実現すると，不正にアクセスされ，IoT 機器が意図しない動作をし，作物が育たないという金銭的な被害や，体調不良等の人的被害が発生する可能性もある．そのため，認証等のセキュリティ機能の必要性が高

[1] NTT セキュアプラットフォーム研究所
NTT Secure Platform Laboratories
[a)] sakurako.tamura.sz@hco.ntt.co.jp

まると考えられる．

IoT 機器のセキュリティ機能として，認証や暗号化プロトコルを実現する場合，IoT 機器側での秘密鍵の保護やプログラムの改ざん防止等も重要である．IoT 機器に認証や暗号化の機能を組み込んでいても，秘密鍵が漏洩してしまうと，その秘密鍵を悪用され，なりすましが可能になってしまう [3]．その解決策として，IoT 機器自体に外部からはアクセスできないセキュア領域を持つデバイスを利用する方法があり，広く利用されている ARM のプロセッサでは，TrustZone という実行環境をセキュアな領域と通常の領域に分けることが可能な技術が搭載されたデバイスが提供されている．以降本稿では，セキュアな実行環境を Secure World，通常の実行環境を Normal World とする．

これら LPWA と TrustZone により，今後の IoT システムでは，LPWA を利用した環境でのセキュア領域を持つデバイスでの認証方式の確立は重要である．しかし，LPWA 環境や，TrustZone の利用を想定した公開鍵ベースの暗号や認証などの暗号プロトコルの実現性は不明である．詳細は，2 章にて述べる．本研究では，LPWA としては，他の LPWA と比較し，低コストであることや中継器を設置し安定した通信路を形成できることが特徴である ZETA，公開鍵ベースの暗号プロトコルとしては，LPWA 環境で有効だと考えられる通信量の少ない ID ベース認証付き鍵交換を採用した．本稿では，IoT システムにおける TrustZone や LPWA 利用時の暗号プロトコルの実現性を検討するため，Cortex-M 向けの TrustZone を用い，LPWA の一種である ZETA を用いた暗号プロトコルの実現性について検討を行い，秒単位のリアルタイム性の必要ないユースケースにおいては実用的であることを示した．2 章には，検討に必要な技術の説明，3，4 章には，TrustZone，ZETA を使った場合の実現性の検討，5 章には ZETA 環境における TrustZone を利用したシステムを提案し，6 章にて今後の課題，7 章にて本稿をまとめる．

2. 準備

本章では，本稿で用いる技術の説明をする．

2.1 Cortex-M 向け TrustZone

TrustZone とは，ARM のプロセッサが提供するセキュリティ技術であり，実行環境を 2 つの world に分割し，通常の実行環境である Normal World と，セキュアな実行環境である Secure World の 2 つに分割することができる機能である．Normal World，Secure World をそれぞれの実行環境に，ROM 領域，RAM 領域が存在する．機密性の高い情報を扱う Secure World からのみ必要なハードウェアや情報へアクセスが可能になる．

TrustZone の構成は，表 1，表 2 に示すように，Android 等に利用されている Cortex-A 向けの TrustZone(ARMv8-A) とマイクロコントローラに利用されている Cortex-M 向けの TrustZone(ARMv8-M) は，実装が異なっている [10]．Cortex-A は，高スペックなデバイスを想定しているため，Cortex-A 向け TrustZone では，Secure World にはセキュア OS が実装され，Normal World と Secure World のやり取りは，Secure Monitor が管理し，Secure Monitor を通してやり取りを行う．またこの仕様は，Global Platform により標準化されている TEE[4] によって定義されており，関数名等も統一されている．一方，Cortex-M 向け TrustZone は，低電力マイクロコントローラ向けに最適化されているため，セキュア OS を持たず，セキュアモニタも存在しない．セキュアモニターの変わりに SG(Secure Gateway) 命令という拡張命令が実装されており，ダイレクトに Normal World から Secure World にある関数をたたくことが可能であり，Normal World から Secure World の処理を呼び出す場合には，必ず SG 命令が実行される．Cortex-M 向けの TrustZone が搭載されたプロセッサとしては，Cortex-M23，Cortex-M33 であるが，まだ評価環境は，FPGA を利用した評価方法がある．TrustZone ARMv8-M を用いた暗号プロトコルの実装方法は複数あるため，それらを整理した結果と具体的な実装方法は 3 章で述べる．

