メインメモリの機密性ならびに完全性保護技術の研究

２０２２年　３月

橋本　幹生

メインメモリの機密性ならびに完全性保護技術の研究

橋本　幹生

理工情報生命学術院

システム情報工学研究群

筑波大学

２０２２年　３月

要旨

本研究ではコンピュータシステムの水平分業化を背景として生じた，データならびにプログラムからなる処理対象の情報オーナーと，実行を行うシステムの管理権限者のと間の不一致問題に関連する，機密保護と完全性保護に関するシステムセキュリティ研究について述べる．

序論では近年のクラウドコンピューティング導入を追って普及が進むコンフィデンシャルコンピューティング技術の背景として，全般的なサイバー攻撃脅威の深刻化に加えて，コンピュータシステム技術全般の水平分業化という組織に関わる要素も大きいことを示した．水平分業化により，従来は組織内で管理されていた攻撃経路が外部組織の管理下に置かれた結果として，コンピュータシステムのメモリにおいて処理される重要データのオーナーとシステムの管理権限に不一致が生じ，新しい脅威が生じたことがコンフィデンシャルコンピューティング導入の大きな要因である．そして，個人所有かつオープンソース OS が稼働する PC にコンテンツプロバイダの重要データであるデジタルコンテンツが配信されるケースや，社会インフラ機器の物理メンテナンスがアウトソーシングされるケースについても，メインメモリに格納されたデータのオーナーとシステムに対する論理的物理的管理者の間の不一致による脅威にさらされるものと解釈できる．メインメモリセキュリティに関する機密性保護技術は前者のデジタルコンテンツ保護，完全性保護技術は後者の社会インフラ機器の保護に関する脅威を背景として取組んだ研究である．

機密性保護技術では, デジタルコンテンツ保護をターゲットとして, オープンソース OS による資源管理の制約の下でプロセスを構成する 3 要素の全てに対する暗号化と整合的なプロセス保護モデルを提案した．暗号化に関わる管理を OS とは独立したプロセッサハードウェアとして実装することにより，エンドユーザがオープンソース OS を自由に改造可能な環境においても，プロセスの保護に対する干渉を防止できる．同機能の FPGA 上ハードウェア実装ならびに対応 OS の実装を行い，プロセス 3 要素が全て暗号化された状態におけるプロセス実行と既存 OS と整合的な提案プロセス保護モデルの妥当性を確認し，メインメモリ暗号化における回路規模のオーバヘッドをはじめて実装に基づき定量的に評価した．

また，提案方式のプロセッサにおける保護プロセス実行を，外来プログラムをターゲットシステムで実行する依頼計算としてとらえたモデルならびに依頼計算内容の保護とプライバシ保護の両立を可能とする管理モデルを提案した．

完全性保護技術では，社会インフラ機器の物理メンテナンス権限が悪用される脅威，中でも不揮発メモリの完全性を担保する技術提案と試作評価を行った．提案方式は従来のハードウェアキャッシュと連携したキャッシュライン単位の改ざん検出用のツリー検証処理を，汎用 CPU の仮想化機構を利用したページ単位の検証処理と CPU 内蔵メモリに対するページング処理により実現するものである．検証ツリーを拡張ページテーブル（EPT） のトポロジーと合致させることにより，特殊なハードウェアを用いることなく未検証メモリの参照の検出が可能となる．また，ページ検証を適用するゲスト OS とページ検証なしのゲスト OS が安全に共存可能な拡張ページテーブル管理方法も提案し，定量評価を行った．Linux ブートにおいては，内蔵ページバッファ 4MiB の環境において，暗号連携 DMAC を利用すればページ検証なしの場合と比較して 3.8 倍の時間を要するものの，変更なしのゲスト OS に対してメモリ参照時の完全性検証が実現できることを実証した．

また，本研究公表後の研究動向および商用プロセッサの技術動向をまとめた．メインメモリ暗号化機能は，2020 年以降に新規発表されたほぼ全てのサーバ向け CPU アーキテクチャに導入されており，PC 向けプロセッサにおいても普及が進みつつある．

本研究はデジタル著作権保護ならびに社会インフラ機器に対する物理攻撃対策をターゲットとした研究だが，情報オーナーと管理権限の不一致としてとらえた場合，近年クラウドにおいて導入が進むコンフィデンシャルコンピューティングと共通の課題に対する取組みと考えることができる．特にメインメモリ暗号化については，本研究の提案方式と共通の枠組みが現在の商用プロセッサに用いられている．また，メモリ完全性については，現時点でも解決されていない課題であり，提案手法が今後活用される可能性がある．

内容

図目次

表目次

1. 序論

1.1. はじめに

コンピュータシステムに対するサイバー攻撃は，攻撃の種類と規模の両面で拡大が続いている．政府や企業，個人の攻撃対策負担も深刻化しており，より網羅的かつ効率的なサイバー攻撃対策が求められている．米 FBI がまとめた米国内のサイバー犯罪被害申告は，2016 年の 16 億ドルから 2020 年には 42 億ドルに増加した [1]．社会インフラの制御システムに対する全世界被害は 2024 年には 500 億ドルに達すると Gartner 社は予測している [2]. また，海賊版被害など知的財産権や機会損失を含む全世界の損害額は 2013 年の 3,000 億ドルから，2020 年には 9,450 億ドルに達したとの報告もある [3]．ランサムウェア攻撃グループが脅迫により得た収入は，1 グループだけを見ても 2020 年に少なくとも 1.23 億ドルに上ると推計されており，ランサムウェアがビジネスとして成立しつつあることを物語る [4]．攻撃対策の費用は全世界で 2025 年には総額 2,250 億ドルに達すると予測されている [5].

コンピュータシステムのセキュリティ構成要素には，大きく分類して通信に適用される暗号プロトコルとシステム内部の保護メカニズム（以下（**内部）アクセス制御**）の２種類がある．インターネットのセキュリティは主としてエンドツーエンドで実行される TLS（Transport Layer Security）などの暗号プロトコルにより担保されている [6]．暗号プロトコルではユーザトラフィックの隔離を担保するのは異なる暗号鍵による暗号化である．暗号化による隔離は通信経路の管理に依存しないため，一元的な管理を前提としないインターネットの仕組みとの親和性が高い [7, 8].

一方，コンピュータ内のアクセス制御は，OS または VMM(Virtual Machine Monitor = 仮想マシンモニタ)とハードウェアそれぞれの連携により，ユーザプロセス間またはゲスト仮想マシン間を論理的に隔離する手法が一般的である [9]．従来ユーザプロセスまたは仮想マシン間の隔離には暗号は導入されていなかった.

だが，近年コンピュータシステム内部の処理過程データについて，準同型暗号に代表される数学的秘匿計算と，メインメモリ暗号化に代表されるアーキテクチャレベルの保護メカニズムからなるコンフィデンシャルコンピューティングと呼ばれる新しいコンセプトの導入が進んでいる [10]. 両者はいずれも暗号技術に基盤を置く保護であるが，前者は暗号方式に閉じるものであるの

に対して，後者は暗号技術とハードウェアアクセス制御や OS/VMM レベルの隔離機構を連携させる点に違いがあり，メモリ上データの保護につき相補的に機能する．このコンセプトに沿って分散コンピューティングのセキュリティ強化やセンシティブデータを安全にクラウドシステムで処理する研究が進められている [11]．また，産業分野においても 2019 年には Linux コンソーシアムにより同コンセプトの普及団体コンフィデンシャルコンピューティングコンソーシアムが設立され，この問題に関する啓蒙活動が行われている [10]．コンフィデンシャルコンピューティングにおいて新たな保護対象と定義される “**Data In Use**”と既存の保護対象の分類と，本研究が扱うメモリセキュリティとの関係を図 1 に示す．コンフィデンシャルコンピューティングの導入によりこれまでネットワーク(“Data In Transit”)，ストレージ(“Data At Rest”)にとどまっていた暗号による保護がメインメモリを含む処理中データにまで拡張され，セキュリティ対策の網羅性が向上する．

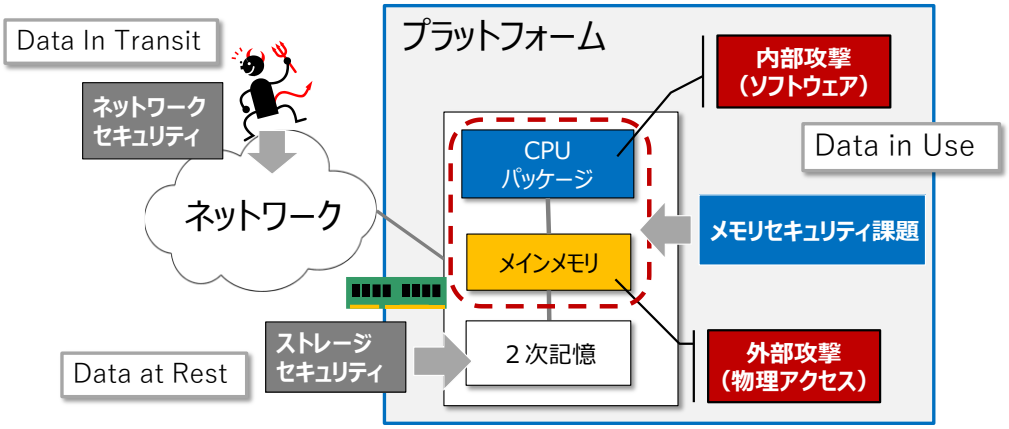


図 1　コンフィデンシャルコンピューティングとメモリセキュリティの関係

本研究ではコンフィデンシャルコンピューティングの構成要素のうち，後者に属するメインメモリ暗号化ならびにメインメモリ改ざん検証に関連する，先駆的な技術提案を述べる．以下序論においては，コンピュータシステムのセキュリティ機構と，関連する組織内アクセス権管理とネットワーク利用の発展をたどり，システム内部のセキュリティ機構にも暗号が導入された背景を説明する（以下改ざん検証を含めて**メインメモリ暗号化**と呼ぶ）．

1.2.　コンピュータシステムのセキュリティ機構の歴史

初期のコンピュータは技術計算支援など単独目的に用いられ，単体で動作しコンピュータ間のデータ交換は主としてテープによって行われた [12, 13]．1950 年代から SAGE [14], Sabre [15] など，空軍迎撃システム，民間航空用の専用システムとして単一のホストコンピ

ュータシステムに 1,000 台以上の専用端末が電話回線を通じたモデム経由で接続されるシステムが構築され，成功をおさめたが，当時のコンピュータの内部保護機能はごく限られたものであった [16]．内部アクセス制御には，あるユーザのバグが他のユーザに影響を及ぼさない頑健性 （Robustness）と情報保護のセキュリティ目的の２つの目的があり，初期のシステムでは頑健性を担保する側面が強かった [9].

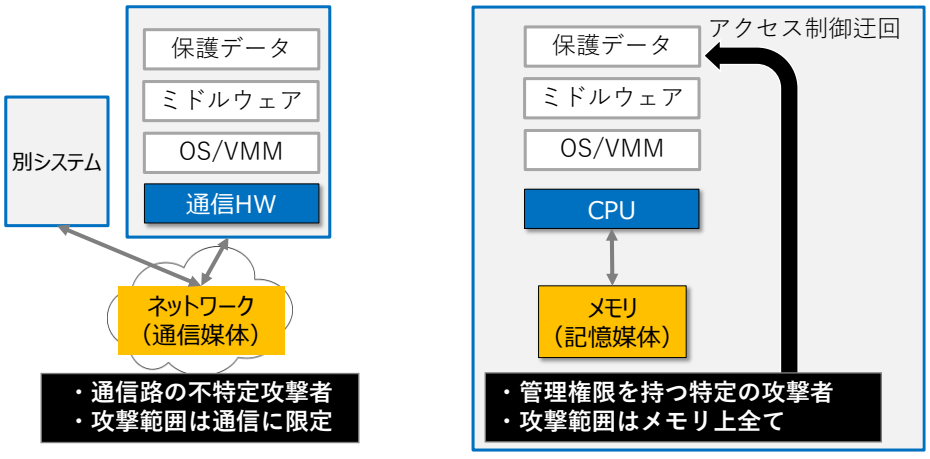
1960 年代の TSS (Time Sharing System)導入後，ユーザ間の隔離や階層ファイルシステムなどの内部アクセス制御が整備され，マルチユーザによるターミナルベースの遠隔利用は広く普及したが [17, 18]，ネットワークによるコンピュータ間のデータ交換が本格化するのは 1970 年代以降である．コンピュータ間のデータ交換が未発達の 1960 年代後半から，単一マシン内の情報管理を目的としたシステム内部のアクセス制御高度化の取り組みが始まっている [19]．1974 年には MLS (Multilevel System Security) の理論的枠組みが発表されている [20]．これに先立って 1969 年には IBM S/360 における MLS の実装報告が行われ [21]，その後主として軍事用に組織階層と整合したデータの一方向伝搬を担保するメッセージングシステムが実用化された [22, 23]．これは今日の SELinux につながっている [19]．MLS については再度触れるが，代表的な応用は，組織の重要機密に対して組織トップである将軍や社長はアクセスできるが，コンピュータ管理者はアクセスできない性質，すなわち組織の機密アクセス管理の階層化とコンピュータシステムの管理の分離が挙げられる．

このように端末とホスト間の通信接続は 1950 年代後半から実用化され，コンピュータシステム内部のアクセス制御も 1960 年代後半にはマルチユーザや階層ファイル管理など今日の OS 機能の基盤機能が実用化された．その後，通信路データに対する暗号ベースの技術的保護が一般に利用可能となったのは，インターネットの商用解放後の 1995 年以降である．Netscape 社により提案された SSL(Secure Socket Layer)プロトコル [24]が PC 用 Web ブラウザに組込まれたことがその契機となった．その後公衆インターネット上トラフィックの暗号化は，2017 年には全体の半数以上を占めるまで普及した [25].

一方，コンピュータの内部アクセス制御にメインメモリ暗号化が適用されるようになったのは，2014 年の Intel SGX 商用化以降である [26]．内部アクセス制御は 1970 年代に高度化しており，セキュリティ要求は存在していたにも関わらず，通信よりも暗号化の導入は遅れた．以下，その理由について，組織のデータ保護管理とデータ処理が行われるコンピュータ管理の視点で考察する．

1.3. メインメモリ暗号化要求の背景

メインメモリ暗号化要求を議論する準備として，通信路とシステム内部それぞれ管理の特性に起因する脅威源と脅威の範囲の概略を図 2 に示す．図 2 (a)に示すインターネット通信路の場合，通信路は一元管理されていないため，脅威として不特定多数の攻撃者を想定する必要がある．暗号化による通信チャネルの隔離は，攻撃者と正規の利用者を識別する手段として機能する．インターネットでは最悪の場合，通信路データの全てが完全に攻撃者の監視下におかれている可能性があるが，利用者毎の通信路暗号化により，攻撃者が介入できるのは利用者毎の暗号管理に不備があった場合に限定される．仮に一部の遠隔利用者によるなりすましが成功したとしても，一般に遠隔利用者はシステムの全管理権を持っているわけではないため，重要データに対して脅威が及ぶ範囲は限定される．これを担保するのが OS ならびにミドルウェアによる内部アクセス制御である．一方，図 2 (b)に示すシステム内部は一般に単一の管理者に帰着する階層的な管理が行われる．通信路とは異なり，システム内部の攻撃者は管理者を含む正規の利用者に限定される．一般の利用者がアクセス可能な範囲は OS ならびにミドルウェアによる内部アクセス制御により制限される（アクセス制御の視点では脆弱性を利用した外部からの攻撃は正規利用者による不正の一部とみなす）．通信路とは異なり，他の利用者のデータに対する参照，操作は内部アクセス制御により禁止されている．ただし，管理者が不正を行う場合，メモリ暗号化なしの既存システムではOSやミドルウェアによるアクセス制御を迂回して重要データを参照することができてしまう．従来のシステムでは大きな問題とみなされていなかったこの脅威がメモリ暗号化要求の背景である．



(a) 通信路に対する攻撃　　(b) メインメモリに対する攻撃

図 2 通信路攻撃とメインメモリ攻撃の概念図

この迂回経路は複数存在する．コンピュータシステムが組織内で管理されているとき，これらの経路は基本的には全てが保護データのオーナーである組織により管理されていた．だが，オープンソース OS や物理メンテナンスのアウトソースを含む広い意味での水平分業化の進展により [27, 28]，迂回経路の管理権限と保護データオーナーの間に不一致が生じるケースが増えている．クラウドシステムは，主要な迂回経路の管理がクラウドプロバイダに集中することから，管理権限の不正利用の懸念が顕著となる例である．しかしながら，迂回経路の一部にのみ不一致がある場合，具体的には個人 PC におけるオープンソース OS の導入や，インフラ機器の物理的管理がアウトソースされるケースにおいても，この脅威は存在する．本研究はクラウド以外において，データオーナーとシステム管理者の不一致が生じる，デジタル著作権保護ならびに社会インフラシステムの保護をターゲットとした２つの技術提案である．

クラウド化以前は組織の重要データは組織内の運用によって保護されていたが，クラウド化によりその前提が成り立たなくなったことを具体的に説明する．一般に，組織には基幹となるデータがある．人事や給与などの組織運用に関わるデータや公共サービスの住民台帳，金融システム，医薬用データベースなどさまざまなものが考えられるが，ここではこれらの重要データを抽象化して組織の**バルクデータ**と呼ぶ．バルクデータの漏洩・改ざんは組織の存続に関わる重大リスクである．上述のように 1960 年代から遠隔ユーザがターミナル経由でメインフレーム上の組織データに遠隔アクセスする運用は行われており，アクセス制御も導入されていた．

図 4 はクラウド以前の組織内システムと遠隔アクセスに関連する脅威を抽象化して表現したものである．脅威源をオレンジ色の項目で示す．右側はシステムに対するネットワークアクセスである．正規ユーザによる外部アクセスについて，通信路の脅威は通信路暗号化により対策されている．また，遠隔アクセスについては，組織内のメンバシップ管理により，利用者が限定されている．正規利用者の中に存在すると考えられる内部不正者は，上述のように OS およびアプリケーションレベルの対策がとられており，バルクデータに対する直接アクセスは対策されている．組織内システムが外部サービスを利用する場合に，同じサービスを受ける別ユーザによる攻撃も考えられるが，これは外部サービス提供組織の内部隔離により対策されている．外部システムとの関係は契約ベースであり，内部統制は及ばないが，外部サービスの利用はアプリケーションによって行われるため，仮に外部サービス経由の攻撃が行われても OS およびアプリケーションレベルのアクセス制御によりバルクデータへのアクセス対策がとられている．

左側にはシステム構築ならびに運用時の脅威源について，それらの作業者を抽象化して(1)-(4)の**アクター**として示している．(1)内部オペレータや(2)ハードウェア管理者によるバルクデータアクセスについては，技術的な対策は施されていないものの，メンバとオペレーションは組織内部で統制される前提を置くことができる．システムの構成要素である(3)OS と(4)ハードウェアについては，他組織にも提供される汎用インフラであるため，不正に対するけん制力が働く．この脅威については導入(の第３者セキュリティ評価により不正を排除する手段も使える．委託開発による(0)専用ソフトは，内部統制の範囲外であり，他組織と共有されないことから不正に対するけん制力は期待できないが，OS レベルの動作ログは，不正動作に対する技術的抑制手段となる．このように，システムが組織内で管理される場合には，バルクデータに直接アクセス可能なアクター(1)-(4)の脅威源に対して，技術的対策に代わって組織内の運用管理が適用されていた．

先のシステムをクラウド上にホスティングした場合の概念図を図 3 に示す．クラウドの場合は，右側に示す基幹システムに対するアクセスは全てリモートアクセスとなるが，遠隔利用における脅威源とその影響は組織内管理の場合と基本的に変わらない（単純化のため外部サービスの図示は省略した）．クラウドでは別組織のシステムも同一の物理マシン上にホスティングされる可能性があるが， VMM の保護機構によって隔離されており，仮に別組織のシステムが不正を試みた場合にも技術的対策が存在する．この点が，通信経路が一元的に管理されていないインターネットと比較してクラウドホスティングとの違いであり，アクセス制御と暗号対策の連携が最近まで顕在化しなかった理由であると考えられる．

だが技術的対策の基盤となる VMM は，(1)VMM オペレータと(2)ソフトインフラ開発に依存しており，これらはクラウドプロバイダの管理下にある．ゲスト OS に対して VMM は等価的であるため，別途提供されない限りゲストである基幹システムの側が適切なオペレーションが行われているかを判断する材料はない．また，多くのクラウドデータセンタは厳格な入退出管理を行っているが，(3)ハードウェアメンテナンスについても懸念は存在する．これらを脅威として(T1)-(T3)と表記する．(T1)-(T3)いずれの脅威源もゲスト OS やミドルウェアのアクセス制御を迂回してバルクデータに対する直接アクセスの潜在能力があるが，ホスティングサービスを受ける組織の側からは技術的対策をとれない．通常クラウドプロバイダは契約上の守秘義務を負っているが，上述の通りクラウドプロバイダ内部者による不正についてクラウドプロバイダ自身もその懸念を表明している [29].

