Vol.2013-OS-126 No.19
2013/8/1

高速なOS切替え機構を有する組込み機器向けセキュアモニタ LiSTEE^{TM}

金井 遵[1]　磯崎 宏[1,2]

概要：本稿では ARM TrustZone 機能を利用した OS の高速な切替え機能を有するセキュアモニタ LiSTEE モニタの設計と実装および評価について述べる．ARM プロセッサなどプロセッサのモード毎にレジスタがバンクされる環境においては，OS の切替え時に待避するレジスタ数が多くなる傾向にあり，OS 切替えが低速になりやすいという問題がある．そこで本研究では，バンクレジスタの破壊や参照が起きると問題があるかどうかをセキュリティ情報と呼ぶこととし，OS が利用するモードとセキュリティ情報から，待避・復帰処理が不要なレジスタと必要なレジスタを判定し，セキュリティを担保しつつコンテキスト待避・復帰処理を高速化する手法を提案した．さらに本手法を組み込んだセキュアモニタ”LiSTEE モニタ”の設計と実装を行った．評価より本手法を適用しない場合との比較より，OS 切替え時間を最大 51.5%削減でき，実アプリケーションを使った評価では AES によるデータ暗号化処理において 9.2%，XOR によるデータ暗号化処理において 28%の高速化効果を確認した．

1. はじめに

近年，ネットワークに接続される組込み機器の増加や高機能化に伴い，組込み機器においてもセキュリティが重要な課題となっている．特に，個人情報が扱われるスマートフォンやタブレット，ヘルスケア機器，スマートメータなどにおいては，情報漏洩がもたらす影響は大きく，セキュリティの確保は急務である．しかしながら，組込み機器の OS としても広く利用されるようになった Linux は OS の規模が非常に大きく，攻撃の糸口となる脆弱性が混入しやすい．中でも Linux の管理者権限が奪取されると Linux 元来のファイルやプロセスへのアクセス制御が効かなくなるため情報の不正な取得を目的とした攻撃の糸口となりうるが，実際に管理者権限への権限昇格が可能なセキュリティホールは毎月のように発見されている [1]．アクセス制御の強化やウィルス対策，ホスト型の侵入検知，パーソナルファイヤウォールなど，Linux のセキュリティを強化する試みも多く行われているが [2][3]，カーネルの脆弱性によってはこれらの対策ソフトウェアも無効化される恐れがあるため，十分な対策とは言えない．

一方で Linux の規模を鑑みるとこのような脆弱性を全て無くすのは現実的ではない．このような背景から，例えば組込み機器で多く利用される ARM 社の一部プロセッサには TrustZone[4] と呼ばれる機能が搭載されている．TrustZone は一種の仮想化支援機能であり，秘匿性や完全性，可用性が要求されるソフトウェア処理を一般の Linux とハードウェア的に分離して実行することができる．TrustZone 機能を利用すると，Linux の管理者権限が奪取されたとしても，Linux 側からは秘匿性や完全性，可用性が要求される処理が実行されている領域のメモリ内容を参照することは不可能であるため，情報流出や処理やデータの改竄などを防止することができる．

TrustZone 機能を利用して Linux と秘匿性等が要求される処理を分離して実行するためには，セキュアモニタと呼ばれる特殊なソフトウェアが必要となる．Linux などの各種 OS はこのセキュアモニタ上で動作することとなる．既に TrustZone 対応セキュアモニタとして多数のソフトウェアが提案されている [5], [6]．セキュアモニタは Linux と秘匿性等が要求される処理を切替えて実行するため，OS 切替えを行う機能を備える必要がある．この OS 切替えが低速な場合，頻繁に OS 切替えが発生するとシステム全体の性能低下を招く．特に OS 切替えにはレジスタ等のコンテキストの待避と復帰が必要になるが，ARM プロセッサではプロセッサのモード毎に一部のレジスタがバンクされており，レジスタの数が多いため，一般に OS 切替え処理が低速になりやすい．上述のセキュアモニタ [5], [6] においても全てのレジスタを待避，復帰させており，OS 切換えが

[1] (株) 東芝 研究開発センター
Corporate R&D Center, Toshiba Corporation, Japan
[2] 慶應義塾大学
Keio Uniersity

頻繁に起きるとシステム全体の性能低下を招きやすい．

そこで我々はOS切替え時のコンテキスト待避，復帰処理を高速する手法を提案し，本手法を組み込んだTrustZone向けセキュアモニタLiSTEEモニタを設計，実装した．本稿では，OS切替えを高速化した組込み向けセキュアモニタLiSTEEモニタの設計と実装，評価について述べる．

2. ARMアーキテクチャの概要

本項ではLiSTEEモニタが前提とするARMv7アーキテクチャについて概説する．

2.1 モードとレジスタの構成

表1に示すようにARMプロセッサにおいては処理内容に応じて細かくプロセッサの実行モードが分かれている[*1]．モードの切替えはOSなどのプログラムによって自発的に切替える場合や，ソフトウェア例外およびハードウェア例外の発生などの要因によって自動的に切り替わる場合の二種類がある．