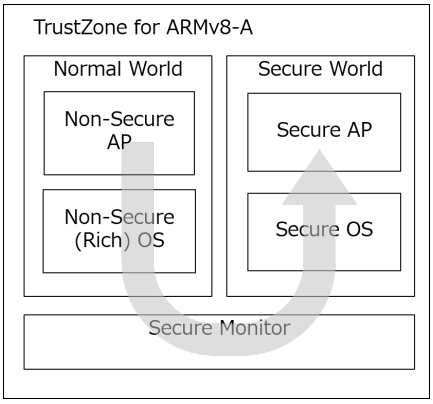

図 1: Cortex-A 向け TrustZone の構成

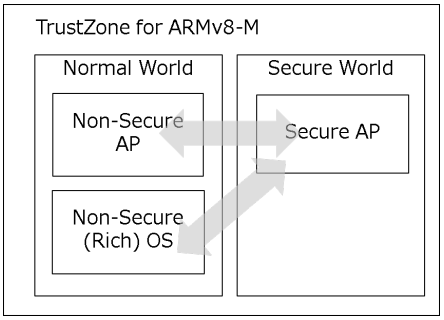

図 2: Cortex-M 向け TrustZone の構成

表 1: LPWA の通信規格

	ZETA	LoRa	Sigfox
周波数帯	920MHz，429MHz	920MHz	920MHz
エリア範囲	～10km	～10km	人口カバー率 94%
通信速度	300bps～2.4kbps	100bps～50kbps	100bps
データサイズ	50bytes	59bytes	12bytes

2.2 ZETA

LPWA(Low Power Wide Area) とは，低消費電力で広域データ通信が可能な通信規格であり，IoT システムで広く使われている．主な LPWA の規格を表 1 に示す．各通信規格ごとに，通信速度，データサイズ等が仕様として決められており，周波数帯に関しては日本での周波数帯を示しておりどの通信規格においても同様の周波数帯を用いている．送信データサイズが限られているため，セキュリティ機能においては，どの通信規格においても共通鍵ベースの署名，暗号化機能となっている．また，認証方式に関しては，デバイスの ID を使った独自の認証方式が行われている．

今回の検討対象である ZETA とは，LPWA の一種であり，2018 年に ZiFiSense 社が開発した LPWA 規格である．ZETA Alliance は，日本と中国で発足しており，アジア発の LPWA である．新興 LPWA であることもあり，他の LPWA と比較し，低コストであることや中継器を設置し安定した通信路を形成できることが期待される，かつ下りの通信への回数や通信帯域等の制限が少なく双方向通信に最も適していると判断し，本研究では ZETA を LPWA の通信規格として採用した．ZETA においても，他の LPWA と同様に共通鍵ベースの暗号化機能が備わっている．ZETA のシステム構成は図 3 の通りである．ZETA 通信 MS とは ZETA が通信可能な通信モジュールであり，UART 通信インターフェイスを使用してセンサ等が接続可能なモジュールである．ZETA AP は ZETA の基地局であり，ZETA AP の設置はユーザが行う．ZETA サーバは，システム制御，接続監視，デバイス管理を行うサーバであり，国内ではアイティアクセス株式会社がクラウドサービスとして提供している．ZETA 通信 MS − ZETA AP 間は，ZETA，ZETA AP − ZETA サーバ間は，LTE または有線 LAN，ZETA サーバ-アプリケーションサーバ間はインターネットにつながるという構成である [9]．Sigfox や LoRa のシステム構成も ZETA の構成とほとんど同じであり，基地局をユーザが設置するか，サービス提供者で設置するか等が異なる [6]．上りの通信では，データは，通信 MS から ZETA AP 経由で ZETA サーバまでは独自プロトコルでデータが送ら，アプリケーションサーバは ZETA サーバからデータを取得する．下りの通信では，アプリケーションサーバから ZETA サーバへデータを送信すると，独自プロトコルで ZETA AP を経由し ZETA 通信 MS へデータが送信される．ZETA AP 及び ZETA サーバのプログラムはユーザが改変することはできないことを考慮し，実現性の検討を行った．

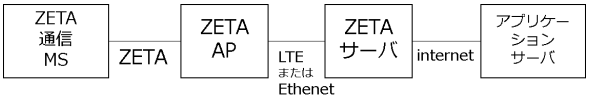

図 3: ZETA のシステム構成

2.3 ID ベース認証鍵交換

ID ベース認証鍵交換とは，公開鍵暗号の発展系である ID ベース暗号を用いた，認証付き鍵交換技術である．今回の検討は，ID ベース認証鍵交換である，藤岡らが提案した方式 [15]，富田らが提案した方式 [14] を用いて検討を行った．本プロトコルは，PKI ベースの公開鍵暗号よりも通信量が少なく [16]，LPWA 等の通信量の制約がある規格では，有効だと考えるため，今回本技術を採用した．3 章では，藤岡らが提案した方式 [15] を用い検討し，4 章では，より計算量の少ない富田らが提案した方式 [14] を用いて検討を行った．機器 A と機器 B 間で相互認証を行い，共通鍵を共有するプロトコルを表 2 に示す．事前に KGC(Key Generation Center) において，マスター秘密鍵 z と各 ID を用いて長期秘密鍵 D_x を生成し，各デバイスへ長期秘密鍵 D_x およびマスター公開鍵 Z を配布する．認証する際には，KGC から配布された長期秘密鍵 D_x，Z 及びセッションごとに生成する短期秘密鍵 x_x，短期公開鍵 X_x を用いて， 各デバイスにおいて共有鍵 K を算出する．なお，$\mathbb{G}_1, \mathbb{G}_2$ の構成として，128 ビット安全である BN 曲線上の Optimal Ate ペアリング (BN462) を用いる．