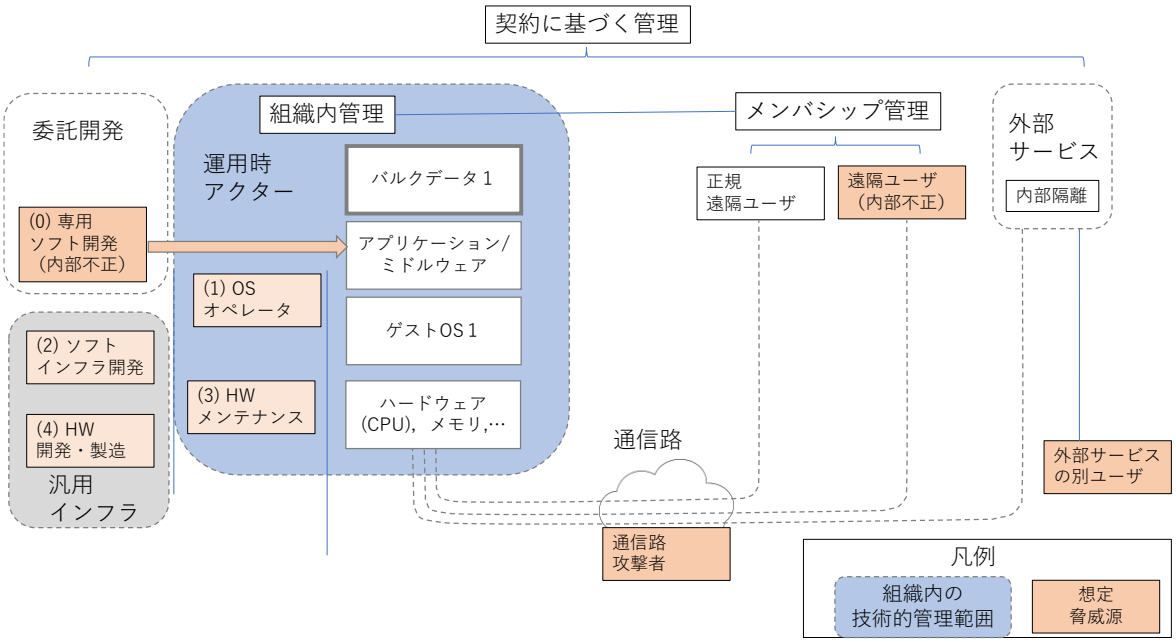


図 4　組織内で管理されるシステムに対する脅威

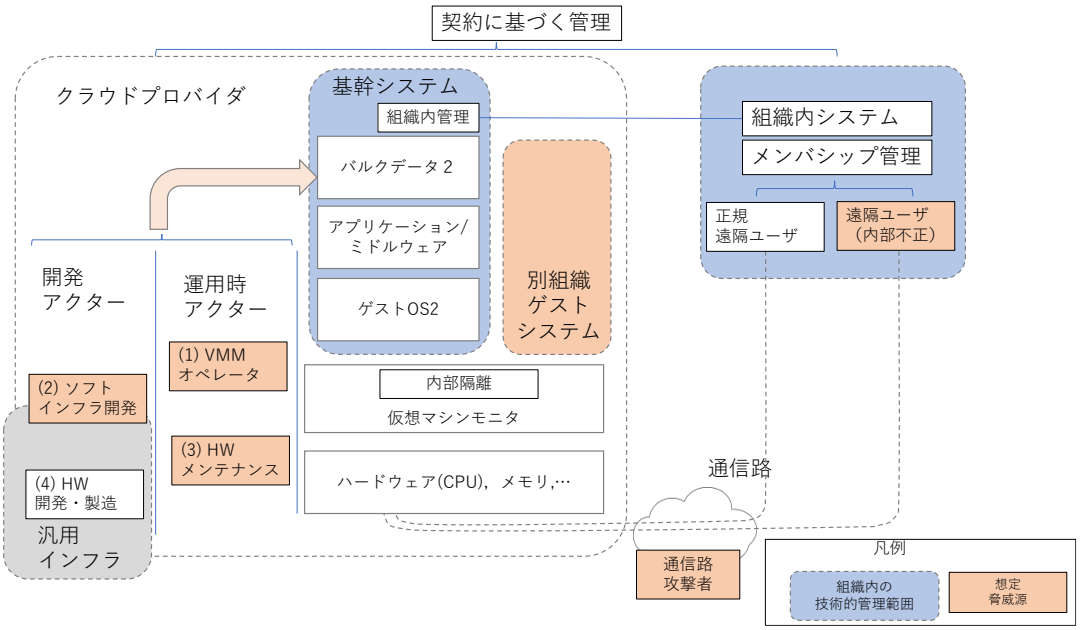


図 3 ホスティングされたシステムに対する脅威

このように，組織内システムをクラウドにホスティングする場合，バルクデータに対して組織が直接技術的対策を担保できない攻撃の迂回路が発生する．文献 [11]には，この脅威を含むクラウドセキュリティに関するいくつかの懸念が示されている．遠隔からの攻撃に対してはクラウドも内部システムも差は存在しないと考えられるため，この迂回路に対する技術的対策要求が，メインメモリ暗号化に代表されるコンフィデンシャルコンピューティング技術が導入された背景にあると考えられる．詳細は後述するが，アクセス制御と連携したメインメモリ暗号化は，前の段落で挙げた(T1)-(T3)の脅威に対する技術敵対策の基盤を提供できる．より抽象的な表現をすれば(T1)-(T3)の脅威は，システムに対する管理権限について 3 種類の側面を表現したものである．3 種類の権限(P1)-(P3)を表 1 に示す．この課題はより一般的に，システムにおいて処理対象となるデータ（バルクデータ）のオーナーと，システムの管理権限を持つ者との間に不一致が起こる場合の問題と言い換えることができる．

ここで説明した問題はクラウドシステムがオンプレミスと呼ばれる組織内システムと比較して，サイバー攻撃に対して脆弱であることを主張するものではない．むしろ，オープンソースインフラの脆弱性の迅速な対策においては，多数のユーザが共有するクラウドシステムのほうが有利である．外部からのサイバー攻撃抵抗性とは別に，クラウドシステムの運用における内部不正者や，クラウドインフラの脆弱性に対して，クラウドの利用者側では直接コントロールできない攻撃経路が存在し，その対策は一般のサイバー攻撃対策とは別途検討される必要があることがここでの問題である．

このようなクラウドホスティングは，見方を変えれば，ホスティング対象のゲストシステムが，バルクデータとともに移動して遠隔実行されるモバイルコードの一種として解釈することもできる [30]．モバイルコードにおいては，コードが実行される移動先ホストが受ける悪影響を排除する研究が行われている [31, 32]．一方，移動先のホストがモバイルコードまたはそのデータに与える脅威対策はモバイルコードの分野では少ない [31, 33]．Brooks はこの問題について，スマートカードなどの tamper proof device の応用を挙げているが [31],

表 1　管理権限の分類

番号	権限
(P1)	VMM/OS オペレータによるシステム管理操作
(P2)	VMM/OS 特権プログラムの変更操作
(P3)	メモリ等 HW に対する直接操作

スマートカードの処理能力は極めて限定されている．コンフィデンシャルコンピューティング技術はこの課題にも貢献できる可能性がある．

ここまで述べたように，メモリ暗号化は，上記のクラウドコンピューティングやモバイルコンピューティングにおいて顕在化した攻撃経路の技術的対策手段を提供するものである．メインメモリ暗号化の位置づけを明確化するため，２つの論点につき確認する．システム管理者による不正を防止するメカニズムとして，上述の MLS があった．実装としても必須アクセス制御機能を持つ SELinux [24]が存在する．脅威対策の有効性について，必須アクセス制御技術とメモリ暗号化技術の間の差異を確認する必要がある．また，データオーナーとデータが処理されるシステム管理者との間に不一致が存在するケースが，クラウドホスティングサービスやモバイルコード以外にも存在するとすれば，メインメモリ暗号化技術の枠組みはそれらを含むより一般的なものと定義できるはずである．

前者の問題について，MLS の有効性は(T1)-(T3)の脅威のうち，(T1)に限定される．SELinux が備えるような強制アクセス制御は，管理者権限を持つユーザによる不正操作をチェックするソフトウェア機構である [34]．管理者権限はいわゆる root 権限を意味しており，たとえ管理者権限をもったオペレータが発行したリクエストであってもポリシーに反した処理はチェックにより拒否される．だが，このチェック自体は OS 内部のコードによって処理されるため，文献 [35]に示されているように OS が変更された場合，チェックは迂回される．メモリの HW アクセスによる情報の流出・改ざん対策としても強制アクセス制御は機能しない．

後者の問題について，システムにおいて処理対象となるデータ（バルクデータ）のオーナーと，システムの管理権限を持つ者との間に不一致が存在する事例はクラウドシステム以外にも存在する．ひとつの例はパーソナルコンピュータにおける商用デジタルコンテンツ利用である．商用デジタルコンテンツは，サブスクリプションも含めてユーザがコンテンツ権利者に対価を支払い，再生利用権を得てコンテンツを楽しむ形態が一般的である [30]．デジタル化されたコンテンツは複製による海賊版作成を防止するために配信や機器間のデジタル接続に暗号化対策が行われている [36, 37]．だが，PC 内部の処理に着目すると，配布された暗号化コンテンツについて，再生利用権の確認後に暗号化コンテンツの復号が行われる．このとき，コンピュータのメモリには復号された平文コンテンツとその復号鍵が一時的に存在する．

個人利用の PC はユーザの所有物である．Linux などのオープンソース OS ベースのシステムユーザ自身が管理者権限を持つケースも多くメモリ内容を参照するツールも利用できる．だが，商用デジタルコンテンツの情報オーナーはコンテンツ権利者であるため，システムの

管理権限とそこで処理される情報オーナーの不一致が存在する．このため，メモリ上データに対するアクセスはデジタルコンテンツデータに対する攻撃界面となりうる．実際にリバースエンジニアリングによりデジタルコンテンツの復号鍵が暴露された例が知られている[38]．ゲーム機においてはメモリバスをタッピングするハッキングも行われている [39].

もうひとつの例として，物理的な社会インフラについても管理権限とそこで処理される情報オーナーの不一致が存在する．IoT 端末などフィールド機器に対して物理装置の維持管理を行うサービスマンは装置の物理的メンテンナンスを行うが，機器が処理するデータのアクセス権は持たないことが多い．ここにも物理アクセスの権限と想定されるデータアクセス権に不一致がある．以下，提案に関する共通の枠組みについてより詳細に述べる．

1.4. メモリセキュリティにおける脅威と基本的対策手段

本研究が扱う脅威と対策手段、そして保護するセキュリティプロパティの分類について概略を説明する。本研究が想定する脅威と攻撃界面を図 5 に示す．想定脅威は汎用メモリに対する物理アクセスによる攻撃（**外部攻撃**）と，CPU 内部で実行されるソフトウェアによる CPU 内部の情報資産に対する攻撃（**内部攻撃**）の 2 種類に大別される。外部攻撃は特殊な保護機能を持たない汎用メモリの外部端子あるいはバスを利用した CPU 稼働時ならびに CPU 非稼働時の読み出しと改ざんである．CPU 内部に保持された情報は、外部攻撃からは安全であると仮定し，CPU パッケージを信頼境界となる．ただし、CPU 内部の情報は無条件で安全ではなく．CPU 内部で実行されるソフトウェアの一部は，内部保護資産に対する読み出しと改ざんを行う内部攻撃の脅威源であると仮定する．また，本稿ではチップ開封による攻撃ならびに電力解析などのサイドチャネル攻撃は対象外であり，「物理攻撃」には含めない．

保護対象のプログラム実行時にはメモリバスを経由して保護資産であるデータはメモリ上と CPU 内を移動する．2 種類の脅威に対して、外部メモリ上と CPU 内のデータをシームレスに保護する機能を、ハードウェア単体もしくはハードウェアと信頼可能なソフトウェアの協調による CPU 内保護メカニズムにより達成することが本研究の課題である．

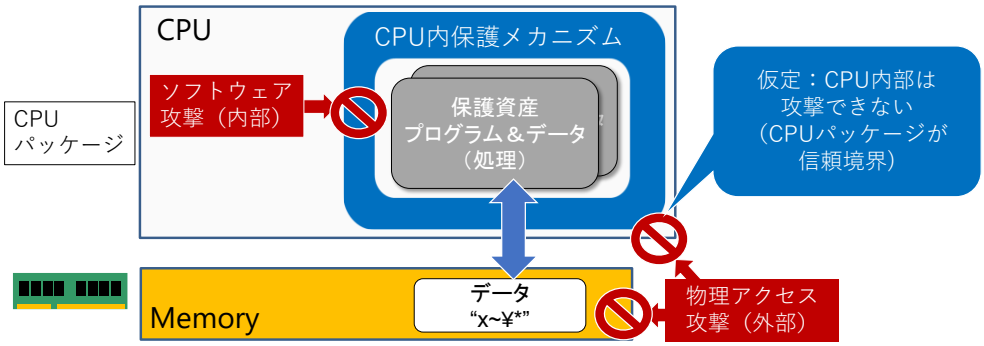


図 5　本研究の想定脅威と攻撃界面

CPU 内の情報は内部アクセス制御により保護されるが，メモリ上の情報は暗号的保護手段により保護される．情報セキュリティにおける保護の特性（セキュリティプロパティ）は大きく**機密性（Confidentiality)**，**完全性（Integrity)**，**可用性（Availability)**の 3 種類のカテゴリに分類される [40]. 本研究が保護対象とする機密性と完全性の保護手段の概要を図 6 に示す．機密性は保護対象の情報が攻撃を通じて第三者に漏洩しない性質であり，メモリ上データを暗号化された状態で保持することで保護が達成される．処理の実行に伴う外部データ参照において，読み込み時に復号，書き込み時に暗号化が行われる．一般的な共通鍵ブロック暗号においては暗号化を行ってもデータ長は変わらないため，アクセス単位と暗号ブロックのサイズが整合している限りメモリデータの配置とアクセスパターンは暗号化なしの場合と共通化できる．

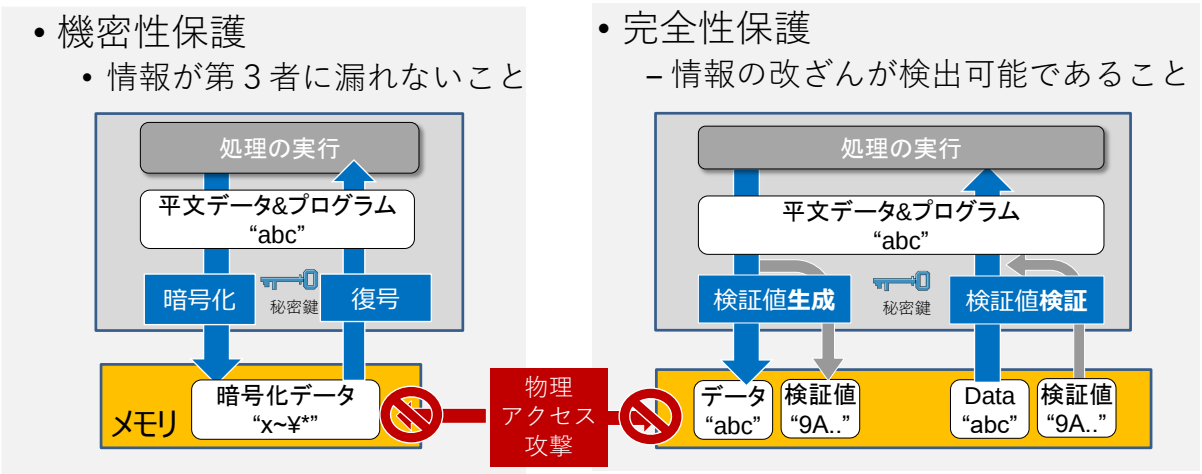


図 6　機密性と完全性の保護におけるデータ処理の違い

一方、完全性は保護対象の情報が攻撃を通じて改変されない性質であり，メモリ上データに対する暗号的な検証処理を行うことで保護が達成される．メモリ上のデータが物理的に改ざんされること自体は避けられない場合に、改ざんがあったことを確実に検出することで以後の処理に改ざんの悪影響が及ばないよう停止などの処置をとることが情報セキュリティにおける典型的な改ざん対策である．メモリデータの書き込み時に秘密鍵を用いてデータの検証値を作成し、メモリデータの参照時にこの検証値が参照データから計算される期待値と一致することをもって改ざんの有無を判定する **MAC** (Message Authentication Code)が代表的な実現手段である(上述の強制アクセス制御も MAC と表記されることがあるが，本稿では MAC は暗号手段について用いる)．AES や SHA など現在一般的なブロック暗号系の共通鍵暗号プリミティブ HW を用いることにより，機密性保護と完全性保護は後述のようにメインメモリアクセスに適用可能なスループット，レイテンシでの処理が可能である．機密性保護と完全保護の大きな相違点は，完全性保護では検証値の分だけ参照データが増えるため、その管理とメモリトラフィックにおけるインパクトである。三種類のプロパティのうち，可用性保護については本研究ではスコープ外とする．理由については後述する．

本研究では攻撃者が物理的に装置内部のメインメモリにアクセスしてプログラムあるいはデータの取得・改変を行う攻撃を「**物理攻撃**」と呼ぶ．本研究ではチップ開封による攻撃ならびに電力解析などのサイドチャネル攻撃は対象外であり,「物理攻撃」には含めない. また，攻撃者がソフトウェア的な操作により OS または CPU の特権モード権限を利用して, メインメモリ上のプログラムあるいはデータの取得・改変を行う攻撃を「(狭義の) **管理特権を利用した攻撃**」と呼ぶ.

なお,「管理特権を利用した攻撃」において，攻撃者は必ずしも正規の管理者とは限らず，上記のユーザ管理機器において特権を取得するケースや，ソフトウェア脆弱性を利用して特権昇格することで特権を取得するケースも含まれる．本来管理者特権は，不正者が取得できないように管理されることが望ましいが，現在特権機能を持つ OS を内蔵した機器がエンドユーザ向けに広く提供されていること，脆弱性による特権昇格があり，攻撃者が特権を取得した場合に備えた多層防御が必要とされる．メモリセキュリティをベースとした「管理特権を利用した攻撃」への対策は，これまで説明した管理権限と情報オーナーの不一致がある場合に加えて，このような一般のサイバー攻撃対策としても有効である．

上述のように，サイドチャネル攻撃は本研究の直接対象外であるが，関連する RowHammer Attack について補足を加える． RowHammer Attack は Architectural side channel Attack に分類され，DRAM に高頻度アクセスすることにより隣接メモリセルのエ

ラーを発生させる物理メモリの性質をした攻撃である [41]．この手法によりアクセス制御を迂回する攻撃手法が知られており，遠隔実行の可能性もある．本研究の直接の課題からは外れるが，Rowhammer Attack 攻撃が及ぼす効果はランダム性をともなった物理メモリ改ざんと等価であることから，改ざん検証を含むメモリ暗号化は同手法の脅威を減少させることができるはずである．

1.5. 本研究の目的

本研究は，近年のコンフィデンシャルコンピューティング実用化に先立って著者が行った，端末コンピュータの物理的・ソフトウェア的管理権限の悪用防止をターゲットとしたメインメモリに対する機密性保護技術，完全性保護技術に関する提案と実装による機能検証からなる．いずれの研究も基盤技術として暗号技術を利用しているが本研究の主要な課題は，CPU ハードウェアによる効率的なキャッシュメモリ管理や OS によるプロセス管理，仮想マシンモニタが行うメモリ管理と，メインメモリに対する暗号処理を整合的に行うアーキテクチャを確立し，ハードウェア規模や性能オーバヘッドを緩和することにある．

機密性保護技術は，DVD や音楽のネット配信をサポートするデジタル家電におけるコンテンツ保護をターゲットとした機密性を保護する CPU の提案ならびに試作検証の報告である．提案方式はオープンソース OS 上でデジタルコンテンツ処理がプロセスの実行により行われるとき，メインメモリの暗号化ならびにプロセスの状態管理機能を含む HW 保護機能である．提案方式により，OS が不正に改造された場合を含めてシステム管理特権を利用してメモリ上のコンテンツまたはコンテンツの暗号鍵を不正コピーする攻撃を防止することができる．

完全性保護技術は，不揮発性メモリを持つシステムをターゲットとして，不揮発メモリに対する物理攻撃を通じてデータの改ざんによるシステムを不正動作させる攻撃を，検出し防止する完全性保護を実現するメインメモリ改ざん検出方式の提案ならびに試作検証である．

表 2　機密性保護技術，完全性保護技術の保護対象と脅威

<table>
<tr><th rowspan="2">セキュリティ属性</th><th colspan="2">情報資産に対する脅威</th></tr>
<tr><td>ソフトウェア攻撃
（OS/VMM特権の乱用）</td><td>メモリに対する物理アクセス攻撃</td></tr>
<tr><td>機密性
Confidentiality</td><td colspan="2">デジタルコンテンツ保護のための機密性保護技術</td></tr>
<tr><td>完全性
Integrity</td><td></td><td>社会インフラ機器向け
仮想マシンの完全性保護技術</td></tr>
</table>

機密性保護技術とは異なり，汎用 CPU 上の VMM により，大容量メモリを持つゲスト仮想マシン全体の改ざん検証を実現することで，1 ビットの改ざんが財産や人命の損害につながりかねない社会インフラ機器のサイバーセキュリティ対策を実現する．機密性保護，完全性保護の保護対象と脅威を表 2 にまとめる．

本研究では可用性(Availability)の保護はスコープ外とする．可用性の担保は適切な処理リソースが割当てられることが前提であるが，機密性保護技術においては OS が不正目的に利用されていることから可用性担保の前提がそもそも成立しない．また，完全性保護技術では改ざん検出後の処理は異常動作防止の観点から停止され，しかるべき回復処理が実行されるべきであり，改ざん検出時に処理中であった機能についての可用性はやはり担保できないためである．

1.6. 本論文の構成

本論文は以下の構成をとる．第 2 章では，DVD や音楽のネット配信をサポートするデジタル家電におけるコンテンツ保護をターゲットとした，機密保護 CPU の提案および試作検証の結果を述べる． 第 3 章では，不揮発性メモリを持つシステムをターゲットとして，不揮発メモリに対する物理攻撃を検出することで完全性保護を実現する，汎用仮想化機能を利用したメインメモリ改ざん検出方式の提案ならびに試作検証と性能評価の結果を述べる．第 4 章では，本研究発表後現在までに行われた商用 CPU におけるメモリセキュリティ実用化と，現在の商用 CPU で未解決の課題と本研究との関係について述べ, 第 5 章で結論として, 本研究の成果と今後の展望をまとめる．

2. 機密性保護技術の研究

2.1. 機密性保護研究の導入

コンピュータソフトウェアおよびコンピュータシステムで扱われる著作物に対する著作権侵害が問題になっている．とりわけ，利用者端末の OS がマルチベンダ・マルチタスクのオープンシステムであるとき，この問題は深刻となる．この解決のため，我々は，ソフトウェアベンダの立場からは利用者端末の OS が信頼できないことを前提として，マルチベンダ・マルチタスクのシステムにおいて，アプリケーションプログラムを解析や改ざんから守ることのできるプロセッサハードウェアアーキテクチャ L-MSP(License-controlling Mulit-vendorSecure Processor)を提案する．L-MSP では，マルチタスク OS に必要な資源管理機能と，アプリケーション保護のための秘密保護機能とを分離することで，信頼できない OS を持つ端末システムにおいても安全なライセンス管理を実現するソフトウェア実装の枠組みを提供する．

2.2. 機密性保護研究の背景

PC や，Linux OS ベースの情報家電システムが普及するにつれ，ソフトウェアの解析・改変が問題となりはじめている．一例としては，PC 向けのある DVD 再生ソフトにおいて，コンテンツスクランブルの復号鍵を秘匿化する実装の不備から，エンドユーザによって DVD の著作権保護システムが解析された事例が知られている［38]．DVD メディア上のコンテンツデータは暗号されているが［36]、ユーザ端末で利用されるソフトウェアやコンテンツの著作権保護には，ソフトウェアに埋め込まれた秘密や動作が，ユーザによって解析・改変されることを防止する秘密保護が要求される［42]．オープンシステムでは OS もユーザが自由に改変可能であり，アプリケーションソフトの秘密保護に対して OS の権限による攻撃が行われることを前提とした対策が必要となる．

オープンシステムに適用可能な秘密保護技術として，ソフトウェアに組み込まれたアルゴリズムや定数を秘匿化する TRS(Tamper Resistant Software）と呼ばれる技術がある．だが，TRS の安全性は実装方式の秘密性に依存するために，その利用範囲は限られている［43]．また，プロセッサチップが物理的な耐タンパ性を持つことを前提として，プロセッサに埋め込まれた固定の暗号鍵で，プログラムを復号して実行することでプログラムを保護する組込み

プロセッサが知られている [44]．だが，上記暗号鍵がプロセッサに埋め込まれている制約から，PC のように不特定のベンダが提供するプログラムの保護が必要なマルチベンダ環境のソフトウェア保護には適用できない．

2.3. セキュアプロセッサ

マルチベンダ環境に対応可能な，非対称鍵と対称鍵を組合わせたプロセッサハードウェアによるプログラム保護方式が提案されている [45, 46]．その概念を図 7 に従って説明する．システムユーザ(ユーザ)は潜在的な攻撃者であり，ターゲットシステム(システム)のストレージや外部メモリ，バスはユーザが読取り・改変可能である．システムには暗号機能を備えた耐タンパ性を持つプロセッサがあり，秘密鍵 A を保持する．秘密鍵 A は潜在的な攻撃者であ