また，ARMプロセッサではモード毎にレジスタの実体が異なるバンクレジスタの概念も導入されている．例えば，通常はスタックポインタとして利用される同じr13と呼ばれるレジスタであっても，USRモードとSVCモードではハードウェアとしては異なる参照先となる．これにより，USRモードとSVCモードを切替える際にはr13レジスタについては待避の必要がなくなるため，モード切替時の処理を高速化することが可能である．その一方で，モード間で実体が同一となる，非バンクレジスタも存在する．非バンクレジスタについてはモード切替時に待避，復帰処理が必要となる．バンクレジスタはモード切替時のコンテキストスイッチを高速化することができる一方で，実レジスタ数が多くなり，プロセッサのハードウェア規模が大きくなるという欠点も存在する．

2.2 TrustZone

ARMプロセッサにはTrustZoneと呼ばれるセキュリティ技術が搭載されている．TrustZoneでは，セキュアワールドとノーマルワールド（ノンセキュアワールド）と呼ばれる2つのワールドでプログラムを隔離して実行することができる．一つのプロセッサ内では，ワールドを切替えてプログラムを実行する．セキュアワールドとノーマルワールド両方で，表1に示したプロセッサの各モードが利用できる．

セキュアワールドとノンセキュアワールドではそれぞれ別々のOSを動作させることができ，それぞれセキュアOS，ノンセキュアOSと呼ばれる．例えばノンセキュアワールドでLinuxを動作させ，セキュアワールドで独自OSを動かすといったことが可能になる．TrustZoneを使うとセキュアワールドで利用するメモリ等のリソースについて，ノンセキュアワールドからのアクセスに対してアクセス制御を行うことができる．このため，たとえノンセキュアワールドで動作するLinuxに脆弱性があり管理者権限が奪取された場合でも，セキュアワールドで動作するOSやプログラムには影響を及ぼさない．

セキュアワールドとノンセキュアワールドでOSを並行動作させるためには，時分割でノンセキュアワールドとセキュアワールドを切替える必要がある．ワールドの切替えを行うソフトウェアのことはモニタと呼ばれ，各ワールドのプログラムからはセキュアモニタコールと呼ばれる命令（smc命令）で呼び出すことができる．モニタはモニタモードと呼ばれる特別なモードで動作し，常にセキュアワールドで動作する．また，各ワールドでそれぞれアクセスできるメモリやペリフェラルをアドレス別やペリフェラル別にTZASC（TrustZone Address Space Controller），TZPC（TrustZone Protection Controller）と呼ばれる機能を使って設定することができる．

ここで，各ワールド間では汎用レジスタおよびセーブドプログラムステータスレジスタ（spsr）はバンクされない．例えばセキュアワールドのUSRモードでr13を参照した場合とノンセキュアワールドのUSRモードでr13を参照した場合には，同じレジスタが参照される．すなわち一般的に，モニタによるOS切替え時においてはOS内でのモード切替時と異なり，非バンクレジスタとバンクレジスタ両方の待避が必要となる．このためARMプロセッサではOS切替え時には多くのレジスタの待避が必要となる．ARMプロセッサにおいては汎用レジスタ(r0–r15)およびspsrの実個数はMONモードのバンクレジスタを除くと40個であり，プログラムカウンタを除く39個のレジスタの待避・復帰処理，すなわち156byteのレジスタ待避・復帰処理が必要になる．

3. コンテキスト切替え処理の高速化方式

既に述べたようにARMプロセッサ環境下でOS切替えを実現する場合，待避・復帰を行うレジスタ数が多く，切替えが低速になりやすい．そこで我々はコンテキスト待避・復帰処理を高速化する手法を提案した．本章では本稿で提案する高速化手法について述べる．

3.1 高速化方式の概要

本研究では，OS切替え時のコンテキスト待避・復帰を高速化するにあたり，必ずしもOSが全てのモードを利用しない点に着目した．例えば，ARM向けLinuxを標準状態でコンパイルするとFIQモードは利用されないコードが生成される．この場合，たとえもう一つのOSがFIQモードを利用するとしてもLinux側ではFIQモードは利用

[*1] さらにOSからユーザモードのコンテキスト等にアクセスするためのSYSモード（特権ユーザモード）が存在する

表 1　ARM プロセッサのモードとバンクレジスタ一覧

モード名	説明	バンクレジスタ	バンクレジスタ数
SVC	スーパーバイザモード	r13–r14, spsr	3
USR	ユーザモード	r8–r14	7
IRQ	通常割込みモード	r13–r14, spsr	3
FIQ	高速割込みモード	r8–r14, spsr	8
ABT	アボート例外モード（データ/命令アボート）	r13–r14, spsr	3
UND	未定義命令例外モード	r13–r14, spsr	3
MON	モニタ用モード	r13–r14, spsr	3

されないため，Linux 側では FIQ モードのバンクレジスタは破壊しない．すなわち，OS 間遷移時に FIQ モードのバンクレジスタの待避・復帰は省略することができる．まとめると，OS の各モード利用の有無によるバンクレジスタ待避の必要性についての基本的な考え方は以下のようになる．