3. Cortex-M 向け TrustZone を用いた実装方式検討

3.1 市場の TrustZone for Cortex-M の利用方法

現在，市中にある Cortex-M 向け TrustZone を利用する方法を表 3 に示す．TrustZone API は，ARM が提供している API であり，ARM 社純正の開発環境である Keil を使って利用できる API である．TEE 準拠の API は，Cortex-A と Cortex-M の両方で利用するアプリケーションを作成す

表 2: 富田らの ID ベース認証付き鍵交換プロトコル

KGC (Key Generation Center)		
	$z \in \mathbb{Z}_q,\ Z = zG_1$	
$i_A = H_1(ID_A)$		$i_B = H_1(ID_B)$
$D_A = \frac{1}{z+i_A} G_2$		$D_B = \frac{1}{z+i_B} G_2$
IoT 機器 A ID_A		IoT 機器 B ID_B
$x_A \in_R \mathbb{Z}_q$	$\xrightarrow{ID_A, X_A}$	$x_B \in_R \mathbb{Z}_q$
$x_A = H_2(D_A \mid\mid r_A)$		$x_B = H_2(D_B \mid\mid r_B)$
$X_A = x_A(z+i_B)G_1$	$\xleftarrow{ID_B, X_B}$	$X_B = x_B(z+i_A)G_1$
	$d_A = H_3(X_A, ID_A, ID_B)$	
	$d_B = H_3(X_B, ID_A, ID_B)$	
$F_A = (x_A + d_A)(X_B + d_B(z+i_A)G_1)$		$F_B = (x_B + d_B)(X_A + d_A(z+i_B)G_1)$
$\sigma = e(F_A, D_A)$		$\sigma = e(F_B, D_B)$
$sid = (ID_A \mid\mid ID_B \mid\mid X_A \mid\mid X_B)$		$sid = (ID_A \mid\mid ID_B \mid\mid X_A \mid\mid X_B)$
$K = H(\sigma \mid\mid sid)$		$K = H(\sigma \mid\mid sid)$

$\mathbb{G}_1$: 楕円曲線上の群 $E(\mathbb{F}_p)$ における部分群
$\mathbb{G}_2$: $E(\mathbb{F}_{p^k})$ の部分群
G_i: $\mathbb{G}_i$ の生成元 (i=1, 2)
H_i: 文字列から $\mathbb{G}_i$ 上の元を生成する関数 (i=1, 2)
H: 鍵導出関数 , ||: 文字列の連結

る場合は，Cortex-A で広く使われている TEE 準拠を使った利用方法の API を提供する SDK を使うことで，効率的にプログラミングできると考えられる．独自の API を使った利用方法では，比較的簡単かつ安全にプログラミングできるが，特定のベンダが作成したライブラリを使うため，特定のライブラリに特化した構成となり，ベンダロックになるというデメリットもある．今回は，IoT 機器への搭載を検討しており，Cortex-A での利用を想定していない，かつ TrustZone の有効性の確認のみであるため，もっとも単純である TrustZone API を使った実装により検証を行った．

3.2 TrustZone ARMv8-M を用いた実装方法

今回開発環境として，ARM 純正の開発環境である，Arm Keil MDK を用いた．開発環境は，通常の開発環境用プロジェクトと TrustZone 用のプロジェクトの 2 つを利用し，実装を行う必要がある．電源 ON または，リセット時のプログラムの動作としては，起動時にはシステムは Secure World でコードの実行を開始し，SAU(Secure Attribution Unit) によりアクセス権限の設定が行われる．その後，Normal World へ実行環境が移り，Normal World のコードが実行される．また，関数は，すべて Normal World から実行され，Secure World には，脆弱性を含まないよう最低限の機能を置く必要がある．よって，各機能や，各データを Secure world，Normal World のどちらで取り扱えばよいか設計する必要がある．今回は，ID ベース認証付き鍵交換プロトコルにおける設計を行った構成を図 4 に示す．