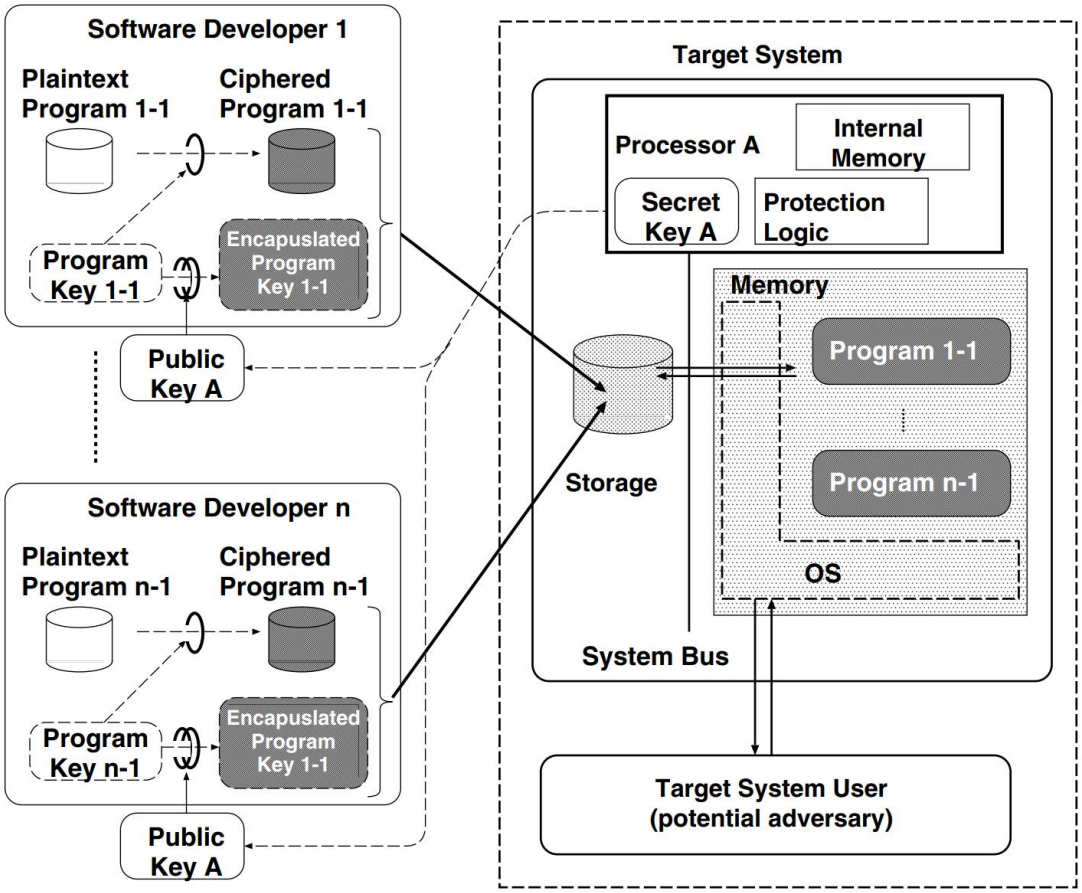


図 7　セキュアプロセッサにおけるソフトウェア配布

ソフトウェアベンダ(ベンダ)1 はプログラム 1-1 を作成し平文実行プログラムを得る．ベンダはプログラム 1-1 に対して，保護のためのプログラム鍵 1-1 を生成し，プログラム鍵 1-1 で平文実行プログラムを暗号化して暗号化済プログラム 1-1 を作成する．そして，ターゲットシステムのプロセッサが持つ秘密鍵 A に対応する公開鍵 A でプログラム鍵 1-1 を暗号化して，配布鍵 1-1 を得る．これら 2 つの情報がターゲットシステムに送られ実行される．これらはそれぞれユーザが知りえない鍵で暗号化されているため，どこに置かれても安全である．

プログラム実行時には，プロセッサが配布鍵 1-1 を復号してプログラム鍵 1-1 を取り出してプロセッサ内のテーブルに格納する．次にプログラム本体をプログラム鍵 1-1 で逐次復号して読み込みながらプログラムが実行される．ここでプログラム鍵の値を扱う部分はプロセッサハードウェアに予め組み込まれたメカニズムによって処理されるため，たとえ OS といえどもプログラム鍵を知ることはできない．

したがって，このようなシステムでは復号されてプロセッサ内に読み込まれた情報が安全であると仮定すれば，敵対的な OS の管理下であっても，実行プログラム本体に埋め込まれた秘密を，暗号の強度に基づいて守ることがマルチベンダ環境において可能となる．だが，プログラムの実行過程を詳細に見るとこの仮定は必ずしも成り立たない [45]．ひとつはプロセッサ内部のメモリに配置された情報に対しての OS 権限による参照がある．例えば，あるプログラムの実行によりキャッシュメモリに読込まれた平文情報は，外部割込みにより制御が OS に移った後も有効なため，OS 特権を利用すれば容易に読み出されてしまう．

また，マルチタスクシステムのプロセス切替では，レジスタ情報の保存・復帰は OS の役目である．実行状態の中のレジスタ情報をプロセス毎に切替えることはマルチタスクシステムに必須だが，切替の過程でプログラムのレジスタ値が，信頼できない OS によって解析されたり，任意の値に改変されるおそれがある．

これらの問題に対して，プログラムの秘密保護に関わるこれらの問題を解決し，マルチベンダ・マルチタスク OS の資源管理と両立を可能とするプログラムの秘密保護を実現するプロセッサアーキテクチャ L-MSP(License-controlling Multi-vendor Secure Processor)を提案する．L-MSP の特徴は名称に示すとおりソフトウェアによるライセンス管理と，そのソフトウェアが扱うデータを安全に保護管理できることである．

2.4. 要求とゴール

2.4.1. プロセッサハードウェア，OS の制約条件

現在のマルチタスクシステムは，プロセッサやメモリといった限られた資源を複数のプロセス間で多重化して，効率よく共有することを目的として作られている．我々の目標の一つは，資源を効率よく多重化するためのプロセッサや OS の基本的な機能と共存できるような秘密管理方式の提案である．本節では既存マルチタスクシステムのプロセッサアーキテクチャと OS のプロセス管理の制約から，要求条件を整理する．

図 8 に既存プロセッサの構成とプロセスの要素を示す．プロセスの構成要素はプロセッサによる扱いの違いにより，プログラム，データ，レジスタ（コンテキスト）の 3 種類に分類できる．プログラムはソフトウェアベンダから供給される．同一プログラム由来のプロセス間でプログラムは共通である．プロセスの実行状態情報はデータ及びレジスタに保持される．状態情報は各プロセスに固有である．プロセスの静的データは，ここでの議論ではプログラムに埋め込まれた定数の一部とみなしてさしつかえない．

プロセスの実行時には，プロセッサのバスインタフェースユニット(BIU)を通じて外部メモリからプログラムが読取られ，命令キャッシュ(I-Cache)に格納され，コアの実行ユニット(EXU)で解釈実行される．命令の実行を反映したレジスタ値の変更および外部メモリへの読書きが BIU， D-Cache を通じて行われる．

実行プロセスが切替えられるときは，プロセスのレジスタ内容は OS によりコンテキスト情報として外部メモリに保存される．メモリ管理ユニット(MMU)はプロセスビューから見える仮想アドレスと外部メモリ上の物理アドレスを変換して，各々のプロセスに他のプロセスから隔離されたメモリ空間を与えている．

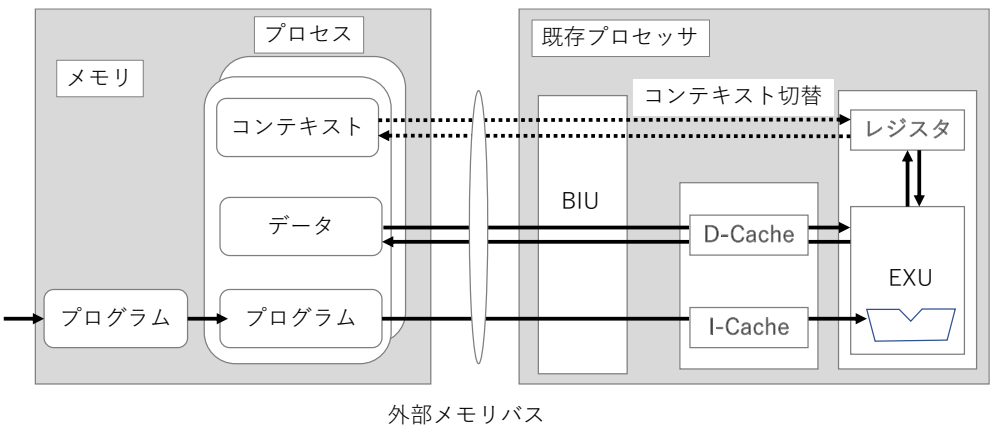


図 8　既存プロセッサとプロセス構成要素

図 9 は OS によるプロセス管理を概念的に示したものである．左側はメモリとプロセッサからなるハードウェアのビューを表しており，右側は資源を管理する OS と，OS が管理するプロセスから見える仮想的な環境からなるソフトウェアのビューを表している．

OS はプロセス対応に PCB(Process Control Block)と呼ばれる管理情報のテーブルを保持し，そこにはプロセスのメモリマップなどが格納されている．OS のメモリ管理機能は MMU のページテーブルを書き換えることで各プロセスビューに見える仮想的なアドレス空間を管理制限する．なお，図上では MMU はキャッシュメモリのバス側に置かれているが，これは説明のための図の制約によるもので，実際のキャッシュメモリには物理アドレスキャッシュを使うことを想定している．すなわちアドレス変換はコアからキャッシュメモリへのアドレス出力で行われる．また，プロセスの切替にあたっては OS のスケジューラが与えたポリシーに従ってディスパッチャがレジスタ情報の外部メモリへの退避および復帰を行う．これらの資源管理操作はプロセスビューからは透過的であり，プロセスは意識しない．

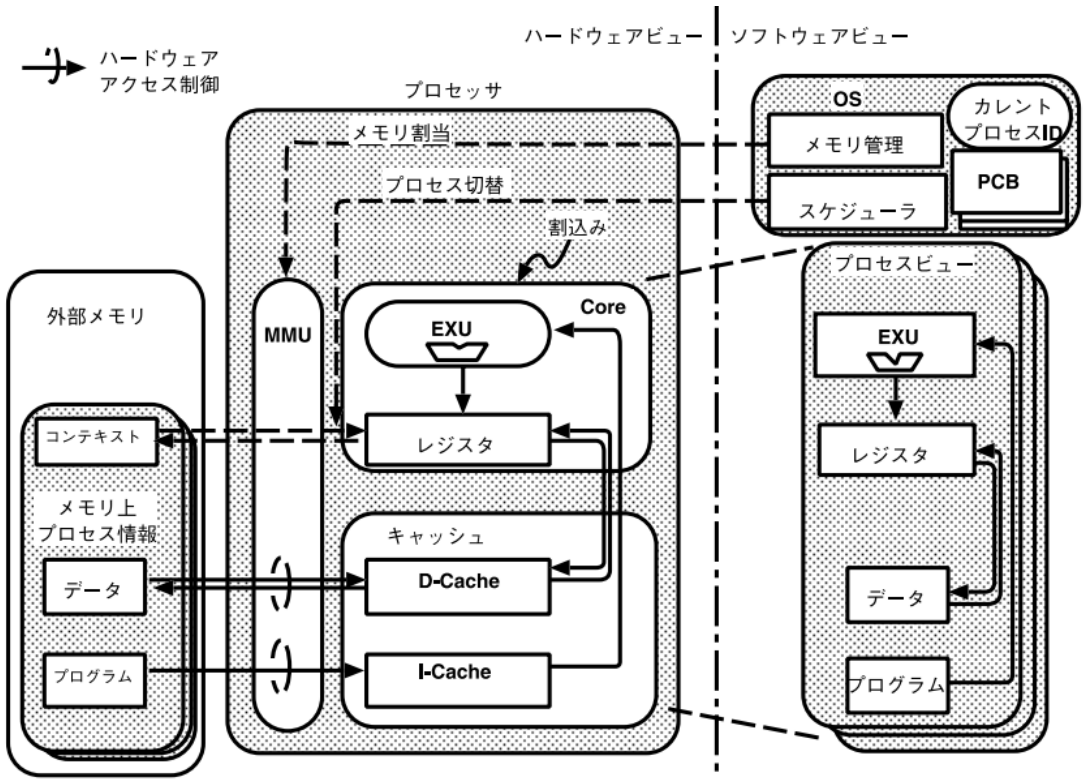


図 9　既存 CPU におけるプロセス管理

表 3　既存マルチタスクシステムの資源管理

対象リソース	リソース種別	管理	
		多重化制御	ポリシー
外部メモリ	メモリ	MMU	OS アドレスマップ管理
CPU 内部	キャッシュメモリ	キャッシュアクセス制御 HW	キャッシュリプレースアルゴリズム
	CPU 時間	OS ディスパッチャ	OS スケジューラ

いったんプロセスの実行が始まると，プログラムを命令キャッシュに読みこみ，命令を実行してデータキャッシュを通じてデータにアクセスするのはプロセッサであり，この過程には OS は介在しない．キャッシュ上には頻繁に参照されるデータが置かれるが，直前に実行されていた他プロセスのものも混在する．ただし，プロセスがアクセスできるアドレス範囲は MMU により制約されており，他のプロセスのデータを無制限に参照できるわけではない．

このようなプロセッサと OS の協調動作により，複数のプロセスに対して，あたかもそれぞれ専用のメモリ及びプロセッサコアが与えられてプログラムを実行して演算を行い，データの入出力を実行できるかのような仮想的なビューが提供される．上記システムの資源の多重利用と管理を表 3 にまとめる．

本研究のゴールは，OS および外部メモリ上の情報は信用できず，プロセッサ内部のメカニズムと情報だけが信用できる前提で，複数のプロセスに安全性を提供する秘密保護アーキテクチャを構築することである．プロセス情報はプロセッサ外部，内部のいずれにおかれた場合でも保護されなければならない．そして，既存 OS との整合性の観点から，秘密保護は OS による資源管理作との整合性を持ち，資源管理操作はアプリケーションから透過的でなければならない．

2.4.2. 秘密保護に関する要求条件

プロセスの秘密保護にはさまざまな観点とレベルがありうる．我々は次に挙げる項目を著作権およびプログラムの保護に関わる秘密保護の要求条件とする．これらの条件の背景については 4 節で詳細に議論する．

- プログラム中，レジスタ演算のみの部分の安全性はプロセッサにより保証する．
- プログラムは安全性を意識して記述されることを前提とする．プログラム自身のバグにより，秘密が漏れることは許容する．
- プログラムの特定部分だけでなく，全体を暗号化保護することでプログラム全体の同一性を保証する
- 異なる鍵で暗号化されたプログラムは識別される．
- 同一プログラム由来の複数のプロセスを識別し，個別に保護する．例えば同一プログラム由来のプロセス 1 からプロセス 2 へ，OS が勝手にデータをコピーすることを防止できる機構を備える．後述のソフトウェアによるライセンス条件管理の安全性確保が目的である．

2.5. 提案アーキテクチャのプロセス保護モデル

2.5.1. 孤立 ECU

本節では提案アーキテクチャ L-MSP 紹介の導入として，暗号化に基づいたプロセス保護モデルを単純化した形で説明する．L-MSP では，プロセッサが保護プロセスを認識してその秘密を保護する．プロセッサの保護対象となる暗号化されたプログラムを保護プログラムと呼ぶ． L-MSP は従来の平文プログラムも実行できるが，平文プログラムは非保護プログラムと呼び，L-MSP の直接管理の対象とはならない．同様にプロセスには保護プロセスと非保護プロセスとがある．

以下の説明では．OS によるプロセス管理と，プロセッサによるプロセス情報管理とを区別するため，プロセッサが認識するプロセスを実行制御単位(ECU: Execution Control Unit）と呼ぶ．ECU のセキュリティ制御情報は OS の PCB とは独立にプロセッサ中に保持される．

表 4　ECU 操作特殊命令

命令	内容
ECU 登録（register）	タグ ID に対する配布鍵復号とプログラム鍵登録ならびにコンテキスト鍵初期化
ECU 実行開始（start）	初期状態からの ECU 実行開始
ECU 実行再開（continue）	中断状態からの ECU 実行再開
ECU 削除（delete）	プロセッサ内部の ECU 情報の削除

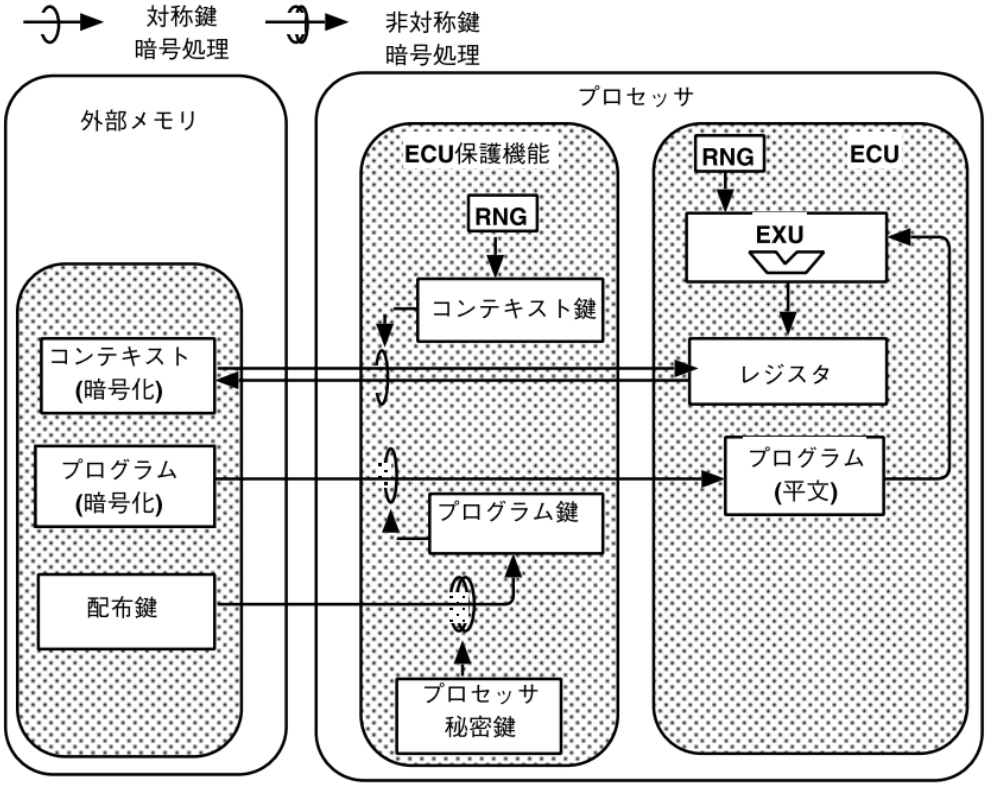


図 10　孤立 ECU に対する保護モデル

図 10 に孤立 ECU の保護モデルを示す．孤立 ECU はデータ入出力を持たない，プログラム，レジスタ，EXU による計算機能だけを持つ仮想的な ECU である．ECU には ECU ID が割当てられ，プロセッサ内部で一意に識別される．実際のプロセッサでは複数の ECU が時分割多重的に実行されるが，ここではただひとつの ECU だけを示している．

プロセッサには ECU 保護機能があり，ECU の状態や暗号鍵を管理している．OS は表 4 に示す特殊命令の発行を通じて ECU 保護機能に指示を出すことで ECU を生成(登録)ならびに実行の中断・再開を行うことができるが，ECU 保護機能内部の情報は ECU 保護機能によりカプセル化されており，OS が ECU の秘密情報を直接操作することはプロセッサの HW 仕様上禁止されるが，プロセスの実行制御は可能である．特殊命令の一覧を表 4 に示す．

プログラムの実行が始まると，命令フェッチにより，プログラムが逐次復号されてプロセッサ内部に読込まれ，EXU により解釈実行される．図では単純化のためキャッシュを明示していないが，プログラムはキャッシュラインサイズを単位としてブロック暗号化されており，キャッシュフィルに同期して復号処理される．

割込みにより ECU の実行が中断されるときは，ECU 保護機能がレジスタ値をコンテキスト鍵により暗号化して，外部メモリに退避する．実行再開時はやはり ECU 保護機能がレジスタ値を復号して復帰する．この処理はハードウェアコンテキストスイッチとして実装されるため，OS は ECU の実行再開を指示することはできるが，ECU のレジスタ値を知ることも操作することもできない．

ここで，プログラム実行中の ECU 要素に対する外部メモリにおける情報の偽造攻撃を考える．配布鍵は ECU 開始時にただ一度だけ読込まれるため，読込み完了後のメモリ上の偽造は攻撃として有効ではない．プログラムはプログラム鍵によって復号されて実行されるので，プログラム鍵を知らずにプログラム断片を置き換えたとしても，でたらめな命令列を挿入するにすぎず，有効な攻撃とはならない．また，ECU の実行停止中にメモリ上のコンテキストを置換する攻撃についても，攻撃者が RNG によって設定された鍵を知らない限り，意図したようにレジスタ値を改変することはできない．異なる ECU 間でコンテキストを置換する攻撃については，置換対象がたとえ同一プログラム由来のプロセスのコンテキストでも，個々のプロセス毎に RNG で設定された鍵が異なるため，置き換えは有効ではない．ECU 保護機能は，外部メモリ上の汚染された情報から ECU を保護するいわばフィルタとして機能し，ECU の安全性を保証している．

L-MSP の孤立 ECU モデルでは ECU ID とプログラム鍵およびコンテキスト鍵が統合されて管理されることにより，たとえ同一プログラム由来のプロセス間であってもプロセス状態を他のプロセスへ複製することを防止している．これはライセンス管理のソフトウェア実装を容易にする．なお，プロセッサ上の ECU の秘密情報は，OS により ECU の削除が指示されたとき，全て消去される．

2.5.2. データを含む一般 ECU

前節の孤立 ECU モデルでは，配布鍵，プログラム，コンテキストの ECU 要素への置換攻撃に対してプロセッサハードウェアが情報の安全性を検証することができた．この検証処理は実行対象のプログラムからは透過的である．

本節では前節のモデルを拡張して，図 11 に示すデータの入出力能力を持つ一般 ECU 保護モデルを説明する．実用的な ECU が扱うデータ入力には，OS から割当てられたメモリ領域のポインタやユーザのキー入力などがあり，ECU の観点から見て安全とはいえないデータを扱うことが避けられない．これはアプリケーションに関わる問題であり，プロセッサが単独でその情報の安全性を判定することは一般には不可能である．本節では L-MSP アーキテクチャにおいて，汚染されたデータの入出力がある場合でも，プログラミングが十分注意深く行われることを前提とすれば，安全なアプリケーションが構築可能であることを述べる．

一般 ECU でも，プログラムの一部に限定すればデータ入出力を行わないように記述することは可能であり，その部分は孤立 ECU として機能する．したがって，外部からの入力値についてプログラム自身が検証を行うことにより安全性を確認できる．入力値に対して検証アルゴリズムが存在し，検証アルゴリズムがプログラムに実装され，かつその部分の実装が孤立 ECU として記述されていれば，そのプログラムは安全に入力値を検証することが可能である．

たとえば通信相手のサーバからのメッセージの署名検証アルゴリズムおよび信頼できるサーバの公開鍵がプログラムに埋め込まれており，それらが孤立 ECU として実装されていれば安全にメッセージを検証することができる．サーバの公開鍵ではなくルート認証局の公開鍵と必要な認証アルゴリズムが埋め込まれていれば任意のサーバからのメッセージを検証できる．このような検証済みの入力値はプログラムに埋め込まれた定数と同様に安全なものとして扱うことができる．

もしこの検証過程の中間値や結果がレジスタに収容しきれない場合には，これらの値に署名を付加した上で，暗号化して外部メモリの作業領域に書き出す．署名および検証に使う秘密値はレジスタに保持し，外部メモリには書き出さない．再度その値を利用する際は外部メモリから読み出した値を秘密値に基づいて検証して利用する．

ECU はデータの暗号化処理入出力のために ECU 保護機能がそなえるバス暗号機能を暗号アクセラレータとして利用することができる．この目的のため，各 ECU 対応にデータ鍵レジスタと呼ばれる特殊レジスタが設けられる．ECU がデータ鍵レジスタに暗号化対象の仮想