- 両方の OS がモード A を利用しない場合：
  モード A のバンクレジスタの待避・復帰は必要なし
- 片方の OS のみがモード A を利用する場合：
  モード A のバンクレジスタの待避・復帰は必要なし
- 両方の OS がモード A を利用する場合：
  モード A のバンクレジスタの待避・復帰は必要あり

また，非バンクレジスタについてはモード利用の有無にかかわらず待避・復帰処理が必要になる．モード利用の有無は OS やプログラムの設計によって決まるため，待避・復帰が必要なバンクレジスタと待避・復帰が必要ないバンクレジスタについて，あらかじめ決定することができる．

3.2　セキュリティの問題

前項の方法により待避が必要ないと判定されたレジスタを待避させない場合，セキュリティ上問題が発生する可能性がある．この問題には，待避させないバンクレジスタの (a) 不正破壊，(b) 不正参照の 2 つがある．

これは，ARM プロセッサではモニタ等からノンセキュア OS のモード変更の検知や制限が困難であることが原因である．すなわち，攻撃者が意図的にノンセキュア OS が通常利用しないモードに切替えて，待避されていないセキュア OS 側の当該モードのレジスタの内容を参照したり，変更したりすることが可能となってしまう．例えば Linux を FIQ を使わないようにコンパイルしたとしても，カーネルレイヤで動作するマルウェアが意図的に FIQ モードに切替えれば，FIQ モードのバンクレジスタにアクセスすることができ，そのアクセスをモニタ等から検出することは困難である．また，OS 開発者に攻撃の意図がない場合でもプログラムのミスにより，利用しないはずのモードに切替えて，バンクレジスタを破壊してしまう可能性もある．

このため，前項の方法により待避が必要ないと判定されたレジスタについても，当該モードを使うワールドからの遷移時に待避処理を行い，その後レジスタクリアを行う（レジスタ復帰は行わない）こともできるようにした．別のワールドからの遷移時には復帰処理のみを行い，レジスタ待避は行わない．これにより，(a)，(b) 両方の攻撃を防ぐことができる．一般にレジスタクリアはメモリ参照に比べ高速である．提案手法では，一回ずつのレジスタ待避・復帰処理に加えてレジスタクリアが一回で済むが，通常はワールド間の往復でセキュアワールドのコンテキスト待避・復帰，ノンセキュアワールドのコンテキスト待避復帰が必要である．すなわち，レジスタ待避・復帰が二回ずつ必要になるため，提案手法による高速化が期待できる．なお，(a) のみを防げればよい場合，すなわちレジスタを破壊されては困るが値を見られても良い場合には待避・復帰処理のみを行えば良く，ゼロクリアは必要がないため，さらに高速化が可能である．

3.3　アルゴリズムの詳細

3.1 節および 3.2 節で述べた考え方に基づく待避レジスタの決定アルゴリズムを図 1 に，コンテキスト待避・復帰アルゴリズムを図 2 に示す．

```
dir = 0(secure->nonsecure) or 1(nonsecure->secure)
for each m of modes do
  type[dir][m] = NONE
  if use_flag[0][m] && use_flag[1][m] then
    type[dir][m] = SAVE
  else if use_flag[dir][m] then
    type[dir][m] = security_flag[dir][m]
  end if
end do
```

図 1　待避するレジスタの決定アルゴリズム

待避レジスタの決定には入力として，セキュア/ノンセキュアワールドで当該モードを利用するかどうか (use_flag[world][mode]，ただし world=0 がセキュアワールド，1 がノンセキュアワールド)，バンクレジスタの破壊・参照を防ぐかどうか（security_flag[world][mode]）を与える．security_flag としては，破壊のみを防ぐ場合には SAVE，参照も防ぐ場合には SAVE_AND_CLEAR，OS 側により退避・クリア処理が行われることが保障される場合など，参照・破壊を防がなくてよい場合には NONE を設定する．

```
function save_restore_context(dir)
  save_nonbanked_registers(dir)
  for each m of modes do
    // save
    if type[dir][m] == SAVE then
      save_banked_registers(dir, m)
    else if type[dir][m] == SAVE_AND_CLEAR then
      save_banked_registers(dir, m)
      clear_banked_registers(m)
    end if

    // restore
    if type[dir^1][m] != NONE then
      restore_banked_registers(dir, m)
    end if
  end do
  restore_nonbanked_registers(dir)
end function
```

図 2 コンテキスト待避・復帰アルゴリズム

各モードについて，両方の OS が当該モードを利用する場合にはレジスタ待避フラグを立て（SAVE に設定）る．一方，そのモードを切替え元 OS のみが利用する場合には，切替え元 OS の security_flag を参照して，レジスタ破壊のみを防ぐ場合にはレジスタ待避フラグを立て(SAVE に設定)，参照も防ぐ場合にはレジスタ待避フラグに SAVE_AND_CLEAR を設定する．それ以外の場合にはレジスタ待避フラグは立てない．