ID ベース認証付鍵交換プロトコルである藤岡らの方式を使い，機器 A が機器 B と相互認証する場合の機能は，以下

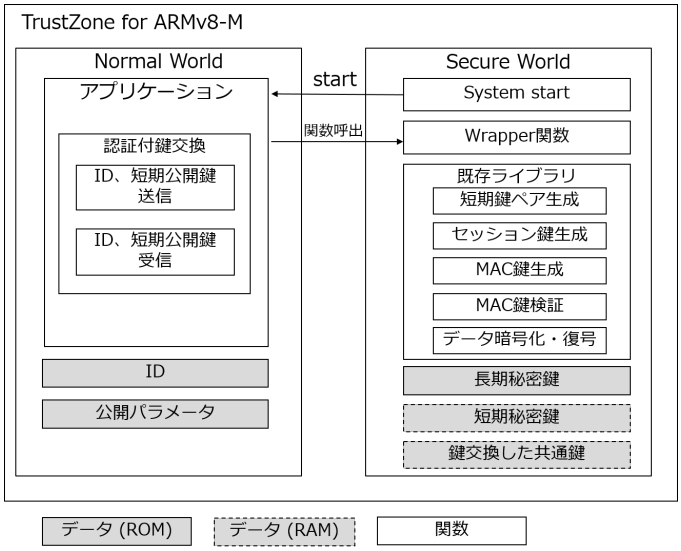

図 4: Cortex-M 向け TrustZone を利用したライブラリ構成

である．生成したセッション鍵が両者で同値であるかの検証は，セッション鍵をもとに各自 MAC 鍵を生成し，MAC 鍵を利用し生成した MAC 値を比較することにより行う．以下，epk は短期公開鍵，esk は短期秘密鍵，$mackey$ は MAC 鍵，key_{cmn} は，$macvalue$ は MAC 値，K は共通鍵，M は平文，C は暗号文を示す．

- 短期鍵ペア生成
  $gen\_ephemeral\_keypair() \rightarrow epk_A, esk_B$
- ID，短期公開鍵の送信・受信
- セッション鍵生成
  $gen\_sessionkey(ID_A, ID_B, epk_A, esk_A, epk_B, sk_A, mpk) \rightarrow K, sid$

表 3: Cortex-M 向け TrustZone の利用方法

利用する API	メリット	デメリット	SDK の例
TrustZone API	下位 SDK にロックインされずに，TrustZone 利用プログラムを作成できる	セキュアプログラミングを徹底する必要がある	Keil CMSIS[11]
TEE 準拠の API	・TEE の API 知識を活用して，TrustZone 利用プログラムを作成可能 ・Cortex-A とコードを共通化可能	・Cortex-M では不要な API が多い ・SDK は実質的なベンダロックインとなる	NetNucleus SP[12]
独自の API	SDK 開発元の定義した API で，比較的簡単・安全に TrustZone 活用プログラムを作成できる	ベンダロックインとなる	Ubiquitus Securus[13] 他

- MAC 鍵生成
  $gen\_mackey(K)$
  $\rightarrow (mackey_A, mackey_B)$
- MAC 値生成
  $gen\_mac(sid, mackey_A) \rightarrow macvalue_A$
- MAC 値検証
  $verify\_mac(sid, mackey_A, macvalue_B) \rightarrow 0$(成功), 1(失敗)
- セッション鍵によるデータの暗号化・復号
  $enc(M, K) \rightarrow C$
  $dec(C, K) \rightarrow M'$

認証プロトコルを実行する際の通信機能は，外部からの攻撃を受けやすい部分であるため，Normal World から実行する必要がある．上記のプロトコルにおいて，認証プロトコルの機能のうち，短期秘密鍵 esk，長期秘密鍵 sk，セッション鍵 K，MAC 鍵 $mackey$ は漏洩するとなりすましされる可能性があるため，これらを使う演算は，Secure World で実行する必要がある．よって，上記のプロトコル機能のうち，ID，短期公開鍵の送信・受信以外のプロトコルは，セキュア領域で行う必要がある．