アドレス範囲と暗号鍵を設定すれば，以後の対象領域のメモリアクセスでは透過的に暗号・復号処理が行われる．

上述のように現実のアプリケーションは汚染されたデータを扱わざるをえない．汚染されたデータを隔離するための，あるいは安全性検証のポリシーをセキュリティプロトコルに基づいて定めるのはアプリケーション自身であり，プロセッサはその実行を暗号ハードウェアなどにより支援する．この隔離もしくは検証動作そのものの安全性を裏付ける機能として，L-MSP ではプログラムとレジスタだけからなる孤立した演算の安全性をプロセッサのメカニズムにより保証している．一般 ECU において，汚染されたデータをフィルタするのはプログラム自身の責任である．

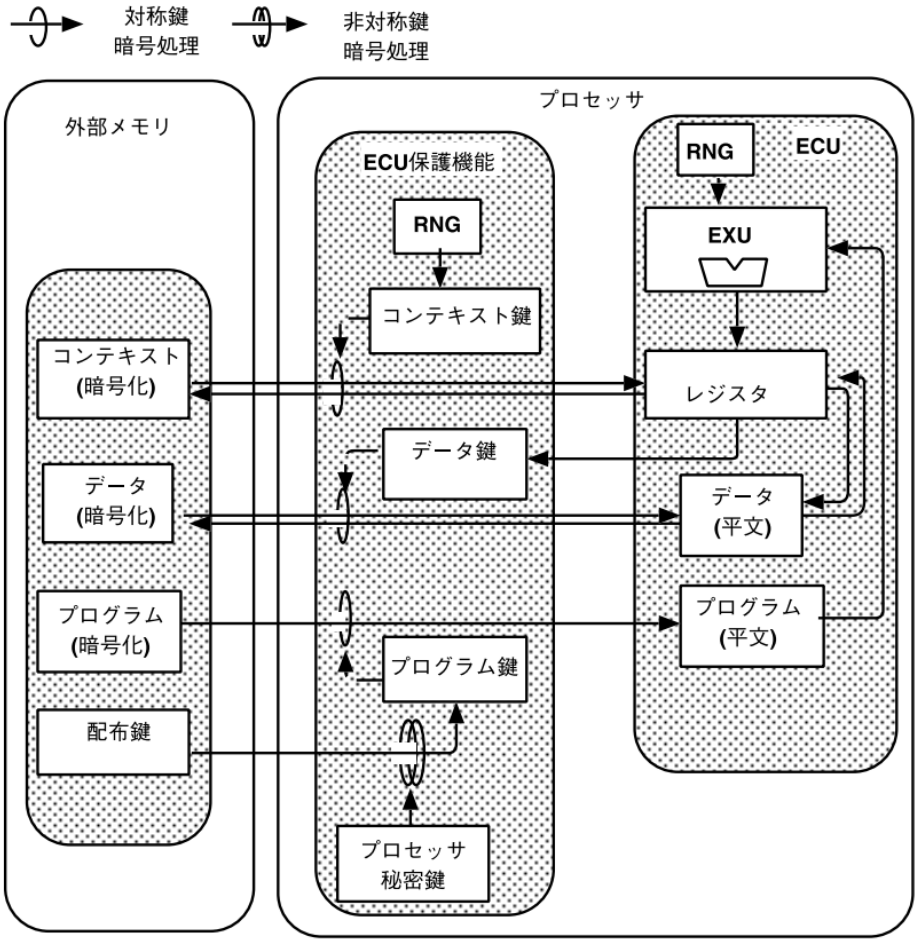


図 11　一般 ECU の保護モデル

2.6. 提案アーキテクチャの詳細

2.6.1. ハードウェア構成

図 12 に提案アーキテクチャのハードウェア構成を示し，既存プロセッサ図 8 との差分を説明する．なお，本稿の目的は提案アーキテクチャにおける秘密情報の管理方法の明確化であるため，ここでは暗号アルゴリズムや鍵長は議論しない．BIU には対象鍵暗号ハードの SCH(Symmetric Cipher Hardware)が組み込まれている．コアに隣接してコンテキスト切替のための HCS (Hardware Context Switcher)がある．ECU 情報全般を管理する ECU マネージャ(ECM)は，鍵テーブル，カレント ECU ID，プロセッサ秘密鍵，ECU 状態管理機能などを含んでいる．ECM はコアが発行した ECU 操作命令に従ってカレント ECU ID や鍵テーブルを更新する．非対称鍵暗号ハードの ACH(Asymmetric Cipher Hardware)はプロセッサ秘密鍵を ECM から受け取り，配布鍵の復号結果をプログラム鍵テーブルへ送る．

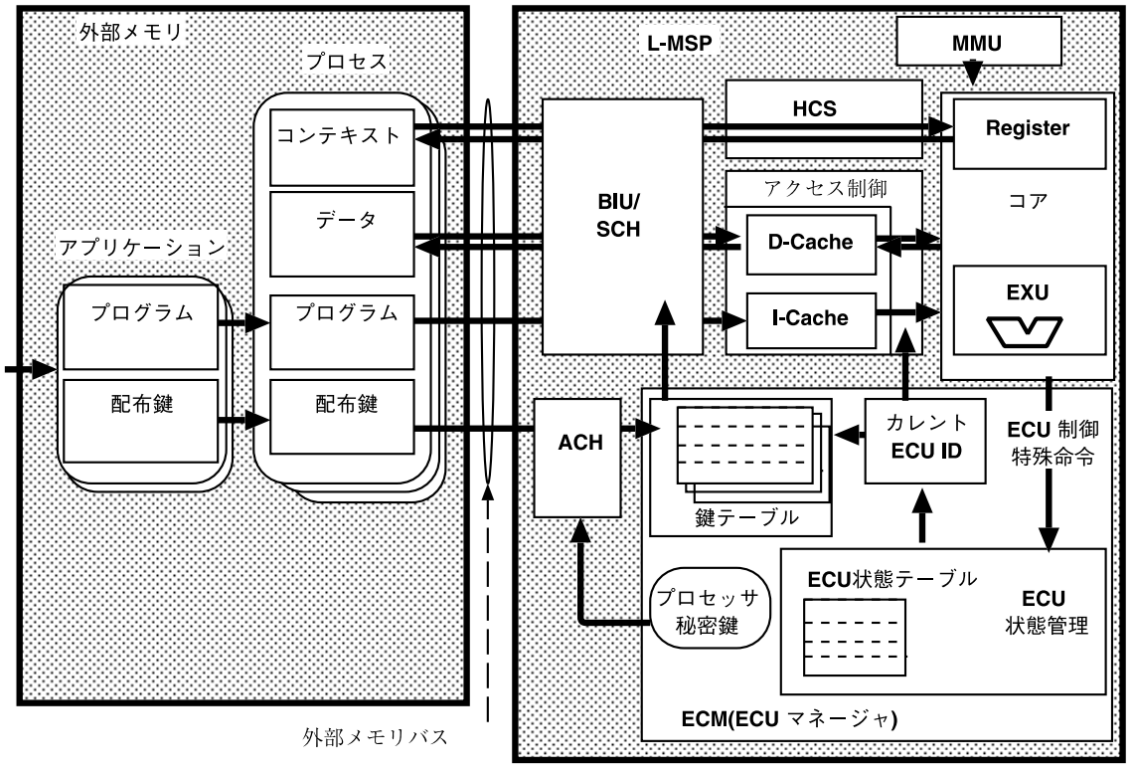


図 12　提案方式のハードウェア構成

キャッシュにも変更がある．図 13 にキャッシュのデータ構造と動作を示す．従来のキャッシュにあった有効ビットとアドレスのタグ情報に加えて，提案方式では ECU ID のタグが追加されている．キャッシュラインのフィルでは読込み時のカレント ECU ID が書き込まれる．キャッシュのヒット判定時にはアドレスタグの比較に加えてカレント ECU ID とタグの ECU ID が一致しなければヒットしない. つまり, 最初にそのラインを復号して読込んだ ECU 以外の ECU は，キャッシュに保持されたラインを読むことはできない．その場合，後から同一アドレスをアクセスした ECU は当該 ECU の暗号鍵により復号してそのアドレスにアクセスする. すなわち同一内容であっても読出し主体の ECU が異なれば読出し結果が異なる．図 11 に示した ECU 保護機能は，上記ハードウェアに含まれる SCH，ACH，HCS，ECM，キャッシュコントローラの連携動作の結果として実現されるものである．

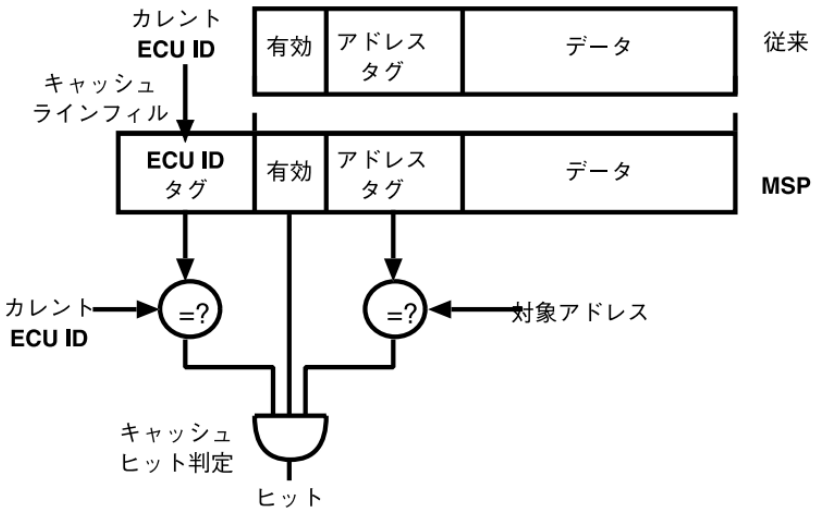


図 13　提案方式のキャッシュヒット判定機構

図 14 に L-MSP における ECU 管理の概念図を示す. これは既存システムにおける図 9 に対応する. 以下提案方式の既存システムに対する変更点を ECU の動作とあわせて説明する．ソフトウェアビューにおける変更は，OS から ECU 生成・削除を ECM に指示するプロセス生成・削除機能が加わったことである．また，プロセスビューにはデータ鍵レジスタが追加されている．以下の説明では，ECU ID=#1 を例として ECU の生成・実行動作を説明する．操作対象の ECU ID =#1 に相当する鍵テーブルエントリを図上では黒く塗りつぶして表している．

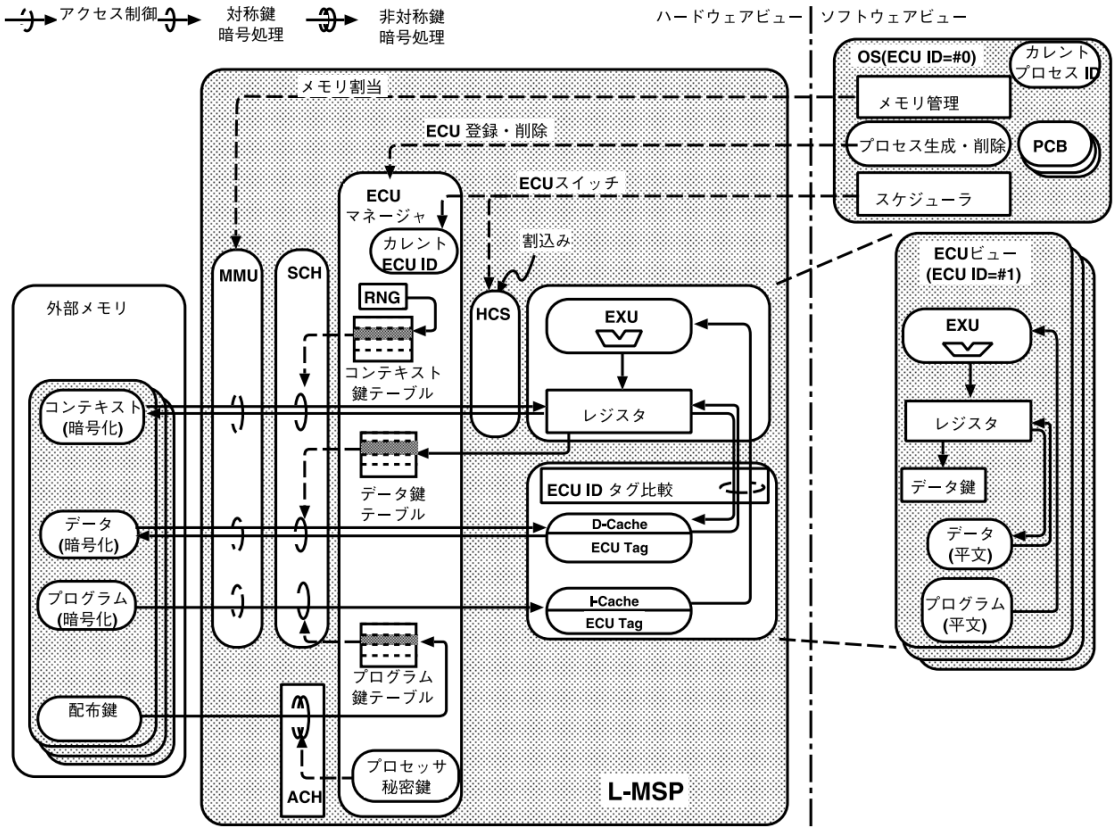


図 14　提案方式における ECU 管理

OS が ECU ID = #1 を指定して ECU 登録命令を発行して ECU 生成を ECM に指示すると，配布鍵が ACH により復号され，プログラム鍵がプログラム鍵テーブルの#1 エントリに格納される. OS の ECU 実行開始命令発行によりカレント ECU ID が#1 にセットされ, ECU の実行が始まると，外部メモリから読み込まれたプログラムはカレント ECU ID＝#1 で指定されるプログラム鍵により SCH で復号され，キャッシュラインに格納される．このとき，キャッシュラインの ECU タグには#1 が格納される．MMU はここでも有効であり，秘密保護の有無に関わりなく ECU は MMU が許可しないメモリ領域を参照することはできない．キャッシュラインの内容が EXU に読み込まれるときには上述のキャッシュヒット判定が行われる.

ECU 実行中に割込みが発生すると，HCS によりレジスタ値がやはり ECU ID =#1 のコンテキスト鍵により暗号化されて外部メモリの所定アドレスに退避される．そして，割込みハンドラに制御が移る前にカレント ECU ID は#0 に更新される．ECU ID=#0 は OS を含む非保護プログラムに割当てられており，ECU ID=#0 のとき暗号処理は行われない.

OS スケジューラが ECU #1 の再開命令を発行すると中断された ECU #1 の実行が再開される．ECM は外部メモリから #1 のコンテキストを読みこみ，コンテキスト鍵で復号して HCS がレジスタ値を復帰する.復帰対象には PC(プログラムカウンタ) が含まれているので，中断した状態から実行が正しく再開される．

ECU は，データ鍵レジスタに暗号化対象アドレス範囲と暗号鍵を設定することで，SCH を暗号アクセラレータとして利用しながらデータの読書きができる．データ鍵テーブルへの書き込みは実行中の ECU と同一のエントリに限定され，他の ECU のテーブルに書き込むことはできない．

これらの機構によって，ソフトウェアビューとしては，個別の ECU 毎に秘密保護された実行環境が提供される．そして，この秘密保護はメモリ管理やスケジュールのような OS の資源管理機構と共存できる．表 5 に L-MSP の秘密保護に関係するハードウェア機構とその動作指示についてのまとめを示す．既存システムの資源管理をまとめた表 3 との比較で特筆すべき点は保護対象としてプロセッサ内部の ECU 鍵テーブルが追加されたことである．ECU 鍵テーブルは ECM によりカプセル化されており，たとえ OS が信頼できず，指示が不適切であっても ECU の秘密は保護される．

表 5　提案方式の秘密保護

対象リソース	**保護対象リソース**	**管理**	
		多重化制御	動作指示
外部メモリ	メモリ暗号化	SCH と ECU 対応鍵	ECU 実行による メモリ参照
CPU 内部	キャッシュメモリ	タグ比較	キャッシュ 参照
	CPU レジスタ	HCS	割込みおよび OS ディスパッチャ
	ECU 鍵テーブル	ECU マネージャ	OS プロセス管理

2.7. 実装と評価

図 12 に示したハードウェア構成を FPGA 上に実装し，対応する RTOS 実装とあわせて機能評価を行った．実装に使用した基盤を表 6 に，FPGA 評価基板の写真とソフトウェア構成を図 15，図 16 にそれぞれ示す． CPU コア，OS はそれぞれ MeP， uITRON を使用した [47, 48] ．

表 6　実装の基盤

ベースプロセッサコア	Toshiba Media Embedded Processor
記述言語	Verilog-HDL
Hardware	Altera EP1S80 FPGA (800KG 相当)
OS	uITRON (TOPPERS JSP)

図 15　評価 FPGA 基板

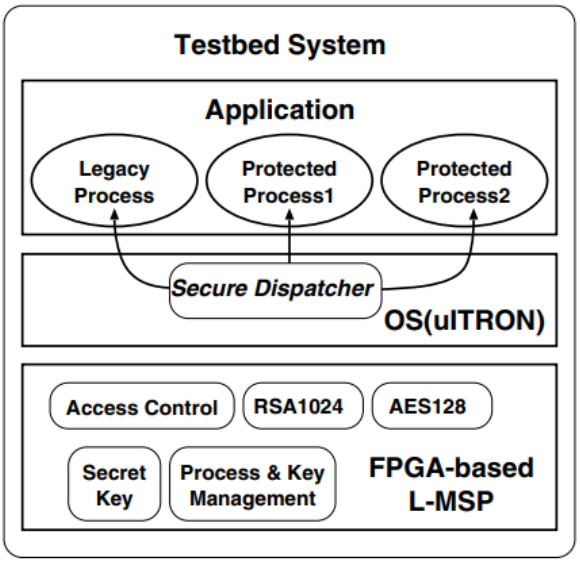


図 16 ソフトウェア構成

回路規模と性能オーバヘッドを表 7 にまとめる．共通鍵暗号は AES128 ビット，公開鍵暗号は RSA1024 ビットを使用している．実装当時の FPGA 規模の制約により RSA は 1024 ビットを使用した．これら暗号コアと 4 個のプロセスの命令，データ，コンテキスト鍵を格納する共通鍵テーブルを含んだ回路オーバヘッドは，ベースとなる CPU コアと同等の 200KG に達した．1NAND Gate が 4 トランジスタに相当とすれば，200KG は 800K トランジスタに相当する．これを SRAM 換算すると，SRAM 1bit は 6 トランジスタで構成されるため，133K bit すなわち約 16.7KiB に相当する．上述のように，追加部分は CPU コアと同等の回路面積であり，決して小さいとは言えない．だが，メモリ換算では 16KiB と同等であり，試作当時の組込みプロセッサでも 16KiB のキャッシュメモリを持つものはまれではなく，実用化のスコープに入る値である．

性能インパクトは, FPGA 実装は回路制約により極めて実行クロックが遅いため, ASIC 向け CMOS プロセス（0.13um）向けに動作速度 200MHz で論理合成した結果に基づくシステムシミュレーションの結果である．データの AES128 ビット共通鍵暗号処理による遅延は約 100ns であり，これは当時の DRAM レイテンシ 約 100 ns と同等である．つまり，暗号化処理が入った場合レイテンシが 2 倍の DRAM を使った場合と同等のパフォーマンス低下が生じる．これはキャッシュヒット率に依存し，キャッシュヒット率 98%ではスループットは約 10%低下，キャッシュヒット率 99%では 1%の性能低下となる．

表 7 回路と性能オーバヘッド

項目	詳細	オーバヘッド
ゲートサイズ	トータル	400KG
	基本 CPU コアと周辺回路 （キャッシュメモリは含まない）	200KG
	提案方式の回路オーバヘッド	200KG
性能オーバヘッド	キャッシュヒット率 98%	10%
	キャッシュヒット率 99%	1%

2.8. 議論

2.8.1. デジタル著作権保護とプロセス保護

デジタル著作権保護とはユーザ端末に置かれたデジタル情報を，権利者が認めた形態でのみ利用可能とするような管理と定義できる．DVD-Video の場合には，物理媒体を保持していることがライセンスであり，コンテンツのネットワーク配信においてはサーバからの許諾などがライセンスである．

ライセンス管理の機構を柔軟に実現するには，実装は固定されたハードウェアよりもソフトウェアによることが望ましい．その場合，利用許諾そのものは物理媒体やサーバからのライセンス許可情報として送信されるなどさまざまな形態がありうる．だが最終的にそれらの利用許諾条件を解釈して利用可否を判断するのは端末プログラムの実行であり，判断結果は一時的にせよプロセスの実行状態として保持されるはずである．

したがって，利用許諾を受けた状態のプロセス状態を複製・保存し，任意の時間，任意のシステムで再実行することが可能となってしまえば，ライセンス許諾の仕組みが破壊されることにつながる. L-MSP ではプログラムだけでなく, プロセスの実行過程を改変から保護し，プロセス状態の複製を禁止する機構を設けたことにより，管理者が信頼できない端末においても安全なライセンス管理を容易にするものである．

既存 CPU をベースとした実装においてもサーバとの間で再認証を行うなどのプロトコルの工夫により，ライセンス管理における危険のかなりの部分は排除できるが，実行状態が他のシステムに移動不可能であることおよび複製不能であることが保証されればプロトコルを単純化することができ，サーバ負荷の低減およびプログラムの作成効率向上が期待できる．

今日のシステムでは，メディア処理はライセンス管理や画像，音声のレンダリングなどが複数のプロセス連携やアプリケーション本体と共有ライブラリとの連携により処理される構成が多い．このような場合，複数のプロセス間で安全なデータ交換のため，暗号化保護されたデータ領域の共有が必要となる．連携する 2 個の保護プロセス(ECU)がともに参照可能なメモリ領域に対して同一の暗号鍵で暗号化保護する設定を行うことにより，安全なデータ交換を行う方式が寺本らの特許出願として公開されている [49, 50]．さらに同一の暗号鍵で暗号化される領域に対して，同一のタグ ID を割当てるハードウェアタグ ID 管理を追加することにより，キャッシュ上のデータを安全に共有することが可能となる．

2.8.2. 秘密保護と資源管理

L-MSP では秘密保護と資源管理の機能の直交化を目標としている．その背景を述べる．著作権保護などの目的により，プロセスの秘密保護を要求する主体は一義的にはソフトウェアベンダである．コンテンツホルダは，暗号化コンテンツの復号鍵をソフトウェアベンダに預託し，コンテンツ再生権の管理を端末で実行されるソフトウェアに委任したものととらえることができる．一方，OS にはユーザの意思に従って CPU 時間およびメモリ資源をプロセスに割当てられる資源管理能力が求められる．これらはソフトウェアベンダとターゲットシステムユーザという異なる立場からの要求である．

ここで，システムのユーザの観点からは，秘密保護の有無に関わらず資源保護機能は常に有効である必要がある．たとえば，あるプロセスが暴走して多くの CPU 時間とメモリを占有してしまった結果，ユーザがそのプロセスの実行中止を指示した場合は，たとえそのプロセスが秘密保護の対象であっても即座に実行を中止し，メモリを回収できなければならない．対象プログラムがいわゆるウィルスであった場合を考えればその必要性が理解されよう．秘密保護されたプログラムの実行によりシステムのリソースが不足した場合には，そのプログラムに割当てられたリソースを回収できなければならない．