コンテキスト待避・復帰時，すなわち OS の切替え時にはレジスタ待避フラグを参照して待避復帰処理を行う．まず，切替え元ワールドのコンテキスト保存領域へ非バンクレジスタの待避処理を行う．さらに各モードについて，type[切替え元のワールド][mode] が参照され，設定された値によってレジスタ待避とクリアが行われる．さらに type[切替え先のワールド][mode] が参照され，値によってレジスタ復帰が行われる．最後に，切替え先ワールドのコンテキスト保存領域から非バンクレジスタの復帰処理を行う．

例えば，セキュアワールドで FIQ モードを使い，ノンセキュアワールドで FIQ モードを使わない場合かつ，セキュアワールド側 FIQ モードの security_flag が SAVE_AND_CLEAR に設定された場合，type[0][FIQ] は SAVE_AND_CLEAR，type[1][FIQ] は NONE となり，ノンセキュアワールドへの遷移時にはレジスタの待避とクリア処理が行われ，セキュアワールドへの遷移時にはレジスタの復帰処理のみが行われる．

3.4 高速化の例

ノンセキュア OS として Linux を，セキュア側に搭載する機能や OS としていくつかのパターンを想定して，レジスタ待避・復帰が必要なレジスタについて考察した結果を表 2 に示す．なお，Linux は SVC/USR/IRQ/ABT/UND モードを利用する一般的な設定を想定する．

- Linux ＋単一機能
  ノンセキュア OS として Linux を，セキュア側の処理として割り込みを利用しない単一機能を動かすような場合，セキュア側には OS を搭載する必要は必ずしもない．例えば特定の暗号化処理や暗号方式の変換処理だけをセキュア側で動かすような場合が該当する．この場合，セキュア側では SVC モードのみを利用してこれらの処理を実装すれば十分である．このとき，非バンクレジスタ (r0–r12) と SVC モードのバンクレジスタ（r13–r14,spsr）のみを OS 切替え時に待避・復帰させる必要がある．よって待避・復帰させるレジスタは 39 レジスタから 16 レジスタまで削減できる（表 2(2)）．
- Linux ＋ Mini OS（割込み入力無し）
  ノンセキュア OS として Linux を，セキュア側の処理として単純な Mini OS とその上で動くアプリケーションを動かすような場合を想定する．さらにこの Mini OS ではデータアボートや未定義命令への対応は行わないものとする．このとき，セキュア側では SVC モードと USR モードを利用することとなる．すなわち，非バンクレジスタと SVC/USR モードのバンクレジスタのみを OS 切替え時に待避・復帰させる必要がある．この場合は待避・復帰させるレジスタは 22 レジスタとなる（表 2(3)）．
- Linux ＋ Mini OS（FIQ 割込み有り）
  ノンセキュア OS として Linux を，セキュア側の処理として単純な Mini OS とその上で動くアプリケーションを動かし，さらに割り込みを受け付ける場合，セキュア側では SVC モードと USR モードに加えて一般的には FIQ モードを利用することとなる．一方 Linux 側ではハードウェア割り込み処理には IRQ モードを利用する．すなわち，こちらも非バンクレジスタと SVC/USR モードのバンクレジスタのみを OS 切替え時に待避・復帰させる必要がある．よって待避・復帰させるレジスタは 22 レジスタとなる（表 2(4)）．なお，FIQ モードのレジスタを破壊されると困る場合には，セキュアワールドからノンセキュアワールドへの遷移時には 8 レジスタの待避とゼロクリア，ノンセキュアワールドからセキュアワールドへの遷移時には 8 レジスタの復帰を行う必要がある（表 2(5)）．
- Linux ＋一般 OS
  ノンセキュア OS として Linux を，セキュア側としてアボート処理や未定義例外対応処理などまで対応した OS を動かす場合，セキュア側では SVC/USR/FIQ/ABT/UND の各モードを利用することとなる．この場合，待避・復帰させるレジスタは

表 2　高速化の例

パターン	NonSecure → Secure 遷移時処理	Secure → NonSecure 遷移時処理
(1) 既存手法	SVC/USR/FIQ/ABT/UND/IRQ/FIQ 待避・復帰	同左
(2)Linux+単一機能	SVC 待避・復帰	同左
(3)Linux+Mini OS(割込み無し)	SVC,USR 待避・復帰	同左
(4)Linux+Mini OS(FIQ 割込み有り)	SVC,USR 待避・復帰	同左
(5)Linux+Mini OS(FIQ 割込み有り/セキュリティ)	SVC,USR 待避・復帰，FIQ 復帰	SVC,USR 待避・復帰，FIQ 待避・クリア
(6)Linux+一般 OS	SVC/USR/ABT/UND 待避・復帰	同左
(7)Linux+一般 OS(セキュリティ)	SVC/USR/FIQ/ABT/UND 待避・復帰，FIQ 復帰	SVC/USR/FIQ/ABT/UND 待避・復帰，FIQ 待避・クリア

ABT/UND のバンクレジスタを加えた 28 レジスタとなる（表 2(6)）．なお，FIQ モードのレジスタを破壊されると困る場合には，セキュアワールドからノンセキュアワールドへの遷移時には 8 レジスタの待避とゼロクリア，ノンセキュアワールドからセキュアワールドへの遷移時には 8 レジスタの復帰を行う必要がある（表 2(7)）．