また，これらの関数をセキュア領域に置いたとしても，関数の引数に秘匿したい秘密鍵が含まれている場合には，関数の呼び出しは Normal World からのみ実行可能なため，結局秘匿したいデータを Normal World が知ることになってしまう．よって，セキュア領域を使う構成の場合，従来のライブラリ構成だとあまり気にしない点である引数についても考慮する必要がある．引数を Normal World へ渡さない方法として，wrapper 関数を定義し，従来のライブラリでは引数となっている秘密情報を秘匿するという方法がある．よって，認証付き鍵交換プロトコルで利用する情報をして，外部へ漏洩したくない情報を洗い出し，それらを関数の引数として利用しないように構成を変更した．ID ベース認証付き鍵交換の演算においては，長期秘密鍵 esk，長期秘密鍵 sk，セッション鍵 K，MAC 鍵 $mackey$ がそれらに当たる．以下に，セッション鍵を生成する際の wrapper 関数の例を示す．図 fig:wrapper に示した関数を Normal World から呼ぶことにより，Secure World でのみ秘密情報を保持することができる．

```
static const char srv_ssk[] =
"040E7FB8044C030A37A3A5EAFA7CC53D6D34749359D6C0D6A660734A3116554BE71
E3BFD95A236BB26973EC08C9AB87EC6647B447720C46E4E4FAE266954A38F96";
                ⋮
// wrapper関数(ses_key(セッション鍵),ssk(長期秘密鍵), epk(短期秘密鍵)
// を引数から除く))
int wrapper_gen_session_key(sid, id_send, id_recv, epk_recv,
epk_send, mpk)
{
  //既存ライブラリコードを呼ぶ
  return gen_session_key(ses_key, sid, id_send, id_recv, device_id,
  srv_ssk, esk_send, epk_send, epk_recv, esk_send, mpk);
};
```

図 5: wrapper 関数例 (Secure World での実装)

長期秘密鍵は，製造時にデバイスごとに書き込まれることを想定している．よってデバイスで固定の値として，Secure World で保持するように実装した．セキュア領域プロジェクト内のファイルに const 宣言した値は，Secure World の ROM 領域にコーディングされるため，長期秘密鍵は，コーディング時に const で指定した．短期秘密鍵，セッション鍵，MAC 鍵については，起動後演算により生成される値であるため，演算プログラムをセキュア領域プロジェクト内のファイルで定義しておくことで，Secure World の RAM 側に置かれることとなる．

Normal World から，セキュア領域の関数を直接たたくことが可能な代わりに，Keil の開発環境においては，関数宣言時に以下のようなセキュアプログラミングをする必要がある．関数を宣言するときに以下の文字列を末尾に付ける．　__attribute__((cmse_nonsecure_entry));
この Attribute が適用されていると，Normal World にレジスタがスイッチするときに初期化されるため，秘密情報が漏洩しない [11].

今回は，ID ベース認証付き鍵交換のコア演算を Normal World へ秘密情報を渡さずに実装する方式を設計し，FPGA ボード上でコア演算を Secure World で実行する動作検証を行った．ARM の FPGA ボードにて，実現性の検討を行い，Cortex-M33 のイメージを書込み TrustZone を用いた検証を行った．表 4 に検証に利用したデバイスの性能を示す．

上記デバイスにおいて，TrustZone を用いて，図 4 に示した構成における ID ベース認証付き鍵交換の暗号プロト

表 4: 検証デバイス

開発ボード	ARM Cortex-M Prototyping System (MPS2+)
CPU コア	ARM Cortex-M
SRAM	8 MB single cycle SRAM
	16 MB PSRAM
FPGA イメージ	Cortex-M33 Dual core
周波数	20MHz
開発環境	Arm Keil MDK

コルである，各機能が TrustZone 上で動作することが分かった．

4. ZETA を用いた実現性検討

LPWA の一種である ZETA を用いて，2.3 に述べた ID ベース認証付き鍵交換方式を実装する場合の実現性の机上検討を行った．

4.1 データフォーマットについて

2.2 で述べたように，LPWA では，一度に送信できるデータサイズは制限されている．また，2.3 の ID ベースの認証付き鍵交換を行う場合に，送受信が必要なデータサイズは表 5 の通りである．なお，表 5 に示す短期公開鍵は楕円曲線上の点であるため，x 座標と y 座標のデータであるが，ZETA での通信速度を考慮し暗号処理の速度よりも，送信するデータサイズを小さくすることを優先するため，x 座標のデータのみ送り，y 座標は受信側で計算することとする．これらより，相互認証に必要な情報を複数回に分け，送られてきたデータを復元する必要がある．受信データを復元するために，データを復元する順序をヘッダで管理する必要があり，またそれ以外にもデータを識別する必要があるため，共通ヘッダを定義し，送受信するデータに付与する必要がある．図 6 に，送信データサイズを今回の検証対象 ZETA のデータサイズである 50byte とした場合の共通ヘッダのデータフォーマットを示す．応答種別とは，今何のデータを送受信しているかを判断するために必要であり，ID ベース認証付き鍵交換の場合には以下のイベントが考えられる．

(1) IoT 機器とサーバ間の認証プロトコルの往復
(2) 暗号化データの送受信
(3) 送信成功，失敗の通知
(4) 秘密鍵の更新

よって，最低限これらを識別できるサイズは必要となる．また，セッション ID は IoT 機器，サーバ間のセッションを識別するための ID であり，同一セッション内では，鍵交換した共通鍵が一意に決まるため，暗号化データをやり取りする場合に重要となる．セッション ID のサイズは，ZETA AP 配下にあるデバイスが識別できれば良いため，16byte とした．Index サイズは，送信するデータサイズは表 5 に示した通りであるため，3，4 回でデータは送信できると考えられるため，今回は 4byte とした．