一方，ソフトウェアベンダの立場からもプログラムの秘密保護機能が資源管理に優先されることは不要な責任を負うことになり，必ずしも好ましくない．一例を挙げれば，秘密保護対象のプログラムが不具合により暴走してしまった場合にユーザが実行の中止手段を持たなければ，ベンダがその結果に責任を負うことになり，プログラム開発の困難度が増えてしまう．ベンダの要求はプログラムの実行が中断されないことではなく，むしろ実行が中断されても秘密が保護されることである．

この状況を，セキュリティの基本 3 要件である機密性(Confidentiality)，完全性(Ingetgrity)，可用性(Availability)にあてはめて考えたとき，次のように解釈できる．ベンダは，提供したプログラムについて，それが実行されるときに秘密保護である機密性と完全性を要求するが，実行が中断されないという，可用性は要求しない．

一方，ユーザの立場からは，ベンダから提供されたプログラムについて機密性と完全性は要求しないが，そもそもプログラムを実行したのはユーザ自身の意思なので，プログラムの可用性が求められる．なお，この場合のユーザが機密性および完全性を要求しないとは，ベンダから提供されたプログラムについてであって，ユーザ自身に帰属するデータについては当然機密性および完全性を要求する．

このようにシステム外部由来のプログラムのセキュリティ要件については，通常の場合とは異なり，3 要件を分離して考えなければならない．L-MSP は，ユーザとベンダの間に立つ**信頼できる第 3 者(TTP:Trusted Third Party)** の役割を果たすものである．OS は信頼できない通信路の一部を構成する．

2.8.3. デジタル著作権保護の依頼計算モデル

上記のベンダとユーザ，そして TTP の関係は図 17 に示す依頼計算の特殊な場合としてモデル化できる．ソフトウェアベンダは依頼計算の依頼者であり，プログラムをターゲットシステムに送付する．また，プログラムが扱うコンテンツはコンテンツホルダから暗号化した形でシステムに送付され，その暗号鍵であるトレードシークレットは別途ベンダに送付される

システム上のバスや外部メモリは信頼できない通信路であり，プロセッサはその通信路を通じて送られた暗号化されたプログラムに改変のないことを検証しながら，外部にその秘密を漏らさないように実行する．OS も同通信路の構成要素である．一般の依頼計算では実行結果が依頼者に返されるが，このモデルでは実行結果は依頼者に必ずしも返されなくてもよい．代わりに，実行の副作用である出力，たとえば画像や音声などが，システムのユーザに渡さ

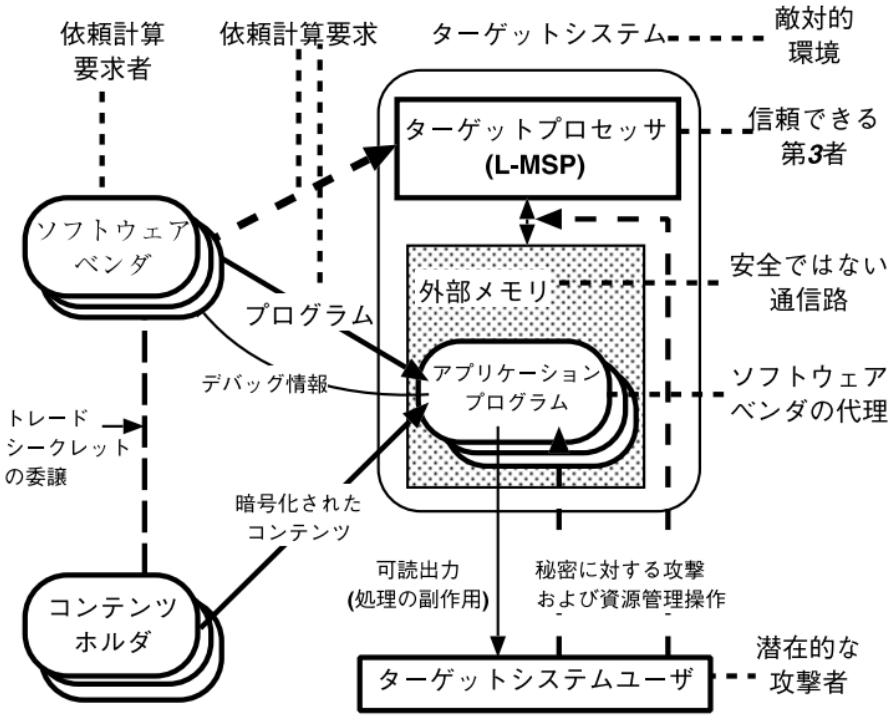


図 17　デジタル著作権保護システムの依頼計算モデル

れる．一般のユーザはサービスの提供相手であると同時に，潜在的にはプログラムに対する攻撃者となる．ここではプログラムはベンダの代理人として機能している．

ここまでプログラム鍵をプロセッサ公開鍵で暗号化したものを配布鍵と呼んでいた．だが，これは鍵の配送よりもむしろ TTP であるプロセッサへのカプセル化されたセキュアなメッセージ転送である．したがって配布鍵にはプログラム鍵に限らず，プログラムのライセンス条件などの，プロセッサに安全に配送する必要のある情報が含まれてよい．プロセッサに埋め込まれた秘密鍵は，プロセッサに安全に鍵情報を配送する手段だけでなく，ソフトウェアベンダに対してそのプロセッサが安全であることを証明する手段としても機能する．その意味で，ソフトウェアベンダがプロセッサに配布鍵を発行する際には，その秘密鍵が正当なものであることを証明書の検証などで確認する必要がある．

提案方式のプロセッサはコンシューマ機器におけるデジタル著作権保護をターゲットとしたものではあるが, 上記の考え方はクラウドコンピューティングと共通するものである. 2004 年の著者論文において，提案方式はターゲットシステムの管理者に対する信頼要件を緩和することにより，間接的にグリッドコンピューティングのメンテナンスコストを引き下げる効果が期待できることを論じている [51]．下記は同論文からの引用である．

> As for grid system operation, L-MSP may reduce system maintenance cost. In current grid computing system, remote system administrators are assumed to be reliable for data security, and those system requires tight authentication. But in L-MSP system the security of remote computation is resolved into the matter of the security level of L-MSP, instead of target system administrator.
>
> （訳）L-MSP によりグリッドシステムの運用コストを削減できる可能性がある．現在のグリッドコンピューティングシステムでは，リモートシステム管理者はデータセキュリティの信頼性が高いと想定されており，それらのシステムには（管理者に対する）厳格な認証を要求している．だが，L-MSP システムではリモート計算のセキュリティは、ターゲットシステム管理者ではなく，L-MSP によって提供されるセキュリティレベルの問題となる．

2.8.4. プライバシ保護との関係

序論ならびに依頼計算モデルで述べたように，モバイルコードやクラウドシステム（グリッドコンピューティング）では、要求者から計算が実行されるリモートシステムにプログラム及びデータを送信する必要がある．だが，送信したデータがターゲットシステムのオーナーに漏洩するリスクがある［31］．本提案のようなターゲットシステムの管理権限から独立したプログラム並びにデータに対する保護手段によって技術的な対策が可能となる．

一方，要求者が送信したプログラム（外来プログラム）が暗号化により保護されていた場合，ターゲットシステムのオーナーは外来プログラムに対する検査ができないため，今度は外来プログラムがターゲットシステムの情報に対する機密漏洩・改ざんの損害を与える懸念が生じる［31, 32］.

この問題の解決として，このような保護された外来プログラムを実行する際にはサンドボックス内で実行する方法が考えられる．サンドボックスは既存 OS が備える MMU に基盤を置くアクセス制御により，外来プログラムがアクセス可能なデータ，ファイルをターゲットシステムのポリシーにしたがって限定する．これにより，外来プログラム内部の動作が不明であってもターゲットシステムのプライバシ及び完全性を守ることができる．外来プログラ

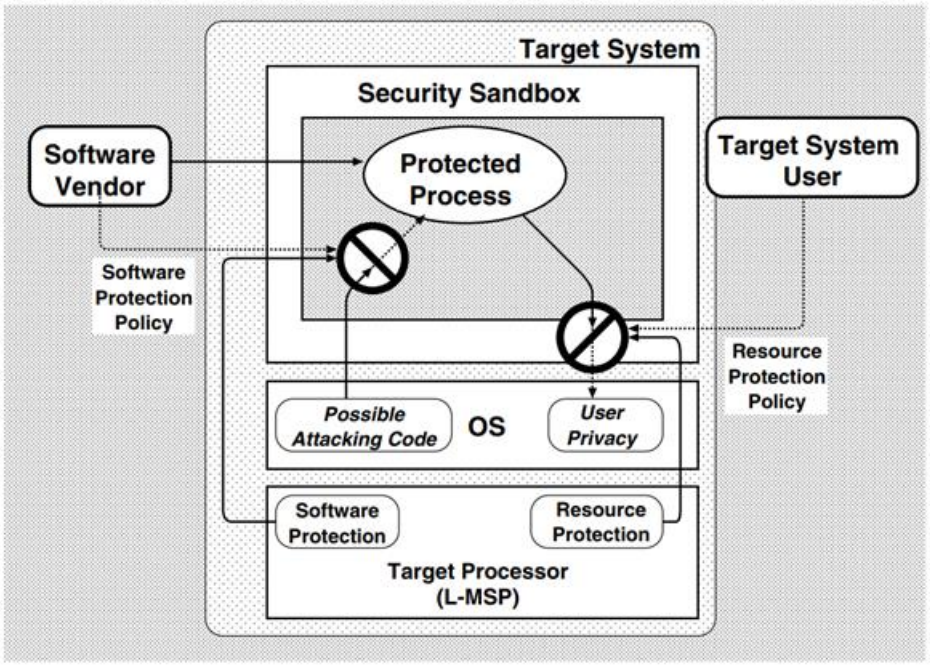


図 18　サンドボックスとの連携

ムに対する保護と，外来プログラムのアクセス可能範囲を制限するサンドボックスとの関係の概念図を図 18 に示す.

ただし，この方式を適用するには，外来プログラムがターゲットシステムの資源管理下で実行されなければならない. Microsoft NGSCB や Arm Trustzone においてはセキュアゾーン（セキュアワールド）においては，非セキュア側からの制御を受けずにプログラムを実行を行うモデルを採用している [52, 53]. この方式はセキュア側の実行に対する可用性を確保できるメリットはあるが，一方でセキュア側から非セキュア側へのアクセスを非セキュア側が柔軟にコントロールすることが困難であるという問題があり，外来プログラム毎にアクセス制御範囲を変更するサンドボックスモデルの適用は難しい．一方，提案方式では，保護されたプロセスも通常のプロセスと同じく OS によるアクセス制御を制限を受けるため，サンドボックスモデルによるターゲットシステムの情報保護との整合性がよい.

さらに，提案方式の保護プロセスに対して，OS は任意のタイミングで実行を中断し，割当てたメモリを回収することができる．したがって，保護プロセスが大容量の動的メモリを用いる学習系のような処理を実行しかつバグがあったとしても，システムが利用不能となることは回避可能である．一方セキュア処理に割当てたメモリを非セキュア側が回収できないモデルではセキュア側にメモリ管理のバグがあった場合に非セキュア側が動作不能に陥る危険がある.

2.8.5. オーバヘッド

実装で説明したように，提案方式の実行時オーバヘッドはキャッシュヒット率に依存し，ヒット率 99%では性能低下は 1%程度にとどまる．既存システムではデジタルコンテンツ再生ソフトにはソフトウェア難読化の手法が適用されている．ソフトウェア難読化にはさまざまな手法があり，当然オーバヘッドも方式に依存する．ただし，命令とデータの両方を秘匿する場合，実行前は秘匿状態でメモリに格納し，実行前にソフトウェア処理により復号してキャッシュにストアし，その後命令として実行またはデータとして参照する手順が必要となるはずである． 1 ワードの命令実行前に，暗号化状態の命令データを読み込み，ソフトウェア処理により復号してキャッシュにストアする処理が必要となり，この手順を復号済み命令がキャッシュからパージされるたびに行わなければならない．実行時間は少なくとも元のプログラムの数倍のオーダーとなるはずである.

提案方式では上述のようにキャッシュヒット率 99%が達成されればオーバヘッドは 1%程度となり，ソフトウェア難読化と比較して大きな性能改善が見込める．ソフトウェア難読

化は PC 向けには広く使われているが，組込みでの採用事例はおそらく性能ならびに消費電力の問題でごく少ない．提案方式は性能と消費電力の制約が強い組込みシステム向けに有効性が見込める．

2.9. 関連研究

Gilmont らはプログラム毎に固有の秘密鍵を持たせ, バス暗号化を行うアーキテクチャ提案を行い，シミュレータによる性能評価を行っているが，レジスタ情報の秘匿や鍵管理は考慮されていない [46].

Lie らは，XOM アーキテクチャにおいて，プロセスの実行状態保護としてデータ及びコンテキスト情報の保護手法を示している [45]．ここではプロセッサ内の秘密情報の保護にタグを適用することが示されている．XOM においてはレジスタ個別に暗号化してキャッシュ保存することにより保護するメカニズムが提案されている．一方，L-MSP においては，プロセスのレジスタ情報の全体をまとめてキャッシュメモリに一時退避し，キャッシュメモリから外部メモリへの移動時に暗号化することを想定している．評価について, XOM ではシミュレータを用いたワークロードの性能評価がなされているが, RTL 実装は行われておらず HW 規模に関する報告はない.

2.10. 機密性保護研究のまとめ

メインメモリ暗号化と内部アクセス制御の連携により，プロセスの 3 要素を暗号化した状態で実行可能なプロセッサとプロセス管理方式を提案，試作した．先行研究としてこのようなプロセッサの提案やシミュレータ評価は存在したが，プロセス管理を含むハードウェア実装を行い機能を確認したのは本研究が初めてである．ハードウェア規模は組み込みプロセッサの枠内におさまるものであり，ソフトウェア難読化の手法が適用できなかった組込み機器におけるデジタルコンテンツ保護強化の技術目標を達成できた.

さらに，提案方式をソフトウェア保護を依頼計算またはモバイルコードの保護に適用する際の課題について検討した．保護プログラムが実行されるターゲットシステムが保持するプライバシ情報に対するアクセスをサンドボックスにより制限することで，保護プログラムの情報資産とターゲットシステムの情報資産のいずれも脅威から守られる方式を提案した．本提案のプロセス保護は，既存 OS の資源管理と直交化されているため，サンドボックスによるアクセス制御と整合性が良い.

4 章に示すように，2022 年現在においては，メインメモリ暗号化はサーバ/PC 向けに広く実用化されているが，基本的な枠組みは本研究と共通するものである．ゲートの高密度化によりインパクトは相対的に小さくなっている．現在主流のマルチコア構成 CPU における問題としてコア間のキャッシュ一貫性制御があり，本博士論文においては詳細を説明していないが，著者がキャッシュ一貫性制御に関する方式提案を特許として公表している [54].

提案方式のプロセッサは，デジタル著作権保護をターゲットとして取組んだものだが，考察を進める中で, データオーナーとシステム管理者の間に不一致があるクラウドシステム（グリッドコンピューティング）やモバイルコードにも適用可能であることを示した.

3. 完全性保護技術の研究

3.1. 完全性保護研究の導入

本論文執筆時点の 2022 年においては IT システムがサイバー攻撃の主なターゲットだが，社会インフラの構成要素としてデータ計測または装置制御を行う組込み物理デバイスもまた，サイバー攻撃のターゲットになりうる．センシングデータおよび制御の完全性と通信の機密性・完全性を担保するための認証秘密鍵等が保護対象である．社会インフラを構成する組込みデバイスは，フィールドや都市部の工場に散在しているため，ネットワーク攻撃のリスクに加えて物理的攻撃のリスクも想定しなければならない．そして，装置の物理的保守を担当するサービスマンは通常制御機器のデータを直接操作する権限を与えられていないと想定されることから，インフラを運用するデータオーナーと物理的な管理権限との間に不一致があり，メモリセキュリティが必要となる．

組込み機器においても “Data at Rest”の保護に相当する，起動時にプログラムをチェックするセキュアブートは, PC やフィールドエンドポイントデバイスの保護メカニズムとして広く使用されている．だが，セキュアブートではプログラム等検証データサイズに比例した起動時間の遅延が避けられない [55]. 不揮発性メインメモリ(Non-Volatile Main Memory 以下 **NVMM**)の利用により，システムの電源オン―オフ時の起動時間を短縮することで頻繁な電源オン―オフを前提として運用が可能になる．これにより平均消費電力削減などの効果を得るノーマリーオフのコンセプトがあるが [56]，NVMM は起動時間短縮の手段として有望な一方で，物理攻撃による新たな脅威をもたらす懸念がある [57, 58].

機密性については，NVMM 上データはパワーダウンされた状態のデータが保持されるため，物理アクセスによるデータ漏洩の脅威は DRAM の場合と比較してより大きくなる．データやプログラムの完全性についても，電源オフ時の NVMM データの改ざんは大きな脅威である．DRAM におけるコールドブートアタックのシナリオでは，稼働中のシステムからメモリモジュールを引き抜いた後，別システム上で読出しを行うまでの間に多くのビットエラーが発生していた．このため DRAM では改ざん攻撃のシナリオは実行困難だったが，NVMM であればピンポイントの改ざんが可能となり，改ざん攻撃が現実の脅威となる．NVMM の社会インフラ機器への導入においては，これらのサイバー攻撃についての対策が必要となる．このようなメインメモリの機密性と完全性を保護する目的でプロセッサハードウェアについての先行研究が行われてきた．後述のように現在，PC/サーバ向けハイエンド CPU にはメイ

ンメモリ暗号化ならびに改ざん検証メカニズムを備えたものが存在する [26]． だが本研究発表時点から現在に至るまで組込み向けプロセッサにはメインメモリ改ざん検証メカニズムを備えたものは存在しない [59].

一方，仮想化技術は，近年組み込みシステムにも広く導入されている． Cloudvisor と TreVisor は，既存の CPU の仮想化機能を利用してフルディスク暗号化と改ざん検証機能を提供する [60, 61]. だが, これらのターゲットは主にクラウドシステムのストレージである. また, メモリへの物理的攻撃に対する CPU ハードウェアレベルの対策提案はあるが [59, 62], 変更なしの OS 動作が可能かつゲスト OS のメモリ上イメージ全体の暗号処理をベースとした改ざん検証機能を提供している CPU 製品は，著者の知る限り本論文執筆時の 2022 年においてもまだ存在しない.

完全性保護技術では，既存組込みプロセッサの仮想化機能を使用して OS イメージ全体を物理的な攻撃から保護する VMM を提案する．3.2 節では、ターゲットと要件を定義する．3.3 節では、提案方式のメカニズムを詳細に説明する． 3.4 節ではプロトタイプ実装と評価結果を紹介する．3.5 節では関連研究との比較や応用における課題を説明し，3.6 節で本章の結論を述べる.

3.2. 完全性保護研究のターゲットとゴール

本研究のターゲットは社会インフラを構成するフィールド機器である．サービスマンによる物理的管理権限の乱用ならびに物理的管理が十分でない野外機器に対する攻撃のいずれかにより，物理メモリが不正操作を受けたことによる機器の誤動作が，機器や人命に重大な損失を与えることを防止するものである.

本論文執筆時のサイバー攻撃では，攻撃者が身元を追跡されるリスクが低く，利用可能な通信脆弱性を持つ対象機器が多数存在することから，通信を経由した遠隔サイバー攻撃が攻撃の大部分を占める．だが，遠隔攻撃への対策が整備されるにつれ，物理的なメンテナンス操作を通じて予め攻撃の種となるバックドアを仕掛けておく攻撃が現実化する懸念があり，対策が必要である．また，制御システムにおけるサイバーセキュリティエンジニアリング標準においてはリスクベースの脅威対策がとられている．たとえ攻撃方法の困難度が高く，これまでに被害が顕在化していない攻撃手法であっても，想定しうるダメージが大きいケースについては対策が必要となる [63]．従来の手法では対策が困難な攻撃に技術対策を備えておくことでサイバーセキュリティエンジニアリングのリスク管理を容易にする効果が期待できる.

3.2.1. ターゲットシステムと利用シナリオ

NVMM システムは電源の再投入後，即時実行可能状態のイメージがメモリ上に保持されていることから，システムソフトウェアのフォールトや不意の電源断などの障害に対して，予備の実行イメージや健全状態のチェックポイントを保持しておくことにより，障害からの回復時間を短縮できる [64]．通常運用時のレスポンスタイムを短縮するため，そして通信断においても運用を継続するディスコネクテッドオペレーションを保証するために，運用ルールなどの情報はメモリに展開した状態で保持されるインメモリ処理を前提とする．運用ルールの内容は，一部の機器が異常状態に陥った場合の安全処理や，法的な安全規制を遵守するための判断アルゴリズムと規定である．したがって，運用ルールを格納するデータベースの改ざんがあった場合，安全処理ルールや法規制の逸脱というダメージが生じる．機密情報の漏洩もダメージとなるため，対策が必要となる．

本提案においては，このような脅威を緩和するために仮想化機能を備えた CPU 上で実行される**メインシステム**と（セキュア）**サブシステム**の２つの機能に分割された図 19 に示す構成を想定する．サブシステムは上記の運用ルールに関わるクリティカルな機能を提供することから，脅威対策として不揮発性メモリの物理攻撃対策機能を備える．サブシステムに対しては対策機能とのトレードオフとなるパフォーマンス低下は許容する．一方，メインシステムはパフォーマンスと多機能性を優先とするシステムである．性能を優先するため，メモリ保護機能の適用対象外とする． プラントなどの生産効率を最大化するための計算など，ス

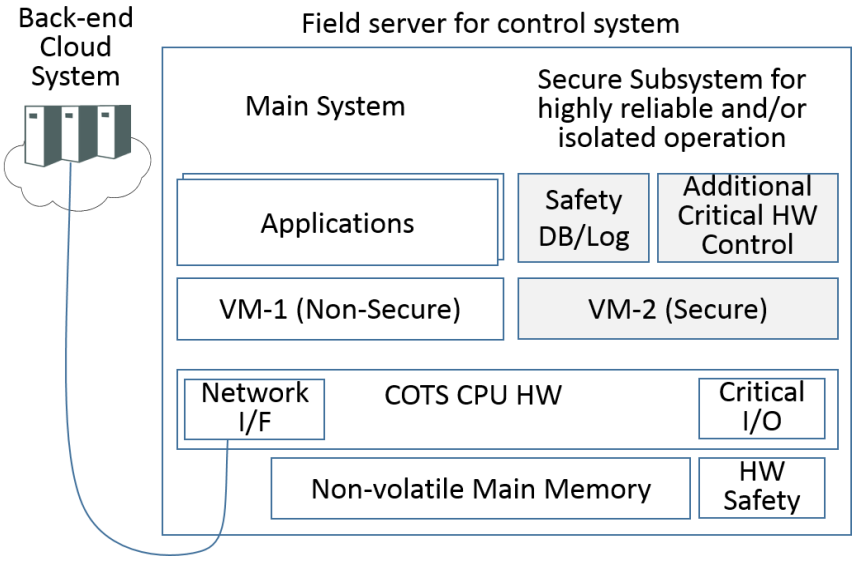


図 19　ターゲットシステムの想定構成

ループットを優先する処理は基本的にメインシステムが分担し，安全や法規制に関わる意思決定処理をサブシステムが補完するのが基本的な役割分担である．このほか，センシティブなデータログ管理や当該システムが直接管理するクリティカルなハードウェアの制御もサブシステムの役割である．システムソフトのフォールトや電源断を想定して，クリティカルなハードウェアには保守的な安全機能が備えられていることを仮定する．