3.5　高速化方式の改善

3.4 節で述べたアルゴリズムには含まれないが，セキュア，ノンセキュアワールド両方で，あるモードを使う場合においても，そのモードでの処理中に OS 切替えが起きないことが保証でき，かつそのモードから抜けた後にそのモードのバンクレジスタを破壊して良い場合，そのモードのバンクレジスタ待避は省略することができる．例えば，IRQ モードでのセキュアモニタコールやアボートを含む割り込みが禁止され，毎回 IRQ ハンドラの先頭でスタックポインタなどを含めたレジスタの初期化処理が含まれるような OS の場合を考える．このとき，IRQ モード処理中に割り込みが起きず，IRQ ハンドラを抜けた後では IRQ モードのバンクレジスタ破壊をしてもよいため，IRQ モードのバンクレジスタの待避処理は必要ない．このような場合，さらなる高速化が可能である．

4.　高速セキュアモニタ”LiSTEE モニタ”

本章では 3 章で述べた高速化機構を有する LiSTEE モニタの設計，実装について述べる．ソフトウェアの全体スタックおよび呼び出し関係を図 3 に示す．図 3 に示すように本研究では，LiSTEE モニタおよび LiSTEE モニタを呼び出すための Linux デバイスドライバを設計，実装した．

4.1　LiSTEE モニタの設計

LiSTEE モニタは (1) ワールド切替え機能，(2) メモリ保護設定機能，(3)OS ブート機能，(4) 例外ハンドル機能，の 4 つの機能を主に有する．なお，LiSTEE モニタにリンクする形でセキュアワールドのプログラムは動作する．

(1) ワールド切替え機能

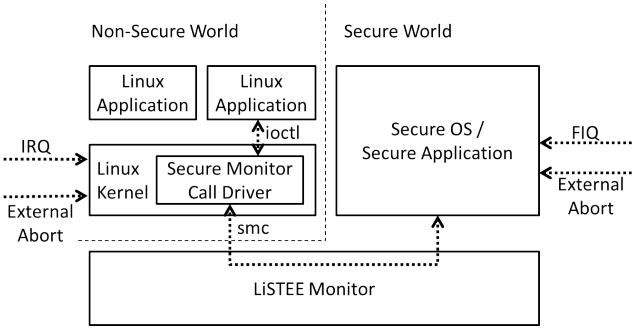

図 3　ソフトウェアスタックと呼び出し関係

ワールド切替え機能は LiSTEE モニタで最も核となる機能である．セキュアモニタコールもしくは後述の例外ハンドル機能からの呼び出しに基づき，ワールドの切替え処理を行う．まず遷移元のワールドを判定し，遷移先のワールドを判定する．遷移元のレジスタ待避後，ワールドを切替え，遷移先のレジスタを復帰する処理を行う．このレジスタ待避・復帰のコードには先述した高速化方式が実装されている．現在のワールドの設定，取得は SCR（Secure Configuration Register）の NS（Non-Secure） bit から設定・取得することができる．

既に述べたようにコンテキストスイッチ処理においてはできるだけ処理を短く終わらせることが望まれるため，不要な判定処理などは極力減らすことが好ましい．このため，あらかじめモニタのコンパイル時に設定された情報から，プリプロセッサが最低限のレジスタ待避・復帰・クリアするモニタコードを出力する．すなわち，実行時にはどのレジスタを待避させるかといった判断は行われない．例えば SVC レジスタのみ待避させる場合には，モニタのバイナリコードには SVC レジスタの待避・復帰コードのみが含まれ，実行時に設定を参照するような動作は行われないため，高速である．組込み機器においては動作する OS は機器出荷後に変わることはほとんど考えられないことに加え，もし出荷後に変わる場合にはファームウェアアップデートという形で，モニタコードも含めて更新することが可能であるため，高速化を重視した仕組みになっている．

(2) メモリ保護設定機能

LiSTEE モニタはデフォルトでは，モニタプログラムを含

むセキュアワールド用領域をノンセキュアワールドから読み書き不可に，ノンセキュアワールド用領域をノンセキュア・セキュアワールド双方から読み書き可能に設定する．モニタプログラムや OS 切替え時のコンテキスト待避領域についてもノンセキュアワールドから読み書き不可に設定されるため，セキュアに OS 切替えを実行することができる．なお，セキュアワールド側プログラムはメモリの一部領域をワールド間の通信用領域として，ノンセキュア側及びセキュア側で読み書き可能に設定することができる．例えば，セキュア側で暗号化処理を行う場合に，ノンセキュア側から共有メモリに平文データを入力して，セキュア側で暗号化後，共有メモリから暗号文データを取得するといったことができる．