表 5: 富田らの ID ベース認証付き鍵交換のデータサイズ (曲線パラメータ：BN462)

データの種類	データサイズ
デバイス ID	50 byte (デバイスの ID サイズによる)
短期公開鍵	59 byte
MAC 値	32 byte

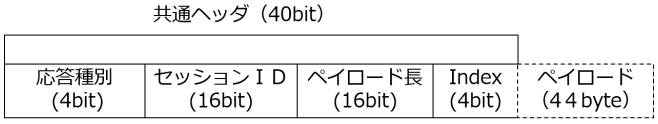

図 6: ZETA 環境におけるデータフォーマット

4.2 通信回数，時間に関する検討

4.1 で設計したデータフォーマットを前提とし，ID ベース認証付き鍵交換にかかる時間を測定し，ZETA 環境で 50byte のデータの送信にかかる時間，ID ベース認証付き鍵交換の処理時間から，ZETA 環境における ID ベース暗号認証付き鍵交換にかかる時間を机上検討する．ZETA 環境の上りの通信は，ZETA 通信モジュールからデータを送信してから，ZETA サーバからデータをサブスクライブするまでの時間を測定し，下りの通信は，ZETA サーバへデータを送信してから，ZETA 通信モジュールのついた PC で受信するまでの時間を測定した．実験環境としては，ZETA 通信モジュールを UART 経由で PC と接続し，データを送信した．また，受信側では，サブスクライブする python のプログラムを作成し，Topic を指定し，データを取得した．また，ZETA 通信プロトコルとしては，ZETA-P と呼ばれる 300bps の通信プロトコルを用いた．表 6 に，上りと下りの 1 回の通信にかかった時間の平均値を示す．10 回の平均値をとった．また，ZETA 環境において想定される通信シーケンスを図 7 示す．表 6，表 7，図 7 より，約 34 (3909 × 5 + 3272 × 4 + 1421) 秒程度で，ID ベース認証付き鍵交換が実施できる．そのため，分単位のリアルタイム性を求めないユースケースであれば，ID ベース認証付き鍵交換は ZETA 環境で実用的に利用できると言える．また，暗号処理時間よりも通信時間が占める時間のほうが圧倒的に多く，リアルタイム性を求めるにシステムの認証においては，通信量を下げることが重要だと考えられる．

5. ZETA 環境における TrustZone を利用したシステムの提案

LPWA 環境では，ZETA と同様に経由する LPWA サー

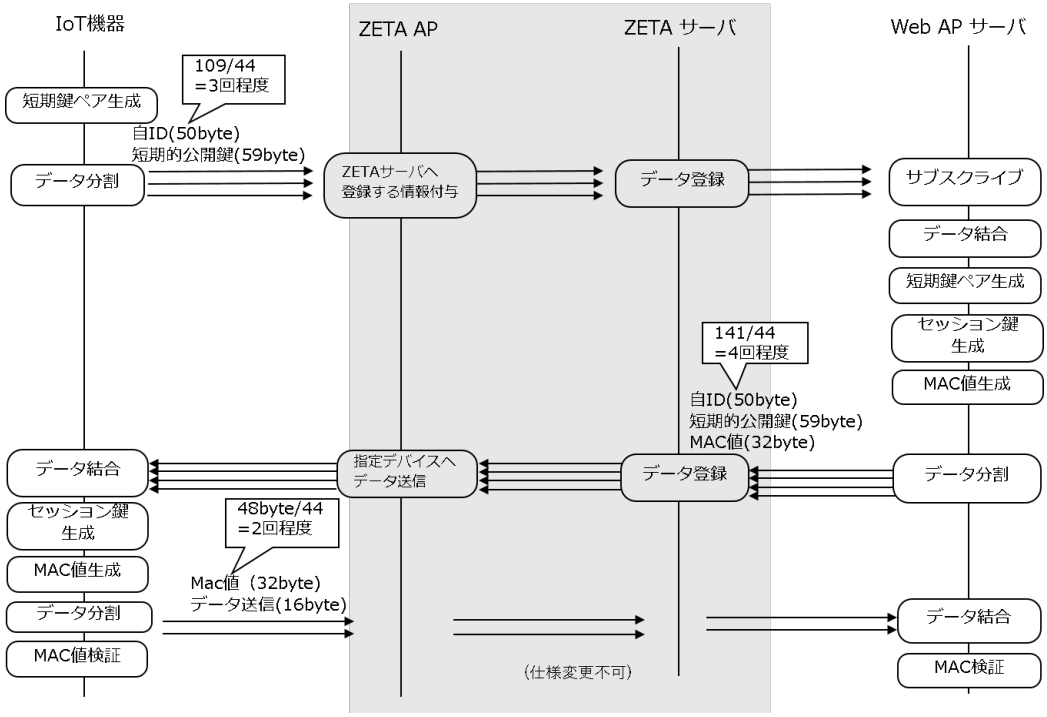

図 7: ZETA のシーケンス

表 6: ZETA の通信速度 (送信データ量：50byte)