この役割分担において，システムの通常運用時には，メインシステムが遠隔のクラウドシステムと連携して，生産効率を重視した最適制御を行い，サブシステムがそれを監視する．クラウドシステムとの通信断やシステムのフォールト，電源断などの異常状態が生じた場合はサブシステムがその後の処理を決定する．一例としてローカルネットワークに接続された他機器を含む安全処理を行う際に，安全処理や法規制の制約下で機器や生産中の仕掛品に対するダメージを最小化する意思決定を行うのが，異常状態におけるサブシステムの役割である．この場合，ローカルネットワーク接続された他機器の状態を考慮したダメージ最小化を計画する，いわば司令塔の役割を担う．一時的な電源断が生じた場合，その直後から動作を再開すべきか，それとも動作を停止すべきかの判断は他機器の状態や電源断の持続時間などの状況に依存する．大部分の機器は，自機器に限定した安全機能を備えていても，他機器の状態を考慮した判断機能を持たない．サブシステムは生産システムを構成するこれらの機器全体に対して，安全と法規制を最優先しつつ，生産再開におけるダメージを低減する機能を提供する．しかしながら，サブシステムもソフトウェアであるため，フォールトが生じた場合再起動までにサービス不能期間が生じる．このサービス不能期間については，各機器が備えるハードウェア安全機能が担う．この各機器に閉じた安全機能では，生産システムの仕掛品に対するダメージ低減などは考慮されない．以上がターゲットシステムの想定利用環境である．

ここでメインシステムに対する監視や，異常状態からの再開判断を誤らせる NVMM 物理攻撃が行われ，誤った判断が実行された場合，生産システムの停止や異常状態の対処誤りが生じ，資産の損害や人命に関わる危険が発生する懸念があり，物理攻撃対策が必要となる．

これらメインシステム，サブシステムのゲスト OS としては，Linux のような汎用 OS を用いる．制御システムには通信経由のサイバー攻撃対策も必要となるが，インターネットを通じた攻撃対策はデスクトップ PC などと共通要素が多いため，母数が多く実績のあるものが利用できる．

不揮発メモリ利用による上記の脅威シナリオが，IT システムであるデスクトップ PC やサーバでは問題視されていない理由について付記する．上記社会インフラでは，機器は無人環

境で運用され，運用ルールを判断する当該機器の判断異常が，直接複数の物理機器の動作に影響して大きなダメージをもたらす懸念がある．一方，デスクトップ PC は通常利用者による監視下で運用され，ダメージは情報に限定され，物理的なダメージに至るまでにはさらにいくつかのステップを経る必要がある．また，サーバ機器は物理的に厳重に管理された環境で運用されるため，社会インフラ機器と比較して攻撃の懸念は小さい．後述するように 2022 年現在のサーバ&PC 向け CPU においては，メインメモリ改ざん検証機能は単一のプロセスを対象としてメモリサイズが限定されており，仮想マシン全体を保護下におくことはできない．以下に説明する提案方式は，現在の CPU ハードウェアもいまだ提供していない物理メモリ攻撃対策に特化した対策を，専用の VMM により提供するものである．

3.2.2. ハードウェアと攻撃者に関する仮定

ターゲットシステムに関するより具体的な仮定を以下に示す．

A) CPU チップ境界の内側は安全．サイドチャネル攻撃はスコープ外とする（1.4 参照）

B) CPU は仮想化能力をサポートする汎用品（COTS: Common Off The Shelf)である

C) CPU はチップは 1-4MiB 程度の内蔵 SRAM(IM: Internal Memory)を持つ．

D) CPU は DMAC (Direct Memory Access Controller)を持ち，IOMMU (IO Memory Management Unit)の制約下で動作する．また，DMAC は暗号エンジンと統合されており，転送中のデータに暗号処理を行うことができる

E) 攻撃者は休止状態のシステムメモリに対する読出しと書換えアクセスが可能

F) 攻撃者は物理メモリに対する攻撃とネットワーク経由によるゲスト OS 内部へのユーザアクセス権限を組合わせることができる

A-D はハードウェアに関する仮定であり，E-F は攻撃者の能力に関する仮定である．B の能力は，マイクロコントローラ以外の組込み CPU においても一般的となりつつある [65]．C，D に相当する機能を持つ商用 CPU も存在する [66, 67]．内蔵 SRAM は攻撃者が操作できないメモリ領域として利用する．

F は安全性の検討において特に重要である．物理メモリの操作は任意の操作ができるという観点では非常に強力だが，有効な攻撃を構成するには操作された不正なメモリ内容を参照させて攻撃を発現させる能力も必要である．たとえば攻撃者が物理操作により OS カーネル内の重要な状態変数を書き換えておき，後からその状態変数を参照するシステムコールを発

行するユーザレベルリクエストを遠隔から送信することでシステム本来の動作に対する逸脱を，遠隔から制御できる．ネットワーク経由の攻撃対策の進展により，このようなメモリ物理攻撃を，ネットワーク経由の攻撃の梃子として用いることが想定される．物理メモリ攻撃の対策においては，このような能力を持つ攻撃者を想定する．

3.2.3. 制約条件

提案方式は汎用 OS を利用したサブシステムに対して，セキュア VMM が物理メモリの完全性と機密性を担保する機能を提供する．以下の 2 項目をソフトウェアに関する制約条件とする．

3.2.3.1. 単純性と OS 全体の保護

ターゲットにはメインシステムとセキュアなサブシステムが共存する．既存汎用 OS の実績ある遠隔攻撃対策を活用するため，メインシステムとセキュアなサブシステムのいずれについても Linux のような汎用 OS を仮定する（Linux 以外の排除を意図したものではない）．クリティカルな判断と重要デバイスの直接制御を担うセキュアサブシステムについては，プログラム，データ全ての領域に対して物理メモリの完全性保護を適用し，一定の性能低下は許容する．Intel SGX では Enclave と呼ばれる特殊プロセス（厳密にはプロセスを構成する共有ライブラリ）に対してハードウェアによる完全性保護機能が提供されているが，この機能は上限 128MiB にサイズ制限されている [68]．提案方式では，汎用 OS のイメージ全体を保護対象とする．

性能低下を緩和するため，メインシステムは物理メモリ完全性保護の対象外とする．ただし，攻撃を受けたメインシステムによるサブシステムに対する攻撃を回避するため，メインシステムとサブシステムの権限分離に必要な VMM の管理メモリに対しては，物理メモリの完全性保護対象とする．

3.2.3.2. リプレイアタックに対する堅固な安全性

メインメモリ安全性についての基本的定義は，直前のメモリ書き込み動作で書き込まれた値と同一の値が，次のメモリ参照において読み出されることと定義される [69, 70]．序論で述べたように，この性質を担保するには MAC が用いられる．MAC に対しては，最新のデータと MAC のペアを，既知のデータと MAC のペアにすり替える脅威がある．MAC を安全なメモリ領域に保持することでこの脅威は回避できるが，上記内蔵メモリのサイズでは OS 全体の MAC を格納するには不十分である．この問題の解決手段として Merkle ツリー手法が

ある [69, 70]．本提案も 基本は Merkle ツリーの手法を用いる．リプレイアタックには時間方向（書換回数）と空間方向（アドレス）の両方に対する考慮が必要である．

3.3. 提案方式のメカニズムとデザイン

3.3.1. 提案方式の概要

提案方式は仮想化メカニズムを使用してデータ検証を処理する． 主な特徴は，拡張ページテーブル（EPT: Extended Page Table）のページウォークと検証ツリーを連携させて取り扱うことである．この方式には 2 つの利点がある． 1 つは専用ハードウェアなしで，アドレス解決を契機として検証処理を行うことである．もう 1 つはメインシステムとサブシステムを隔離するページテーブルの改ざんを保護することである．メインメモリ全体をカバーする検証と暗号化には，Merkle ツリーをアレンジしたカウンターツリーを使用する [71]．ハードウェアキャッシュを拡張した検証ツリーでは，ツリーの階層が深くなり，性能予測が難しくなる問題があるが，ページテーブルをベースとした検証ツリーでは階層を 3-4 階層に抑えられる．以下，本節では VMM による検証ツリーを用いたメインメモリ改ざん検出を**物理メモリ保護**と呼ぶ．

図 20 に、ハードウェアおよびソフトウェア構成要素のブロック図を示す．ゲスト OS は VMM の管理下で運用される． VMM の実行イメージは起動時に CPU の内蔵メモリに全て格納された状態で実行されるため，実行時の改ざんからは安全である．起動時に TPM 等に基づくセキュアブートを適用することで，起動前の改ざんも排除できる．ゲスト全体にセキュアブートを適用した場合，起動時間の遅延が大きくなるが，実験結果に示す通り提案方式の VMM は 50KB 未満の小規模であるためセキュアブートを適用した場合でも起動時間の遅延はごくわずかである．

前章の機密性保護 CPU ではコンシューマ機器を対象としていたため，エンドユーザが自由に OS を変更できる想定の対策をとった．以下の提案は，社会インフラ機器をターゲットとしていることから，機器ベンダが使用する VMM を安全なものに限定している．

VMM は，ゲスト OS 間の分離とデータセキュリティ機能のみを担う．データセキュリティ機能は，①IM ページバッファ管理、②検証付き EPT ウォーク，③データ検証と暗号化で構成される． CPU は，仮想化をサポートし，暗号化 DMAC および SRAM IM を備える．ゲスト OS データは外部不揮発性メモリ（**EM: External Memory**）に配置される．EPT（ステ

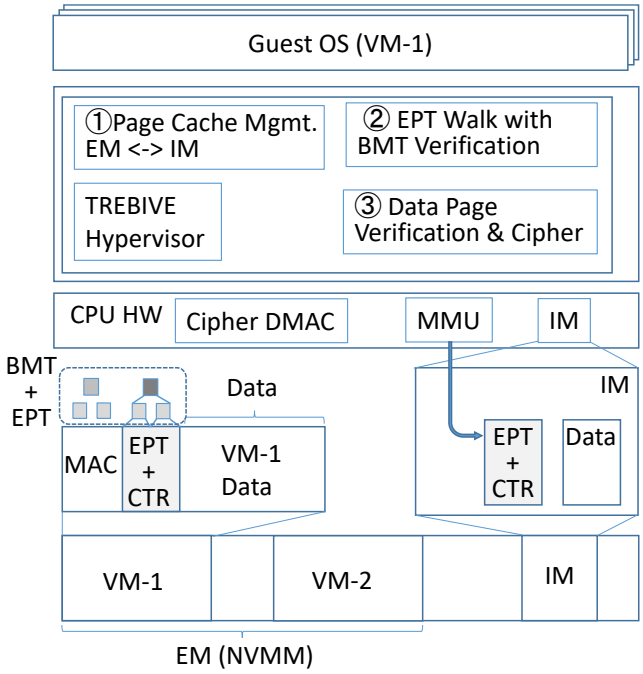


図 20　提案方式の概略構成

ージ２ページテーブルとも呼ばれる）と検証データである MAC およびカウンタアレイ(CTR)も NVMM に配置される．

EPT と CTR（PT + CTR）は連続したページに配置され，VMM によって連携管理される． ゲスト OS によるデータ参照により，データ（ページ）の解決のためのフォールトが発生したときの動作は以下の通りである．VMM は参照されたページテーブルを EM からをページ単位で IM にロードする．VMM は Merkle ツリー検証によりページテーブルの完全性を検証する．データページのアドレス解決に必要な全てのページテーブルのロードと検証が完了すると，データページにも同様の IM ロードと検証が行われ，ゲスト OS による処理が再開される． ゲスト OS が参照したデータのアドレス解決に用いる EPT およびページデータは，全て物理攻撃に対して安全な IM にロードされたデータが用いられる．VMM (TREBIVE Hypervisor)は，改ざん検証に特化した小フットプリントのプログラムであり，制御変数も含めて全てが IM 中に常駐する．CPU パッケージ内は安全であるとの仮定の下，検証処理は IM と CPU によりパッケージ内で完結することが，提案方式が外部メモリ（EM）に対する攻撃から安全であることの根拠となる．

このように，ゲストに対しては IM に格納された安全なメモリ参照が提供されるが，VMM は IM と EM を意識して検証操作を行う必要がある．特に注意が必要な点は，IM にあるデータは物理攻撃からは安全だが，上述のように CPU 内で実行されるプログラムからの参照に

対しては，適切なアクセス制御がなければ無防備なことである．ゴールで述べたように，物理メモリ保護を適用するサブシステムと保護なしのメインシステムが共存するとき，メインシステムから IM 上の検証済みサブシステムに対するアクセスを防止する必要がある．この隔離は VMM が管理するメインシステム用の EPT について，IM および EM 上のサブシステム領域をメインシステムの参照可能範囲から除外することで達成される．したがって，メインシステムのゲストデータは物理メモリ保護の対象外だが，メインシステムのアクセス可能範囲を定義する EPT は物理メモリ完全性保護の対象としなければならない．メインシステム向けの EPT 保護を低オーバヘッドで実現する詳細方式については後述する．

3.3.2. 暗号処理

図 21 にデータ保護に用いる暗号化手順を示す．提案方式においては，検証データとして直接 MAC を用いるのではなく，検証対象データの書換世代に相当するカウンタ(CTR)をパラメータとした MAC を用いる BMT (Bonsai Merkle Tree)を用いる［70］．MAC は CTR，ターゲットアドレス(GPA: Guest Physical Address = ゲスト物理アドレス)，ターゲットデータとゲスト仮想マシン毎に割り当てられる秘密鍵から計算され，CTR は書換回数に関する時間的リプレイ攻撃，ターゲットアドレスはアドレス空間に対する空間的リプレイ攻撃を防止するパラメータとなる．ページデータとカウンタの連接に対する SHA-256 ハッシュ値 H を生成する．このハッシュ値を AES-XTS モードで暗号化して MAC を生成する．AES-XTS モードには tweak と呼ばれるパラメータがあり，同一データを同一秘密鍵により暗号化した場合でも，tweak が異なっていればリプレイアタックができないことが暗号学的に担保されている．データ本体に対しても同様のパラメータで，AES-XTS モードにより暗号化する．おお，データ本体に対する機密性が不要の場合，暗号化処理を MAC にのみ適用し，データ本体に対する暗号化は省略することができる．

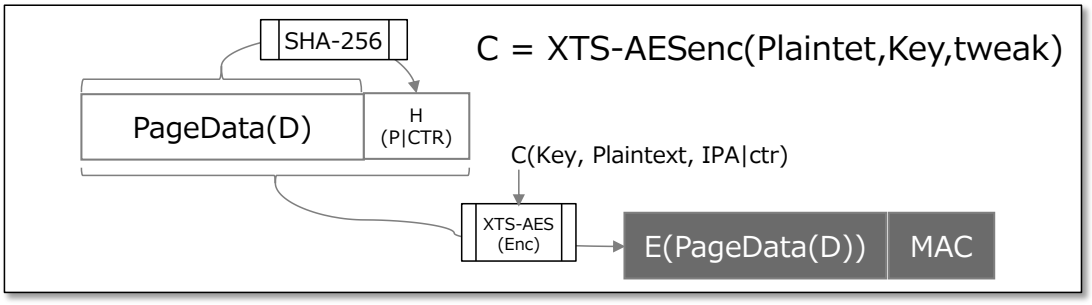


図 21　ページデータの暗号処理

3.3.3. 検証ツリーの構成と管理

このパラメータを用いたデータ検証を図 22 にそってより具体的に説明する．ここで灰色で示された EPT は，上から下にレベル１，レベル２，レベル３と階層的に構成されるページテーブルを表す．ページテーブルは，ページテーブルエントリ (PTE: Page Table Entry)の配列であり，それぞれの PTE が下位のページテーブルまたはデータページの参照先アドレスと制御情報を保持している．ARMv7 アーキテクチャにおいて PTE サイズは８バイトである．

各 PTE 毎に参照先ページの世代を管理する 8 バイト(128 ビット)のカウンタが割当てられ，カウンタの配列は EPT の１ページと同一サイズに格納されている．図ではカウンタの配列は緑色で表現されている．PTE と対応する CTR に対する MAC がメモリに保持されている SHA-256 を用いた場合，MAC は 32 バイトとなり，全体を格納するには CTR の 2 倍のサイズが必要となる．

保護されたゲスト OS 実行で最初に行われるのは，データ参照のためのアドレス解決である．実行に先立ってゲスト OS に対応する Root MAC を VMM が読込んでいる．Root MAC は最上位のレベル１　EPT と CTR 全体に対する MAC であり，ゲスト OS の停止中は TPM などの安全な不揮発性ストレージに保存されている．

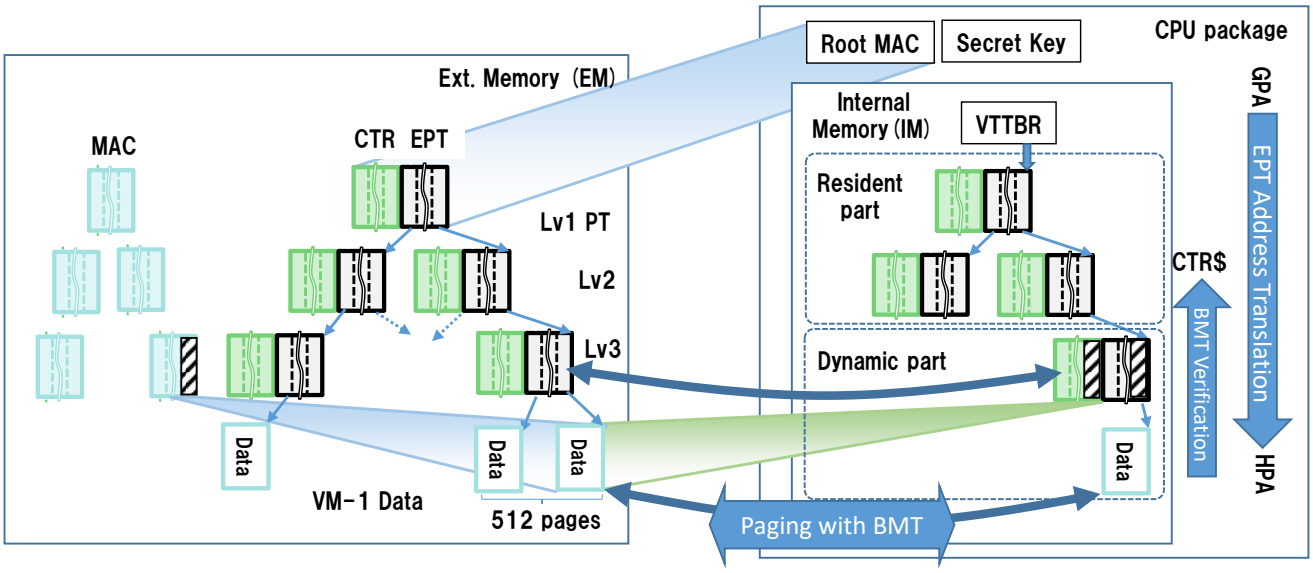


図 22　EM,IM におけるページテーブルと検証データ

ゲストのデータ参照では GPA に対してホスト物理アドレス(HPA)への解決が行われる．最初に最上位のレベル１ EPT と CTR が IM 内のバッファメモリに読込まれる．このページ単位のデータ転送は DMAC により行われ，上記のハッシュ及び MAC 計算と暗号化データの復号は DMAC と連携した暗号エンジンにより処理される．SHA,, AES の暗号計算は 20-100 クロック程度のレイテンシで実行可能であり，パイプライン処理を行うことでページデータ全体の転送時間に占めるペナルティは小さく抑えられる [66].

レベル２ EPT の読み込みにおいては，参照先ページテーブルを示すレベル１PTE に対応するカウンタ（IM に読込み済み）と，外部メモリに配置された対応する MAC(期待値)を用いる．レベル１の場合と同様，レベル 2 ページテーブルと対応するカウンタページを IM に読込み，参照元の PTE に対応するカウンタと参照先アドレスを tweak として，読込んだレベル２ページテーブルとカウンタのセットに対する MAC 値を計算し，外部メモリ上にある参照元 PTE に対応する MAC(期待値)と比較して，一致すれば検証は成功する．

先取りして書換時の処理を簡単に説明する．レベル２ページテーブルおよびカウンタが書換え後に外部メモリに書き出されるときに参照元 PTE に対応する，上位のレベル１カウンタのインクリメントが行われたうえで MAC が再計算されて暗号化データとともに外部メモリに保存される．したがって同一のページテーブルに対して複数回更新が行われた場合でも都度異なる tweak が用いられ，リプレイアタックは防止される．

レベル３ページテーブル，そしてデータの読込み時にも同様の IM 上に読込まれたカウンタと外部メモリ上 MAC(期待値)を利用した検証が行われる．ページテーブルは枝の多いツリー構造であるため，この手順を適用することにより，ゲスト OS に与えられるメモリ空間(IPA)がどれだけ巨大なものであったとしても，ページテーブルにより管理可能であるかぎり，検証ツリーによる改ざん検証を行うことが可能である．検証ツリーによる改ざん検証では，対象メモリ空間内のデータはハッシュ関数のツリー操作を通じて最上位の Root MAC に反映される．よって，不正変更はたとえ 1 ビットの変更であっても，読込み時に検出することが可能となる．

ページテーブルを改ざん検証に使うことで，不連続，あるいは実メモリが未割当の領域も管理可能となる．ASLR (Address Space Layout Randomization)は，バッファオーバフロー等の脆弱性検出と，既存コードの特定番地に対する分岐による攻撃に対する有効な緩和策であり，カーネルコードにも適用されている [72, 73]．ASLR ではコード配置先のランダム化により，必然的に未使用のメモリ領域が生じるが，ページテーブルを用いれば VMM がゲスト仮想マシンに対して実使用する領域のみメモリを割当てることが可能であり，カーネル

ASLR が利用されるケースでも効率的に実メモリを利用できる．ページテーブルエントリにはソフトウェアが使用可能な予備ビットが存在するため，予備ビットを利用して当該ページエントリの使用/未使用を管理することができる．なお，ページテーブルエントリには有効無効を示す Valid ビットが存在するが，提案方式においては以下に述べる未検証ページの検出に用いるため，未割当ページの管理に用いることはできない．

本節ではページテーブルと検証ツリーの関係と読込み手順の概略を説明した．有効・無効ビットの操作を含む詳細手順を次節で説明する．

3.3.4. 未検証ページの検出と読込み時のページテーブル操作

ページテーブルを利用したゲスト OS による未検証ページ参照の検出には，Chen らが Overshadow において提案した方式 Shadowing をアレンジして用いる［74］．図 23 にそってページデータ読込み時の操作を説明する．未検証状態のページデータは EM に配置されている．ページを参照する PTE3 を含むページテーブルは，ページデータ参照に先立って IM 内に読み込まれ，検証が完了している．EM に配置されているページテーブルのページテーブルエントリは，Valid ビットが無効となっており．よって，ゲスト OS が Data を参照するとフォールトが発生し, VMM によるデータ読込みと検証処理が行われる．この時点で, PTE3 の参照先は EM 上のアドレスを示しているが，読込み完了後に PTE3 の参照先はデータが読み込まれた IM 上のデータバッファのアドレスに書換えられる．そして, Valid ビットを有効化する．アーキテクチャに依存して，Valid ビット操作後に TLB フラッシュなどの操作が必要となるが，この状態でゲスト OS の実行を再開すれば，EM 上データに代わって IM 上データがアドレス変換され，ゲスト OS によるデータ参照が可能となる．