(3) OS ブート機能

LiSTEE モニタが組み込まれた機器を起動すると，まず LiSTEE モニタが起動する．さらにそこから，セキュアワールド，ノンセキュアワールドで動作する OS 等のプログラムを起動する必要がある．このため LiSTEE モニタは各ワールドの初期化後，設定されたプログラムを起動する．デフォルト設定ではセキュアワールド側では LiSTEE に直接リンクされたセキュアワールド用プログラムを，ノンセキュアワールドではブートローダ u-boot を起動し，さらにノンセキュアワールドの u-boot は Linux を起動する．

(4) 例外ハンドル機能

TrustZone においては FIQ/IRQ/外部例外のそれぞれを OS に入力するか，モニタに入力するかを設定することができる．図 3 に示すように LiSTEE モニタでは初期設定としてこれらを全て OS に入力する設定になっている．モニタに例外が入力された場合，セキュア側の例外ベクタを呼び出す処理を行う．この際にワールド切替えが伴う場合，すなわちノンセキュアワールドにいる際に FIQ や外部例外が発生した場合には OS 切替え機能を呼び出す．これらの設定は，セキュア OS により設定を変更することができる．例えば FIQ を使わない場合には FIQ 割り込みを無効にすることや，外部例外をモニタに入力するように設定することも可能である．なお，割り込みコントローラの設定により，セキュアワールドでプログラムを実行中の場合のみに FIQ 入力を許す設定なども可能であり，FIQ をセキュア OS のみで扱うといったこともできる．

4.2 Linux 用デバイスドライバ

ノンセキュアワールドで動作する Linux のプログラムからセキュアワールドのプログラムを呼び出すためにはデバイスドライバが必要である．本研究で設計したデバイスドライバはデバイスドライバに対して ioctl 命令を発行するとセキュアモニタコール命令を実行し，モニタを呼び出す．また共有メモリをユーザプロセス空間にマップする機能や，共有メモリに対する read/write システムコールによる読み書きについても対応する．これにより，Linux のユーザプロセスからセキュアワールドで動作する処理を呼び出すことができる．本デバイスドライバはキャラクタデバイスとして動作する．

4.3 定義生成ツール

LiSTTE モニタはコンパイル時に待避が必要ないバンクレジスタの定義を行うことで，待避を省略するコードを生成する．LiSTEE モニタを使ったプラットホーム開発者が容易にこの定義を行えるようにするため，各 OS が扱うモード等を入力すると待避の必要がないバンクレジスタの定義を出力する定義生成ツールを作成した．この出力をモニタのコンパイル時に利用することができる．入力としてはファイルに OS のセキュアノンセキュアの別，扱うモード，扱うモードに関してはセキュリティ設定を記述し，入力する．すなわち，3.3 節における use_flag および security_flag を入力すると，type 相当の結果が出力される．

このうち，セキュリティ設定については省略可能である．セキュリティ設定を省略するとノンセキュア OS 側のセキュリティ設定は NONE，セキュア側のセキュリティ機能は SAVE_AND_CLEAR として結果を出力する．これは，セキュア側では悪意あるプログラムやバグが混入する可能性が低いため，あえてノンセキュア側のコンテキストを保護する意味が小さいためである．

4.4 実装

LiSTEE モニタの実装は TrustZone 機能が搭載された ARM 社の Cortex-A9 リファレンスボード Versatile Express（Coretile Express A9x4）に対して行った．CPU の動作クロックは 400MHz，メモリ動作クロックは 266MHz，メモリ容量は DRAM が 1GB，SRAM が 40MB となっている．また，ノンセキュア OS の Linux としては Linux 3.6 を利用している．ノンセキュアワールドでは，LiSTEE モニタによりまずはオンボード NORFLASH に格納された u-boot が起動され，その後 u-boot が SD カードから Linux を起動する仕組みとなっている．

なお，メモリ空間としては Linux に 256MB，LiSTEE モニタに 1MB，Secure ワールド用プログラムに残りを割り当てている．割り当て方法は固定ではなく，LiSTEE モニタ及び Secure 側動作用には DRAM もしくは SRAM を割り当てることもできる．LiSTEE モニタのコード自体は ARM アセンブラで 1000 行程度，利用するメモリは 64KB 以下と十分に小さい．これは十分に検証が可能なサイズであり，モニタ自体への脆弱性が混入しにくくなっている．

5. 評価

本章では，LiSTEE モニタの高速化効果を検証するため，ワールド間遷移にかかる往復時間の評価及び暗号化処理を

表 3　ワールド間遷移時間の評価結果

パターン	往復時間 [us]	高速化率 [倍]
(1) 既存手法	1.66	–
(2)SVC 待避・復帰	0.805	2.06
(3)SVC/USR 待避・復帰	0.809	2.05
(4)(3)+FIQ クリア	1.15	1.44
(5)SVC/USR/ABT/UND 待避・復帰	1.29	1.29
(6)(5)+FIQ クリア	1.47	1.13

セキュア OS 側で行うことを想定して，各コンテキスト待避のパターンで処理時間の計測を行った．以降では評価結果について述べる．

5.1　ワールド間遷移時間の評価

本稿で提案する高速化手法がワールド間遷移速度に与える影響を計測するため，いくつかのレジスタ待避のパターンでワールド切替えに要する時間を計測した．Linux ドライバからセキュアモニタコールを発行し，LiSTEE モニタによるワールド切替え後，セキュアワールドからもセキュアモニタコールを発行し，Linux ドライバに戻ってくるまでの往復時間を計測している．これを 500 万回繰り返した時間から，1 往復あたりの時間を計算した．