データ方向	通信時間
上り	3272 msec
下り	3909 msec

表 7: Cortex-M7 における処理性能 [14]

処理	処理時間
富田らの方式	1421 msec
G1 のスカラー倍算	128 msec
G2 のスカラー倍算	237 msec
ペアリング	737 msec

ビス提供者が運営するサーバを経由する構成がほとんどである．また，LPWA サービス提供者が運営するサーバと IoT 機器間は仕様として認証機能や暗号機能が含まれていることが多いが，LPWA を利用した Web サービスを構築する場合に，Web サーバー IoT 間の認証や暗号化等の機能は各自対応する必要がある．よって，そのような環境では以下のようなシステムがよいと考えられる．図 8 に提案する ZETA 環境における TrustZone を利用したシステム構成を示し，図 9 には ZETA 環境でのシーケンス図を示しており，図 9 の IoT 機器内の Normal World，TrustZone 間のやり取りを示した．Secure World で生成した通信相手に送信する情報を Normal World へ渡し，データの分割，結合するデータは秘匿情報ではないため，ZETA 送信向けにデータを加工する処理は，Normal World で実施する．ID ベース認証付き鍵交換は IoT 機器－ Web アプリケーションサーバ間で行い，それぞれに秘密鍵を配布し認証，鍵共有を行うことで，データのやり取りに分割，結合は必要になるが，end-to-end で安全なシステムを構築できる．

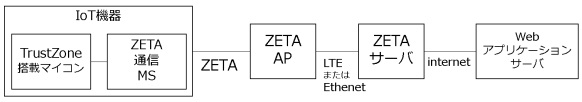

図 8: ZETA 環境における TrustZone を利用したシステム

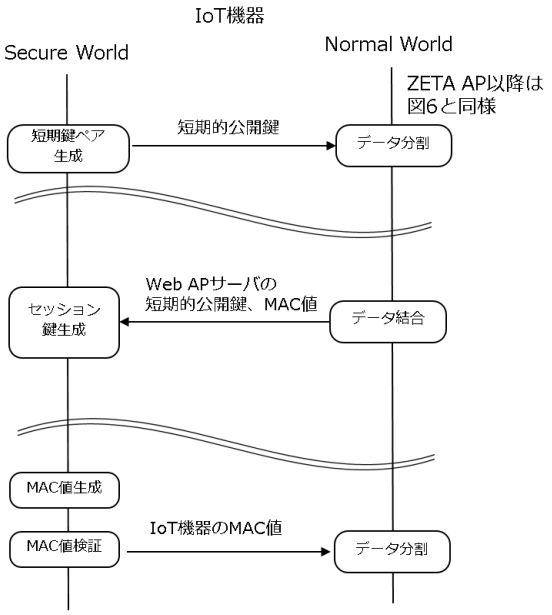

図 9: ZETA 環境における TrustZone のシーケンス

6. 今後の課題

TrustZone については，今回の実験では，認証アプリケーションを既存のライブラリを用いて実装する場合の実装方針について示した．しかし，定期的に長期秘密鍵を更新する利用方法も考えられ，更新時にセキュア領域のみで長期秘密鍵が扱われるように考慮したライブラリ構成の設計が必要である．また，通信機能は外部からの侵入の可能性を考え，Normal World に置くほうが良いと考えられる．しかし，そのような機構にする場合，Secure World で生成した情報は Normal World からは閲覧ができないため，送信する短期公開鍵の情報を Normal World から閲覧できるようにメモリコピーする必要があり，メモリ開放用の API が必要となる．セキュアに Secure World のデータを Normal World へ渡す API の設計が必要である．デバイスについては，今回はメモリ量が潤沢な FPGA ボードに実装し実現性を検討したが，実際の Cortex-M33 搭載ボードを用いて検証し，処理時間等計測する必要がある．

ZETA については，今回机上で最大送信データサイズである 50byte のデータ送信にかかる時間，及び認証にかかる時間から ID ベース認証付き鍵交換にかかる時間について算出したが，実データを用いて実際の認証付き鍵交換プロトコルにかかる時間を計測する必要がある．