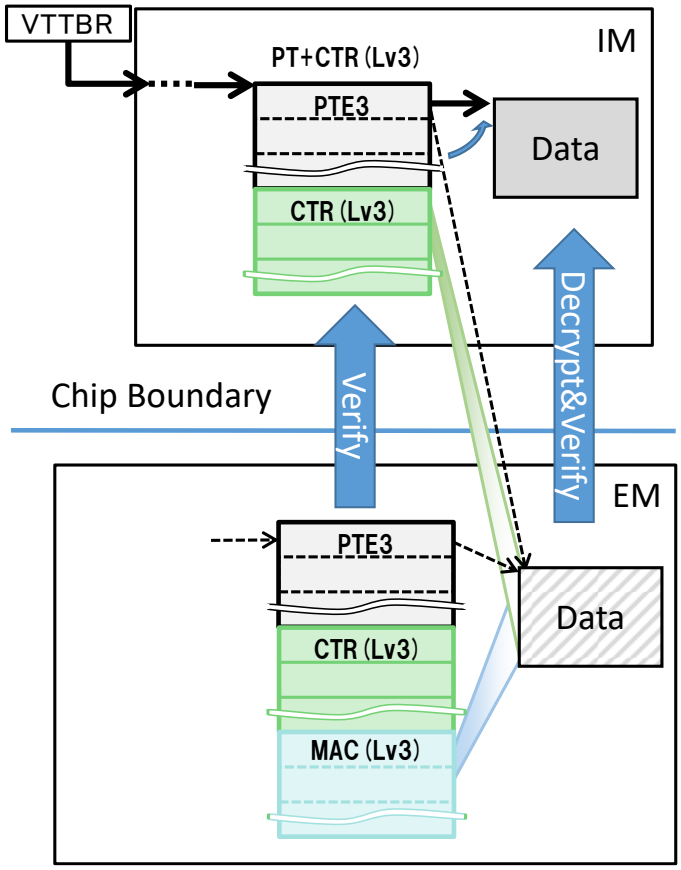


図 23　ページの読み込み動作

説明順序が前後したが，レベル１，レベル２，レベル３のページテーブル参照時にも同様に上位 PTE の参照先を下位ページテーブル読み込み先 IM アドレスに書き換える処理と上位 PTE の有効化処理が行われる．

ページデータが不要となり IM 上バッファからフラッシュされる際には，フラッシュされるページに対応する IM 内の上位カウンタのインクリメントと MAC 再計算が行われ，暗号化されたページデータと MAC が EM に書き込まれる．ページデータの暗号化と MAC 計算は DMAC 連携したハードウェアにより処理される．ページテーブルのフラッシュにおいては，ページテーブルに加えてカウンタも MAC 計算および暗号化の対象となる．

なお，元々PTE が参照していた EM 上アドレスは，IM 上のページアドレス退避テーブルにいったん退避され，ページデータが不要となり IM 上バッファからフラッシュされる際に対応する PTE に書き戻される．もしフラッシュに EM 上の配置先が変更される場合は，新たに EM 上のアドレスを PTE に書き込む．退避テーブルは IM 上バッファサイズに対応するエントリをあらかじめ静的に確保しておく．

3.3.5. 非保護ゲスト OS に対する EPT 管理

3.2.3 節に述べたように，本提案では物理メモリ保護ありのサブシステムと，保護なしのメインシステムが共存し，保護なしのメインシステムは低オーバヘッドで実行できることをゴールに置いた．ただし，3.3.1 で触れたように，メインシステムによる IM 上データアクセスを防止するためにメインシステムのアクセス範囲を制限する EPT の改ざんは防止されなければならない．これを両立させるページテーブル構成を図 24 に示す．ここでは，IM 上ページテーブルからの EM 上データの直接参照と，ラージページの２種類の手法を組み合わせることで，EPT に対する改ざん検証を行いつつ，非保護 OS によるデータ参照時のフォールト発生を回避している．

最上位レベル１のページテーブル PTE1 の読み込みは，保護ありのゲスト OS と同様である．PTE1 が参照するレベル２ページテーブルの読み込みも同様に行われる．だが，データページを参照するレベル２ページテーブルはカウンタを伴っていない．レベル 2 のページテーブルエントリ PTE2 が参照するデータページに対しては検証処理を行わないため，カウンタは不要だからである．また，PTE2 はラージページであり，１エントリについて 2MB の外部メモリデータを参照できる．各ページテーブルには 512 個の PTE を格納できるため，１ページのレベル２ページテーブルにより 1GB の外部メモリ領域の参照が可能となる．レベル１ページテーブルも同様に 512 個の PTE を持つため，２階層のページテーブルだけで最大 512GB のメモリ領域をゲスト OS に提供できる．

前節に説明した物理メモリ保護ありの OS では，データページについても読込み時の改ざん検証を行うために，データページを参照する PTE2 は EM 上では無効状態としていた．それに対して，保護なしのゲスト OS についてはデータページを参照する PTE2 は EM に配置された時点であらかじめ Valid ビットが有効に設定されている．したがって，PTE2 を含むレベル２ページテーブルが読み込まれ，参照元の PTE1 の Valid ビットがセットされた時点で，ゲスト OS は検証なしで外部メモリを参照できる．レベル２については読込み後のページテーブルエントリ書換は不要である．注意すべき点は，保護ありゲスト OS との隔離を担保するため，保護なしゲスト OS の PTE の参照先に IM が含まれないよう，保護なしゲスト OS の EPT が VMM により適切に管理されなければならないことである．

このように，提案方式ではいったんページテーブルの検証読込みが完了すれば保護なしゲスト OS によるデータ参照は，EPT のアドレス変換オーバヘッドのみで実行可能であり，保

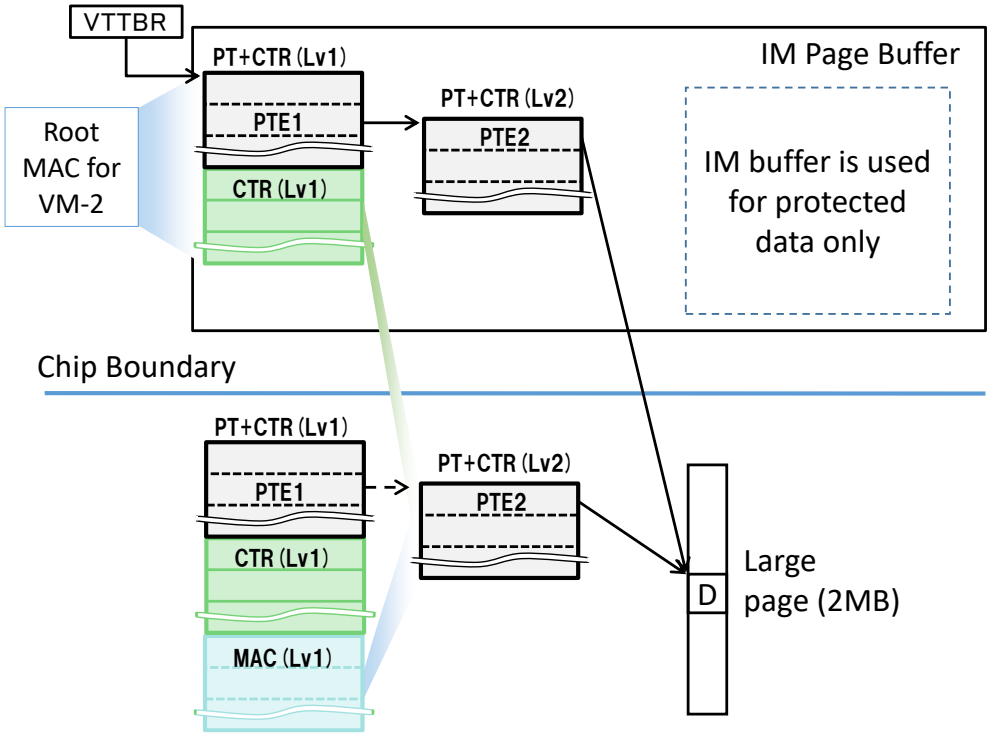


図 24　非保護ゲスト OS に対する EPT 管理

護なしゲスト OS の実行時オーバヘッドを一般の仮想化と同等に抑えつつ，物理メモリ保護あり OS との安全な共存が可能である．

3.3.6. EPT 管理とデータ共有の制約

提案方式では，EPT のアドレス解決を契機として，書換回数と IPA に対するリプレイ対策を含むツリー検証を行っている．したがって，同一データを参照する EPT エントリであっても，ゲストに対する IPA 割り当てが異なる場合は個別に EPT エントリを割り当てる必要がある．また，慎重に管理すれば更新のないリードオンリーデータを複数のゲスト OS 間で共有することも可能なはずだが，いずれかのゲスト OS が書換を行うデータについてはゲスト OS 間の共有と改ざん検証は両立しない．この場合や通信データバッファ，ハードウェアレジスタなどは，非保護 OS の EPT 管理で示したようにゲスト OS が直接参照を行う形で EPT を管理する必要がある．

3.3.7. メモリオーバヘッドの推定

4GB のゲストメモリスペース全体を保護するためのメモリオーバーヘッドサイズ見積もりは以下のとおりである．　CTR および MAC サイズは、lv3、lv2、および lv1 でそれぞれ 8 +

32 MB、16 + 64 KB、および 4 + 4KB となる． 上記の合計は 4GB 領域に対して約 1%となる． 各 CTR アレイのサイズは 4KB で，カウンタ長は PTE サイズと同じ 64 ビットである．MAC エントリのビットサイズは 256 である．

3.4. 実装と評価

3.4.1. 実験セットアップ

プロトタイプ実装の概要を表 8 に示す．評価は、HW 仮想化サポート機能を持つ ARM Cortex-A15SoC 評価ボードで行った． VMM は，4.2 KSLOC のオリジナル実装である． プロトタイプには後述の多くの統計データ処理が含まれるため、約 200KB の作業領域が必要となるが，機能上必要な作業領域はより小さい．

これまでのターゲットおよび提案したメカニズムと，評価の間にはいくつかの差異がある．表 9 にそれらを要約する． 項目 1〜5 は，評価結果に影響がある項目である． IM バッファメモリとしてオンチップ高速 SRAM を想定しているが，実装では EM と同じ DRAM を使用している．DRAM 使用により実行時間については追加のオーバーヘッドが発生するが，ページング回数の評価には影響はない．データ転送と暗号処理は DMAC によるストリーム処理を想定していたが，実装では SW 処理である． 代表的不揮発メモリ MRAM と DRAM のレイテンシはおおむね同等である．ゲスト OS としては Arm 用の Linux を無改造で使用している．なお，統計情報や測定に関して VMM とコミュニケーションするための専用 LKM をいくつか導入している．

表 8　基本データ

Hardware	ARM Versatile Express with CoreTile Express A15x2 A7x3
Memory	DRAM 2GB (DDR2 400MHz 32bit bus)
CPU	Cortex-A15 x 1 with 1MB L2 U$
Guest OS Image size	Linaro Linux stable 3.14 for vexpress 2015.03 1512KB
VMM	43.2K with XTS-AES,　4.2K SLOC

評価パラメータをに示す. 検証には 64 ビットカウンタのカウンタベースの MAC ツリーを使用する． 実験時間を短縮するために，ターゲットメモリサイズは 32MB または 128MB とした.

表 9　想定と評価実装との差異

	実システムにおける想定	本評価における実装
① IM page buffer	On-chip fast SRAM	DRAM (same as EM)
② Data Transfer and Cipher	DMAC with cipher engine	Full software
③ Main memory	MRAM	DRAM
④ Verification tree building	pre-built at last hibernation	Build at specified point of evaluation
⑤ Linux launch	Hibernation wakeup	Cold Boot
⑥ VMM boot	Secure Boot	Normal boot
⑦ Verification key	Load from TPM	Embedded in VMM image

表 10　共通評価パラメータ

	Common evaluation parameters
Protection type	Non protect (PT+CTR only)/Protect(PT+CTR and Page)
Verification method	BMT (simple 64 bit counter)
Table protection	Verification only (pseud MAC/HMAC)
Page (data) protection	Verification and cipher (pseudo Cipher/XTS-AES with SHA256)
Page replace	pLRU with priority control
Target memory size	32MB/128MB

表 11 測定項目（Linux ブートにおける測定結果を含む）

	imem-in		imem-out		Imem WB		pLRU flash	TLB flash	time (sec)	estimated time			
test_name	table	page	table	page	table	page			measured	pseudo	AES-XTS	DMAC 10M	DMAC 100M
DT 4MB	7	3182.0	0.0	2172.0	0.0	1984.3	5.0	2172.0	142.8	-	-	-	-
EMU 4MB	7	3203.3	0.0	2193.3	0.0	2002.7	5.0	2193.3	20.0	144.5	1853.3	22.2	20.2
DT 2MB	9	19607.3	2.0	19109.3	2.0	7347.7	92.3	19111.3	749.9				
EMU 2MB	9	18928.3	2.0	18430.3	2.0	7100.7	88.0	18432.3	119.8	741.0	9278.9	130.5	120.9

ページバッファは 1MB（256 ページ）と 4 MB（1024 ページ）とした． テーブル管理と実験結果の解釈を簡素化するために、VMM フットプリントおよび作業領域は無視している．ページバッファのキャッシュ置換は pLRU アルゴリズムを用いており，データページに対してページテーブルの保持を優先する優先制御を適用している．優先制御がない場合，データ読込み時にページテーブルが破棄される傾向があり，テーブルが破棄された結果，テーブルに参照されるページデータの大量パージが発生することを抑制するためである．

3.4.1.1. 測定項目

測定項目を表 4 に示す．測定項目とともに Linux Boot 計測時の測定値も示されている． 測定期間開始，終了はゲスト Linux 上のアプリケーションから LKM を通じた HVC (Hypervisor call)発行によって指定される．実行時間は、CPU システムカウンターを使用して VMM で測定，保存される． その他のアイテムは、対応するイベント処理時に VMM でカウントされ，VMM 内で保持される．ワークロードごとに測定を 3〜20 回繰り返し，平均を測定値として採用した．

マイクロベンチマークと DB の測定では、外れ値の除去を適用した． Linux バックグラウンドプロセスの相互干渉による実行時間増加の影響を回避するため，9 個のデータから上下の外れ値 2 個または 4 個のデータから 1 個の外れ値除外を適用した． 実験全体では Linux のバックグラウンドプロセスの影響を織り込むデザインをとっているが，過度の外れ値は一部の測定特に長期間の実行を要するこれら測定の結果をゆがめるため，過度の外れ値を除外するデータ処理を適用した．

3.4.1.2. エミュレーションと測定結果に基づく性能推定

実験に使用したハードウェアは想定と異なり，暗号 DMAC 機能がないために暗号データ処理の性能が低い．実験時間を短縮するため暗号処理は疑似暗号(XOR)で行った．疑似暗号

による実行時間は，想定ターゲットにおける実行時間とは一致しないが，暗号処理，ページデータ転送，および TLB や CPU キャッシュに対するパージなどの統計値から想定ターゲットに関する性能指標が得られる．

予備実験において，ページ転送が実行時間の大半を占めることが判明した．そのため，時間のかかる実験では，データ転送をスキップして代わりにページキャッシュ操作の回数を記録した．以下，データ転送をスキップした結果は，推定データ転送オーバーヘッドを含む「エミュレーション」と呼ぶ．ページ置換および TLB 無効化などの関連する管理は、データ転送と暗号化を除いて想定ターゲットの場合と同一である．エミュレーションで得られた統計値に基づいて暗号 DMAC を利用した場合の性能推定結果を後述する．

3.4.2. 測定結果

3.4.2.1. データ転送と暗号処理

バルクメモリ転送と暗号処理のパフォーマンスを評価するため，暗号を使用した DRAM-DRAM データ転送性能を VMM 環境で測定した．結果を表 12 に示す．memcopy によるソフトウェアデータ転送速度は 1.28MB /秒であり、Linux 環境で得られた 1.07 MB /秒の LMbench の結果とおおむね一致した． ただしこの値は DDR2 DRAM の公称値に基づく予想よりもはるかに遅い．考えられる理由の 1 つは，評価基板では DRAM が CPU からコネクタで接続される別基板に配置されていることである．XTS-AES を使用する SHA256HMAC は、前述のように、メモリ転送よりも 10^2 オーダーで遅くなる．Linux LKM を使用してボード上で測定された DMAC データ転送速度も DMAC 適用時の性能を議論するため結果を示す．

表 12　データ転送と暗号処理

		bytes/sec
Data transfer	memcopy	1,275,513.5
Pseudo cipher	MAC only	260,416.9
	Cipher&MAC	164,473.9
Cryptography	SHA256 only	29,959.3
	AES-XTS & SHA256	11,565.4
HW	DMAC	10,055,000.0

3.4.2.2. Linux Boot

Linux の起動時間を測定した． あわせて OS イメージ全体を暗号化した状態で実行を開始し，議事暗号処理を行ったうえで正常にブート処理が完了することを起動メッセージの表示により確認した． 測定は，カーネル命令の実行の開始から対話型シェルの初期化の終了までを HVC により指定した． ページバッファサイズとデータ転送の有無を組合わせた 4 つの測定結果を表 5 に示す．DT と EMU は、それぞれデータ転送とエミュレーションを意味する．

測定項目には，ページ転送統計，pLRU および TLB 操作，実実行時間とこれらの統計値から算出された推定時間を示している． 推定時間はエミュレーション時間，合計ページ転送時間，およびパフォーマンスデータに基づいて計算した． 測定結果は 3 回の平均である． バッファサイズが同じ場合，データ転送ありの場合のページ統計数および関連する操作のデータ転送結果は，データ転送なしのエミュレーションの結果と一致している．図 25 は、測定時間と推定時間を標準偏差（SD、赤い線で表す）つきで示している．DT の測定時間と EMU の推定時間は非常に近いことがわかる．この結果に基づき，この後に示す実験では測定時間を短縮するため，データ転送を省略したエミュレーションに基づく推定を採用した．

図 25 は、データ転送ありの場合のブート時間測定値（DT: Data Move Time），データ転送なしの測定値からソフトウェア処理によるデータ転送の測定値を外挿した値（EMU: Emulate Time），そしてソフトウェア処理によるデータ転送代わりに 10 MB /秒の暗号データ転送処理を持つ DMAC を使用した場合の外挿値（DMAC10M）を表している．標準偏差

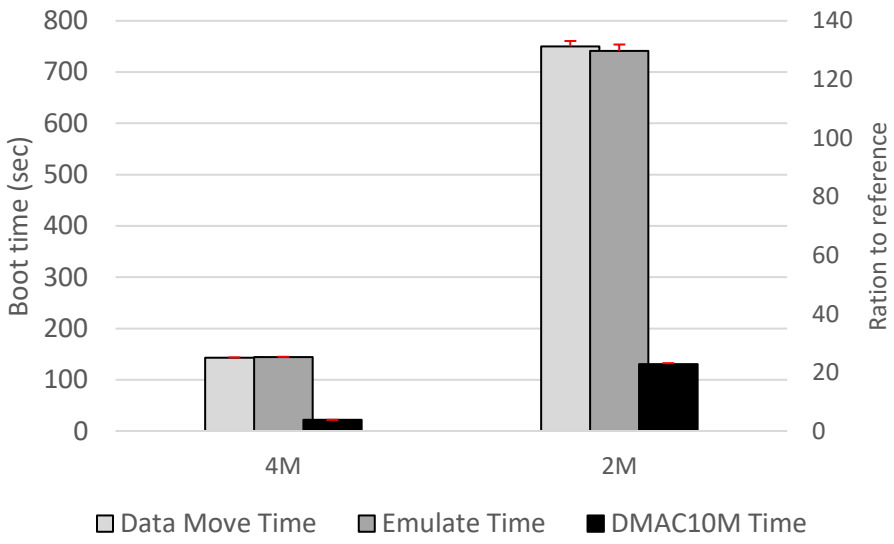


図 25　Linux ブート時間

は赤のバーで表しているがブート処理の測定ではおそらくバックグラウンドタスクがないため非常に小さい．DT の実測値と EMU の外挿値は非常に近いことがわかる．この結果に基づき，この後に示す実験では測定時間を短縮するため，データ転送を省略したエミュレーションに基づく推定を採用した．保護なしのゲスト OS 実行と比較するため，ページングなしの VMM での起動時間の測定をデータ保護に対するコントロール測定とした．この測定は以下，st2-only と呼ぶ．St2-only の起動時間は 5.8 秒である．グラフの右側の縦軸は、st2 のみのコントロールに対する比率を表している． 4MB および 2MB のバッファの疑似暗号には、それぞれコントロールに対して 25 倍および 120 倍の実行時間となった．DMAC10 MB の推定時間は、約 3.8 倍と 22.2 倍になった．4MB のバッファメモリと DMAC を仮定すれば，メモリ保護を Linux ブートに適用してもオーバヘッドは保護なしと比較して 3.8 倍程度にとどまる．

3.4.2.3. マイクロベンチマーク

システムコールとプロセススイッチのオーバヘッドをマイクロベンチマークを使用して測定した． アプリケーションと OS との一般的な相互作用で生じるオーバーッドの測定を意図している．測定手順を図 26 に示す．他のアプリケーションワークロードについても同じ方法を用いている． ハイパーバイザー呼び出しによってトリガーされた測定間隔の開始時に VMM は統計をリセットし，ページキャッシュをフラッシュして測定を開始する．測定期間中，VMM は未検証のページの参照にともなうページキャッシュの更新を行い，統計を更新する． 測定終了時に，HVC を発行し VMM は統計を記録して結果を報告する． これらのハイパーバイザー呼び出しはワークロードループの開始と終了に配置されるため、ターゲットワークロードをわずかに変更する必要がある．

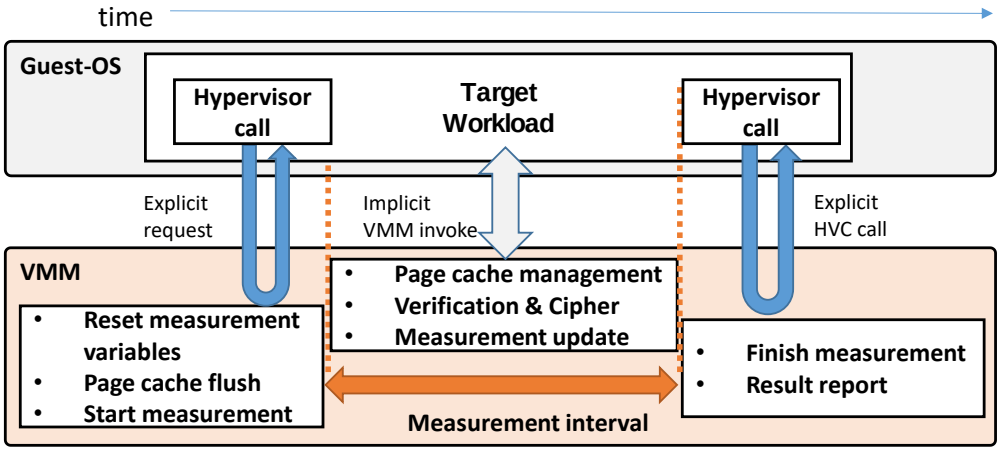


図 26　測定手順

ターゲットワークロードとして Unixbench5.0 [75]の syscall と context を用いた．syscall は，dup，close，getpid，getuid，および umask で構成される一連の 5 つのシステムコールを発行する． この操作において参照されるデータには，ユーザースに配置されるライブラリと対応するカーネル部分のワーキングセットが含まれる． この測定の目的では，ワーキングセットの大まかなサイズを把握することを意図している． また，ゲスト OS のプロセススイッチでは，ページテーブルのスイッチが必要となるはずである． 拡張ページテーブルを利用した検証操作がある場合，プロセススイッチによりどのように影響を受けるかの測定を意図している．

システムコールとコンテキスト測定の両方で、4MB のバッファの場合にページアウトは観察されなかった． 測定のループ数が少ない場合、ページインの最小数はそれぞれ 195 と 193 である．これらの数値は，これらのワークロードに必要な最小限のフットプリントに対応すると考えられる．1 MB のバッファ内には 256 ページがあるため，各ワーキングセットをバッファに収容できるはずである．ただし，ループ数が多い場合の実験ではより大きなページング回数が観察された．