本評価では，ノンセキュアワールド側では FIQ モードのみ利用しない Linux を動かし，ワールド切替え時に (1) 全てのレジスタを待避・復帰させた場合（既存手法），(2) 単一機能をセキュアワールドで動かすことを想定して非バンクレジスタと SVC モードバンクレジスタのみを待避させた場合，(3) 割り込みを利用しない簡単な OS をセキュアワールドで動かすことを想定して USR モードのバンクレジスタについても待避させる場合もしくは FIQ による割り込みを利用するが FIQ モードのバンクレジスタ待避させない場合，(4)(3) でセキュリティを担保するために FIQ レジスタを待避させる場合，(5)ABT/UND モードも利用する OS を動かす場合，(6)(5) でセキュリティを担保するために FIQ レジスタを待避させる場合の 5 パターンで評価を行った．なお，モニタコードについては DRAM に配置して評価を行っている．測定結果を表 3 に示す．

評価結果より (1) に当てはまる場合，すなわち単一機能をセキュアワールドで動かす場合などに最大 2.06 倍の高速化効果（ワールド切替え時間の 51.5%削減効果）が得られることがわかった．(1) では待避するレジスタ数は 39 レジスタから 16 レジスタへと 59.0%削減できており，この値に近い値となっている．実際に TrustZone のワールド切替えにおいてレジスタ待避・復帰処理がボトルネックとなっており，レジスタ待避・復帰処理の高速化が性能向上に大きく寄与することがわかる．特に組込み用途では TrustZone による処理分離を暗号化処理，完全性検証処理など単一の用途に用いることも多いと考えられ，高速化の恩恵を得られる用途も多いといえる．

また，yield 命令などによる自発的なユーザプロセス切替えを伴う簡単な OS を載せる場合にも，FIQ による割り込みを利用しない場合には (3) の 2.05 倍の大きな高速化効果が得られる．加えて，タスクのプリエンプションに対応するような場合で，FIQ による割り込みを利用してさらにセキュリティを考慮する場合においても，(4) の 1.44 倍の高速化効果を得られることがわかった．(4) では ABT や UND モードの利用を想定していないが，組込み用途，特にセキュアワールドで動かす処理においては導入されるソフトウェアは限定的であることが多いと考えられ，テストをあらかじめきちんと行っておけば ABT モードや UND モードが必須となる場面は少ないはずである．

また，セキュアワールドで FIQ を扱い，セキュアワールドからノンセキュアワールドに切替える際に FIQ レジスタをクリアする (5) の場合にも性能は 1.13 倍となっており，ABT モードを利用したメモリ保護機能やコピーオンライト機能など大きな機能を持った OS をセキュアワールドで動かした場合にも一定の高速化効果が得られる．

なお，本評価は 400[MHz] で動作するプロセッサで行っており，(2) の 0.805[us] はサイクル数に換算すると 322 クロックであり，一般的な OS 内でのコンテキストスイッチのコストと比べても遜色なく，OS 切替えのコストとしては十分に小さいと言える．これは多くの用途で TrustZone による処理分離を利用できることを意味している．

5.2　暗号化処理による評価

実アプリケーションでの提案手法の効果を検証するため，セキュアワールドで暗号処理を動作させ，性能評価を行った．評価に用いた暗号化のアルゴリズムは，XOR によるマスク処理と AES（128 ビット，EBC モード）の 2 種類である．16byte 毎にノンセキュア OS から共有メモリを介して与えた平文に対して暗号化を行い共有メモリに出力する．ノンセキュアワールドでは Linux を動作させるとともに，セキュアワールドでは暗号化機能を SVC モードでのみ動作させ，全てのレジスタを待避させた場合と，高速化機構によりバンクレジスタは SVC モードのみを待避させた場合について評価を行った．また，比較としてセキュアワールドに遷移せず，Linux 側のみで暗号化を行った場合の評価も行った．結果を図 4 に示す．

評価結果より，高速化機構有りと無しの場合で XOR では 38%の高速化，AES では 9.2%の高速化効果が得られることがわかった．特に暗号化処理自体が軽量な XOR 処理においては大きな効果が得られていることがわかる．性能向上効果により，従来 OS 間遷移がボトルネックになっていて TrustZone による処理分離が行えなかったような処理においても TrustZone が適用可能になると考えられる．