7. まとめ

今回は，今後広がると想定されるセキュリティを考慮した IoT システムを対象とし，暗号プロトコルの実現性の検討を行った．特に TrustZone の利用における，セキュア領域を用いた ID ベース認証付き鍵交換プロトコルにおけるライブラリの構成について検討した．TrustZone 等で実現できるセキュア領域を用いない既存のライブラリと同じ構成では，Secure World で実装できてもセキュアに利用できないことを示し，TrusZone 利用時のライブラリ構成案について示した．提案した構成により，外部から攻撃可能な Normal World へは漏洩するとなりすましが可能になる長期秘密鍵やセッション鍵の情報を公開させずに，Secure World での処理が可能になった．

また LPWA の利用における ID ベース認証付き鍵交換を実現するためのデータフォーマットの提案及び，実現性の検討をおこなった．机上検討の結果，34 秒程度で相互認証が実現できることが分かった．分単位のリアルタイム性であれば十分に担保できることが分かった．また，セッションを張った状態であれば，認証が必要なのは初めの一回であり，それ以降は交換した共通鍵ベースの共有鍵で暗号化データを授受できるため，この程度の時間で実現できれば，秒単位のリアルタイム性を求められない IoT システムでは十分に実用的だといえる．また，暗号処理時間よりも，LPWA の通信の占める割合が多いため，低通信量で認証できる ID ベース認証付き鍵交換のようなプロトコルが LPWA でには適していると考える．

これらより，将来 IoT システムの構成として考えられる LPWA の環境における TrustZone を使った場合の公開鍵ベースの暗号プロトコルにおけるシステムを提案し，分単位のリアルタイム性で十分なシステムであれば実用的に利用できることを示した．

参考文献

[1] 総務省：平成 30 年版　情報通信白書 p13, http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h30/pdf/n1100000.pdf(2018).

[2] Usman Raza, Parag Kulkarni, and Mahesh Sooriyabandara : Low Power Wide Area Networks: An Overview, IEEE Communications Surveys & Tutorials(2017).

[3] IPA : IoT 開発におけるセキュリティ設計の手引き, https://www.ipa.go.jp/files/000052459.pdf(Aplil 2019).

[4] Global Platform : TEE System Architecture Version 1.1.0.10 (Target v1.2), https://globalplatform.org/wp-content/uploads/2018/09/GPD_TEE_SystemArch_v1.1.0.10-for-v1.2_PublicReview.pdf (September 2018).

[5] FFRI Inc. : ARMv8-M TrustZone:組み込みデバイス向けアーキテクチャとセキュリティ機能，https://www.ffri.jp/assets/files/monthly_research/MR201602_ARMv8-M_TrustZone\%EF\%BC\%9A_A_New_Security_Feature_for_Embedded_Systems_JPN.pdf (2019-8-6 閲覧).

[6] 鄭 立 : IoT ネットワーク LPWA の基礎 -SIGFOX，LoRa，NB-IoT- ，リックテレコム (2017).

[7] 京セラコミュニケーションシステム株式会社：Sigfox ネットワーク紹介資料, http://www.soumu.go.jp/main_content/000597413.pdf(2019).

[8] Sigfox : Device management, https://support.sigfox.com/docs/device-idpac-couple(2019.08.05 閲覧).

[9] Techsor Inc. : 次世代 LPWAN 規格 ZETA の紹介，http://www.jasa.or.jp/TOP/download/technical/ZETA_2018-06-29_printed.pdf(2019.08.05 閲覧).

[10] ARM Ltd. : Arm TrustZone Technology for the Armv8-M Architecture Version 2.1, https://static.docs.arm.com/100690/0201/armv8_m_architecture_trustzone_technology_100690_0201_01_en.pdf(2018).

[11] ARM Ltd. : Using TrustZone on ARMv8-M MDK-Professional Tutorial，http://www.keil.com/appnotes/files/apnt_291.pdf(2016).

[12] 東芝情報システム株式会社 : IoT 機器のデータを守る「NetNucleus SP」，https://www.tjsys.co.jp/embedded/netnucleus-sp/index_j.htm (2019-08-05 閲覧).

[13] 株式会社ユビキタス AI コーポレーション：Ubiquitous Securus, https://www.ubiquitous-ai.com/products/security/securus/(2019-08-05 閲覧).

[14] J.Tomida, A. Fujioka, A. Nagai, and K, Suzuki : Strongly Secure Identiry-Based Key Exchange with Single Pairing Operation，ESORICS2019(2019).

[15] A. Fujioka, K. Suzuki, and B. Ustaoglu,Ephemeral key leakage resilient and efficient ID-AKEs that can share identities, private and master keys, Pairing 2010, pp.187-205(2010).

[16] 木下，鈴木，永井：TLS1.3 への ID ベース認証鍵交換の適用と実装評価，CSS2019(2019).