一方で，特に XOR 処理においてはワールド間遷移自体

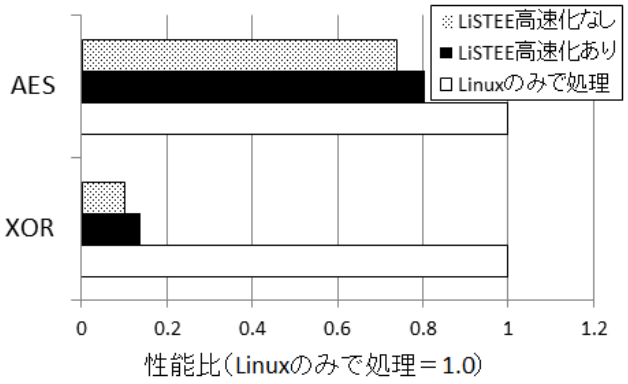

図 4 暗号化における高速化の評価

のオーバヘッドが暗号化処理に比べて大きく，性能が低下している．セキュア側で暗号処理を行うことによりノンセキュア側に対して，鍵漏洩などのリスクは低減できるため，セキュリティの強化効果と性能のトレードオフとなる．今回は AES のブロックサイズに合わせて一度に 16 バイト毎に処理をしているが，緩和策としてこの処理粒度を大きくするとワールド間遷移回数を少なくできるため性能低下は小さくなると考えられる．ただし，単純に処理粒度を大きくすればよいというわけではなく，処理粒度を大きくするとワールド間での通信に使うメモリサイズが増えるほか，セキュアワールドでの処理時間が長くなるため，ノンセキュアワールドの応答性が下がるという欠点もある．この場合にも，同じブロックサイズの場合においても本提案手法ではセキュリティや機能を犠牲にすることなく，応答性を上げることが可能である．

6. 関連研究

TrustZone 対応のモニタとして Sangorrin らによる SafeG[5] や，Open Virtualization[6] が存在する．また，x86 向け仮想化技術として VT-x[7] が存在する．

SafeG や Open Virtualization においては，OS 切替えの際に全てのバンクレジスタを待避・復帰させており，本稿で述べたような OS 切替えの高速化については実装されていない．特に組込み機器においてはプロセッサ能力が未だ x86 向けプロセッサに比べて貧弱なことが多く，本稿で提案したような高速化手法が有用と考えられる．また，SafeG においては，OS 切替え時にコンテキストを SRAM に待避させることで高速な OS 切替えを実現しているが，LiSTEE モニタにおいても同じく SRAM にコンテキストを待避する設定をすることもできる．

仮想化技術によるセキュリティ強化の試みとしては，x86 プロセッサを対象として多くの研究が行われている [8], [9]．特に x86 向け仮想化技術で近年多く利用される Intel 社の VT-x においてはコンテキスト待避復帰を高速化させる方法として，VMCS と呼ばれる領域へのシステムレジスタ待避を自動化することが可能である．一方，VT-x では汎用レジスタの自動的な待避は行われないため，ソフトウェア的に待避させる必要がある．x86 プロセッサではモード別のバンクレジスタがないため，本技術をそのまま適用するすることはできないが，利用しない汎用レジスタの待避を省略する技術を併用する考え方自体は適用可能で，OS 切替えの高速化が可能である．

また Linux 自体のセキュリティ強化を行う試みとして，強制アクセス制御手法の導入 [2][3]，アドレス配置をランダム化して侵入攻撃を成立しにくくする手法である ASLR（Address space layout randomization）対応，データ領域を実行不可能にする NX ビット（No eXecute bit）対応などがある．これらは根本的には侵入不可能な Linux を作ることは不可能であるため，本提案のようなセキュアモニタとの併用が望ましい．

7. おわりに

本稿では ARM TrustZone 機能を利用した OS の高速な切替え機能を有するセキュアモニタ LiSTEE モニタの設計と実装および評価について述べた．評価より本手法を適用しない場合との比較よりデータ暗号化処理において，最大 38%の高速化効果を確認した．今後の課題として LiSTEE モニタのマルチコアプロセッサ対応，Cortex-A15 向け仮想化技術への対応などがある．

参考文献

[1] MITRE: Common Vulnerabilities and Exposures (CVE), http://cve.mitre.org/
[2] P. Loscocco et al.: Integrating Flexible Support for Security Policies into the Linux Operating System, Proc of the FREENIX Track: 2001 USENIX Annual Technical Conference (FREENIX'01), pp 29-42 (2001).
[3] 原田ら: TOMOYO Linux - タスク構造体の拡張によるセキュリティ強化 Linux, Linux Conference 2004 (2004).
[4] ARM: TrustZone, http://www.arm.com/ja/products/processors/technologies/trustzone.php
[5] D. Sangorrin et al.: Dual Operating System Architecture for Real-Time Embedded Systems, Proc of the 6th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT2010), pp. 6–15 (2010).
[6] Open Virtualization: http://www.openvirtualization.org/
[7] Intel:Intel Virtualization Technology, http://www.intel.co.jp/content/www/jp/ja/virtualization/virtualization-technology/hardware-assist-virtualization-technology.html
[8] A.Seshadri et al.: SecVisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity OSes, Proc of 21st ACM SIGOPS symposium on Operating systems principles(SOSP '07), pp. 335–350(2007).
[9] 品川ら: 準パススルー型仮想マシンモニタ BitVisor の設計と実装, 2008 年並列／分散／協調処理に関する『佐賀』サマー・ワークショップ（SWoPP 佐賀 2008）(2008).