図 27 にループ数が多い場合のページ回数と実行時間のグラフを示す． 実行時間は，エミュレーション結果に基づく DMAC 10M の推定値である． ループカウントはそれぞれ 100,000 と 1,000,000 の結果を示す． 4 MB のバッファの場合，各ワークロードで 400 ペー

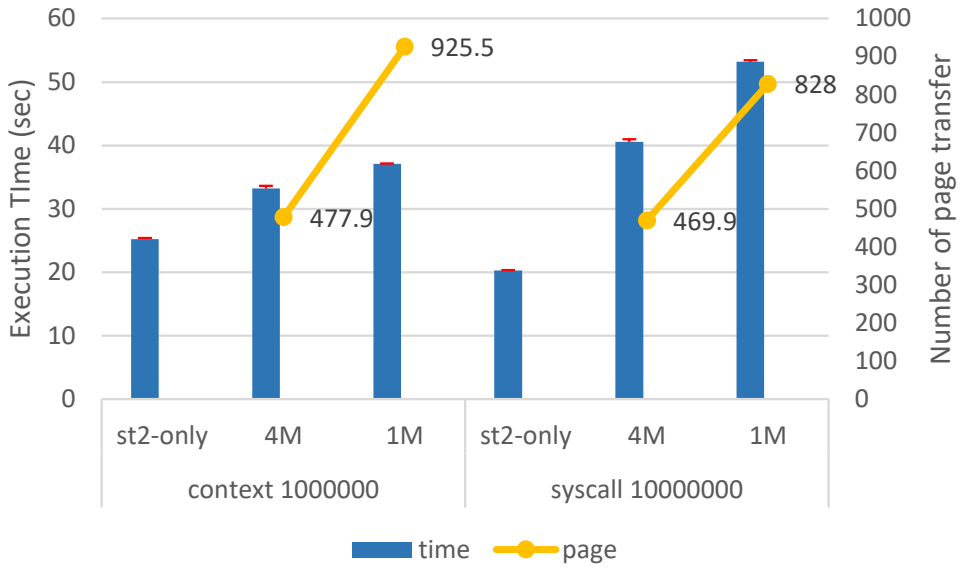


図 27　マイクロベンチマークにおける実行時間とページング数

ジの転送が発生している． ページ転送がループの都度発生する場合は転送の数はループ数と同じかそれ以上となるはずだが， 測定値はループ回数を大きく下回る． おそらくこのページング発生は周期的または何らかの外部イベントに応じて実行される Linux バックグラウンドプロセスとの干渉によるものと思われる．

このような並行処理されるプロセス実行の影響の受けやすさについて，2 つのワークロード間に違いがある．st2-only の場合，すなわちページングなしの場合は context は syscall よりも時間がかかっているが，4M および 1M のバッファの場合は syscall のほうが実行時間が長い． これは，ページ数とフットプリントサイズでは説明できない． 考えられる解釈としては，context ワークロードには、プロセススイッチを行うため，ページングなしの場合でも TLB 無効化を呼び出す処理が多数含まれているため、他のプロセスの介入の影響を受けた場合でも実行時間の増加は相対的に小さい．一方，ページング処理がない場合，syscall ワークロードには自発的なプロセススイッチは含まれていないため，ページングに伴う TLB 操作により実行時間が大きく増加している解釈が考えられる．

3.4.2.4. データベース

データベース（DB）の評価は、安全なサブシステムでの意思決定およびデータロギングの応用に対応する．SQLite 組み込みデータベース [76]を使用して，2 種類のデータベース操作を評価した． 1 つは、ランダムに選択されたキーを使用した DB 上の 1 つのレコードに対するクエリ（select）に対する応答時間である．もう 1 つは，DB へのレコードの insert の応答時間である．図 29 は 4MB のバッファケースでの選択と挿入の結果を示している． 実行時間はエミュレーション結果に基づく DMAC 10M の推定値である． 6 種類のサイズの DB（1，10，20，40，60，80 MB）は、ランダム化されたキーを持つ郵便番号データから生成した． SQLite はインデックス作成に BTree アルゴリズムを採用しているため、必要な時間とスペースは DB サイズに対して O（log n）となるはずである．実行時間と転送されたページ数の両方についてこの性質が現れていることが図 29 から読み取れる．

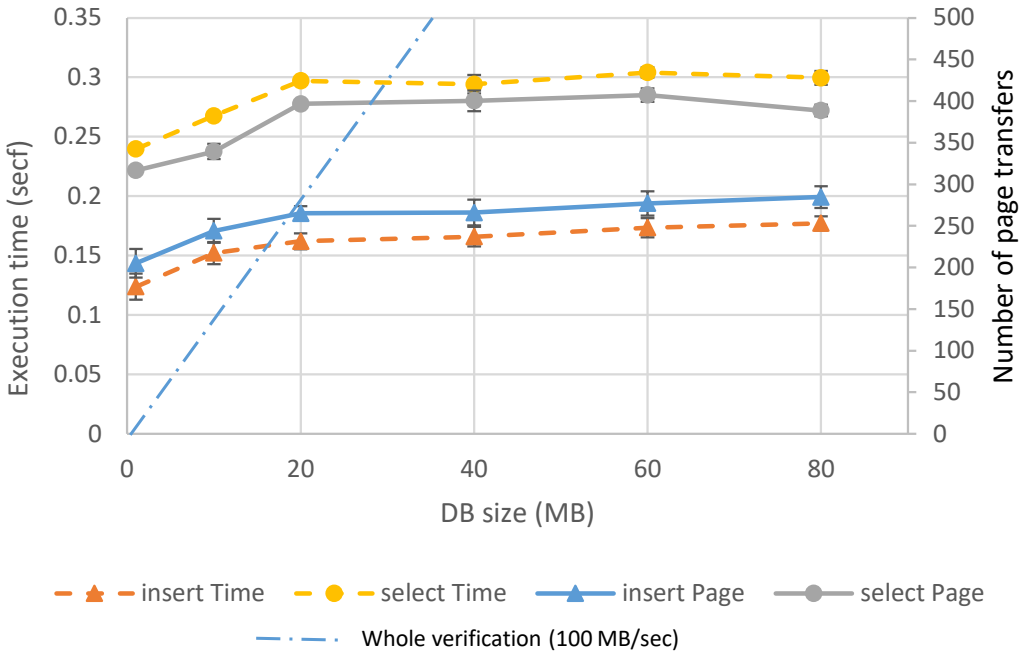


図 29　4MB バッファにおけるデータベースクエリ応答時間

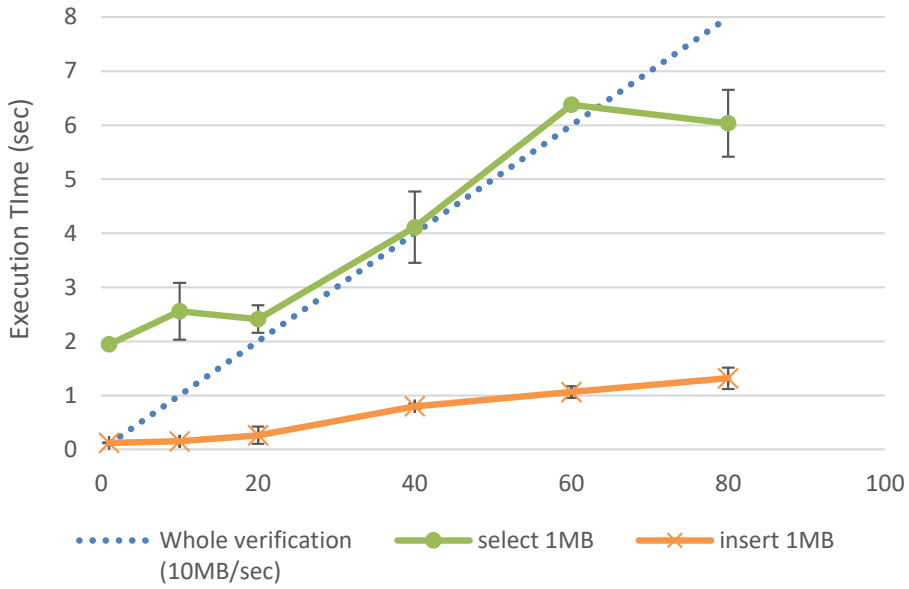


図 28　1MB バッファにおけるデータベースクエリ応答時間

st2 のみの参照実行時間は，すべての場合で 1 ミリ秒未満である． 参照に対するオーバヘッドは 100〜1000 倍に達し非常に大きい．ただし，既存のシステムで使用する直前に DB イメージの整合性を検証が必要な, DB サイズに比例した整合性チェック処理の時間がかかる．たとえば，10MB /秒の暗号 DMAC を使用したサイズ 10MB DB の検証には 1 秒かかる．提案方式によるメモリ検証処理では上記のようにサイズに対して応答時間は log(N)オーダーであり，データベースサイズが大きくなれば提案方式が有利になる．一点鎖線で示した線が 100MB DMAC によるデータベース全体の検証時間に相当する．データベースサイズが 30-40M を超えた領域では提案方式が有利になることがわかる．加えて，提案方式ではデータベースイメージの完全性検証だけでなく，関連する全てのデータとプログラムについても完全性検証が行われる．

1MB のバッファの結果を図 28 に示す．select クエリ処理に必要なページ数（約 300 ページ）は 1M バッファからあふれるため，4MB の場合と比較して非常に長く実行時間は 2 秒を要した． insert クエリ処理に必要なページ数（約 200 ページ）は 1 MB バッファーの場合よりも少ないため，オーバヘッドはそれほど大きくならなかった．DB サイズが増加すると，オーバヘッドが増加する．マイクロベンチマークの実験で見られたように，実行時間が長いワークロードはバックグラウンドプロセスの影響を受けてオーバヘッドが増加する傾向がある．実行時間が長くなると測定データの SD が増加する傾向も，バックグラウンドプロセスによる干渉の有無が本来のワークロードの性質とは無関係に生じることを示唆している．

決定木の決定などのルール検索には、通常、複数のクエリが必要となる． 2 回目の select クエリの応答時間を最初のクエリの完了後に測定した． 2 回目のクエリでは DB プログラムと DB インデックスはすでにページバッファにロードされているため，応答は初回よりも速くなります． 図 12 に 4MB バッファの結果を示す．1 回目の応答と 2 回目の応答の比率は約 30 である．2 回目の応答は大きく改善しているが，st2 のみと比較した場合依然として約 100 倍の実行時間が要している．縦軸は対数目盛で表されている．

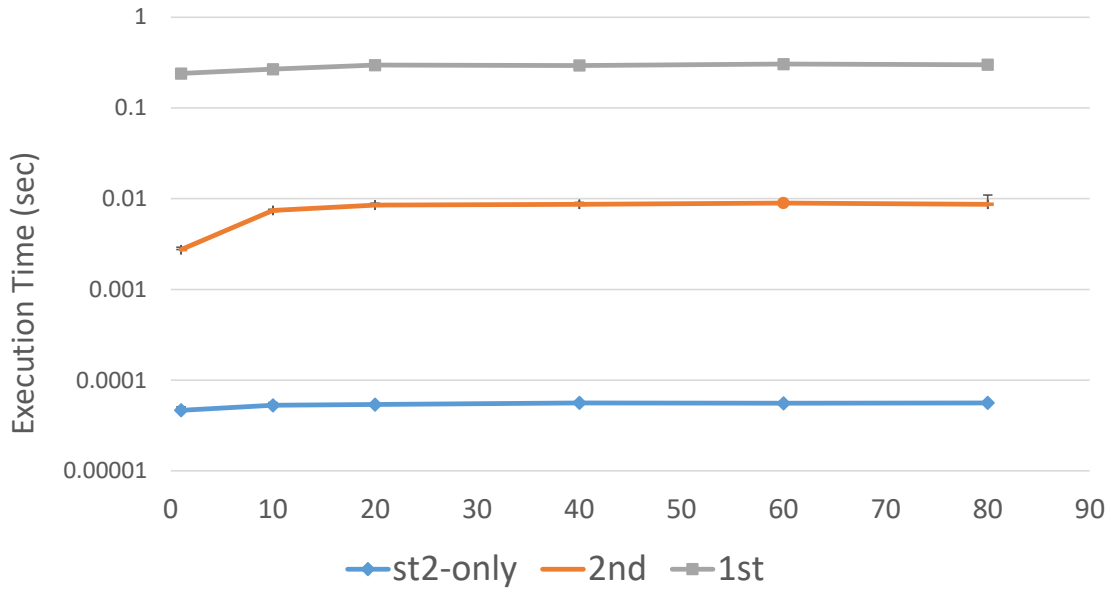


図 30　4MB バッファにおける 1 回目と 2 回目の応答時間比較

3.4.2.5. 実験結果のまとめ

上記の実験では，ページテーブルベースの検証ツリーを使用したページング処理によるオーバヘッドとワークロードの関係を調査した．比較的小さな SoC オンチップ SRAM に対してページングを行っても Linux システムは動作することは確認できたが．また，データベースなどフットプリントの大きいワークロードについては，セキュリティの利点と引き換えにかなりのオーバヘッドが発生することが観察された．

提案方式にはページバッファ効率に関して 2 つの弱点がある．1 つは，一般の HW キャッシュと比較してキャッシュの粒度が粗いため，バッファメモリに不要なデータまでロードされてしまい，効率が悪化する点である． もう 1 つは，検証ツリーの依存関係に起因する追加のページングが発生することである． これらは HW キャッシュに対してソフトウェアベースの手段を選択しことに起因するペナルティである．検証処理におけるオーバヘッドの最大の要素はデータ転送時間であるため，高性能な暗号付き DMAC を利用することが最も効果的である．本研究公表当時でも先端 SoC の暗号 DMAC は 100MB /秒に達していた．

２番目に挙げられるのが TLB とキャッシュの一貫性の改善である． ARMv7-仮想化アーキテクチャには IPA を指定した TLB 無効化命令がないため，プロトタイプの実装では，ページのフラッシュ操作ごとにすべての TLB エントリ無効化が必要だった． これにより，か

なりのオーバヘッドが発生していたと考えられる． ARMv8 アーキテクチャでは IPA を指定した TLB 無効化が導入されているため，TLB エントリの再ロードを減らすことで改善が期待できる． 本実験ではバッファメモリとして外部 DRAM を用いたが，SoC 内部メモリを使うことでより高速化が期待できる．さらに、ワークロードの特性を学習する等の改善手法も考えられるが，複雑化により内部脆弱性を導入しないよう留意が必要である．

実験結果は，オーバヘッドの中にバックグラウンドプロセスに起因する要素があることを示唆している．ワークロードに高レベルの優先度を指定することでオーバヘッドが軽減される可能性がある．セキュアサブシステムについては，不要なバックグランドプロセスを停止し，障害解析用データベースに対して高い優先度を指定することで性能が改善する可能性がある．

アプリケーションと OS のチューニングも候補に挙げられる．カーネルのフットプリントを縮小することも効果が期待できるが，大規模な変更は深刻な脆弱性をもたらす可能性がある． 脆弱性を回避するために，フィールドで実証済みのカーネルのライトウェイトバージョンが提案方式の応用に合致していると思われる．

3.5. 議論

3.5.1. 関連研究

仮想化技術を応用したシステム保護提案を表に示す. Cloudvisor と Trevisor は, x86 COTS HW のバッキングストア（HDD）にある OS イメージ全体の暗号化保護を提供する [60, 61]. これらは HDD 上のデータに Merkle ツリーベースのデータ整合性を提供する．だが，Cloudvisor には物理メモリ攻撃への対策はない．Trevisor は，メモリではなくレジスタのみを使用する特殊な暗号ソフトウェアモジュールによって，重要な秘密鍵処理について限定的な保護を提供する．Sentry はモバイルデバイス向けに提案された方式である [62]．利用者に与える暗号化のオーバヘッドを軽減するために，デバイスがロックされた状態でのみメモリ暗号化を適用する． だが，制御システムは常時運用，常時保護が基本であるため，Sentry の手法は適さない. また, Sentry は完全性保護の機能を欠いている. 提案方式 TREBIVE は，既存の COTS ハードウェアにおいて物理メモリ攻撃対策を提供する最初の VMM 実装である．

表 13 関連研究の比較

	COTS HW	Protection target	Entity	Memory attack	Merkle tree
Cloudvisor [57]	Yes	Whole OS	VMM	No	HDD only
Trevisor [58]	Yes	Whole OS	VMM	Key only	HDD only
Sentry [59]	Yes	Selected Part	OS inside	Yes	No
TREBIVE (提案方式)	Yes	Whole OS	VMM	Yes	Memory

3.5.2. ターゲットシステム内の役割分担

提案されたメカニズムはエンドポイントデバイスへの物理的攻撃に対するセキュリティを提供するが，エンドポイントのセキュリティはネットワーク攻撃対策も考慮が必要である．ネットワーク攻撃に対して VMM に組み込まれた仮想ファイアウォールも存在する．また，現在の OS には、ソフトウェアの脆弱性を軽減するためのネットワーク攻撃に対するさまざまな対策が用意されている． 既に述べた ASLR が代表的な手法である．ソフトウェア脆弱性に対する OS の対策と、物理的なメモリ攻撃に対する提案方式の VMM の組み合わせることが望ましいと考える．理由は，物理メモリ攻撃対策を要求するシステムの母数は，通信経由の攻撃対策を必要とするシステムの母数（ネットワーク接続された全てのシステム）より少ないためである．母数が少ないことにより，潜在的な脆弱性や機能停止につながるバグが見過ごされる可能性が高くなる. 提案方式には方式自体ではなく, 利用者数という弱点がある．この弱点は機能をシンプルにするというデザインポリシーでカバーするべきであり，そのためには通信経由の攻撃は OS レベルで対策するという，一種の住み分けが望ましい．物理メモリ完全性を必要とするシステムが, 少なくとも初期の段階では少ないという条件において，専用ハードウェアによる解決よりも汎用の仮想化機能を利用する提案方式が導入ハードルの低さの点で有利となる．

3.6. 完全性保護研究のまとめ

改ざん検証ツリーに基づき，物理メモリの整合性と機密性を保証する VMM TREBIVE を提案した．プロトタイプ実装の評価は，OS，システムコール，および DB アプリケーションに対して行った． オーバーヘッドは大きいものの，Linux システム全体のメモリを保護するという目的は達成できた．また，DB アクセス応答時間は DB 全体の検証よりも高速になることが期待できる． メインシステムとサブシステムを分割してサブシステムのみ物理メモリ保護を適用することでペナルティを緩和する利用シナリオも合わせて提案した．このシナリオでは，セキュアサブシステムはクリティカルな操作専用であり，スループット指向のサービスは，オーバヘッドの小さいメインシステムによって提供される．

完全性保護技術の想定脅威は物理メモリの改ざんであり，ゲスト OS の情報オーナーと，メモリに物理アクセス可能なサービスマンの間に生じる潜在的利害対立を緩和する完全性保護技術を提供するものである．

4. 本研究公表後の技術動向について

4.1. メインメモリ保護研究の進展

キャッシュハードウェア処理によるメインメモリ暗号化とハードウェアアクセス制御の連携による機密性保護技術は 2003 年から 2004 年にかけて発表された．その後，アカデミックな研究ではキャッシュハードウェア処理による完全性保護の研究が行われた [69, 70, 58, 77]．機密性保護については，仮想化機構を利用して専用ハードウェアなしで実現する Overshadow も提案された [74]．商用システムについては，組込み機器のデジタル著作権保護には Arm Trustzone が利用されるケースが多く見られた [53]．PC においてはソフトウェア難読化が多く用いられた [43]．

検証ツリー処理をキャッシュハードウェアベースで実現する際，メモリサイズが大きくなると検証ツリーの高さが高くなり，キャッシュ管理が難しくなる問題がある [69, 70]．この問題に対して，Chhabra らは，2009 年にページ内のキャッシュベース検証ツリー処理と，ページテーブルエントリ毎の検証ツリー処理を組合わせることで，検証ツリーの高さを半分に抑える提案を行っている [71]．この提案は単純なシミュレーションレベルでは良好な結果が得られている．しかしながら，ハードウェア実装を想定した場合，ページテーブルエントリの TLB 読み込みと連動して対応する検証ツリーの読込みと検証を並行して行う，いわば TLB のトレースキャッシュをハードウェアで構築する必要がある．同方式では，TLB とは別に検証ツリーキャッシュが保持され，正しく同期して更新されなければならない．同方式について，このような細部を考慮しないシミュレーション評価は行われているが [78]，ハードウェアレベルの実装評価はまだ行われていない．

本論文の研究以外には，ソフトウェアベースで検証ツリーによる物理メモリ保護を実現する提案は少ないが 2020 年にカリフォルニア大バークレイのグループがソフトウェアベースの検証ツリー実装に関する発表を行っている [79]．

4.2. 商用 CPU における保護技術の展開

商用プロセッサに対するメインメモリ暗号化導入は 2014 年に PC 向けに導入された Intel SGX に始まる [68]. Intel SGX は MAC ツリーによる物理メモリ保護もあわせて提供している. 半面, おそらく MAC ツリーの制約により SGX の最大メモリサイズは上限 128MiB に制約されていた [80]ため, 仮想マシン全体の保護には適用できなかった. MAC ツリーによる物理メモリ保護は, Intel 社所属となった Chhabra による先行研究を発展させたものと推定される [70, 71, 58].

AMD は 2017 年に商用化した AMD SEV において仮想マシン管理機構と連携して仮想マシン全体の暗号化を行うアーキテクチャを実用化した [81]. オリジナルの AMD SEV に対しては, ゲスト OS のページテーブル改ざんを通じて平文を出力させる攻撃が知られているが [82], 現在の AMD SEV-SNP ではページテーブル改ざんに特化した攻撃対策が盛り込まれている [83].

Intel も 2021 年に仮想マシン全体の暗号化を行う MK-TME を備えた CPU を製品化している [84]. ただし, MK-TME は SGX の MAC ツリーによる改ざん検証を含まず, 仮想マシン全体に対する保護は AES-XTS モードを用いた暗号化に限定されている. ページテーブルに対する改ざん攻撃対策は TDX として導入されているが, TDX は物理メモリ改ざん攻撃対策としては部分的にしか有効ではないとされている [85].

これら Intel, AMD の商用プロセッサには共通してキャッシュベースのメインメモリ暗号化機能が搭載されているが, キャッシュアクセス時の暗号処理と内部アクセス制御を組合わせる点は, 本研究の機密性保護技術やこれに先行する研究 [46, 45]と共通の枠組みが活用されている. このほか, IBM Power 10 アーキテクチャにおいてもメインメモリ暗号化が導入されている [86]. 組込み CPU を手掛ける Arm も既存の Trustzone に追加する形でメインメモリ暗号化を含む CCA (Confidential Computing Architecture)と呼ぶ新アーキテクチャを導入した [87]結果, 2020 年以降にサーバ向けに発表された Intel , AMD, IBM, Arm の CPU アーキテクチャは全てメインメモリ暗号化機能を備えている.

Arm CCA においては, 非セキュアワールドで実行される保護対象を Realm , 非セキュアワールドを管理する OS や VMM 環境を hosting environment と呼んでいる. Realm に対しては機密性, 完全性などのセキュリティ保護が提供されるのに対し, 非セキュアワールドの hosting environment はメモリやスケジューリングなどの資源割当てをフルコントロール

する権限を持つデザインとなっている [87, p. 16]．従来の Trustzone より本研究の 2.8.2 節で紹介した秘密保護と資源管理を分離する考え方に近いと考えられる．

なお，2021 年に更新された Intel SGX 仕様では最大メモリ容量が 1TB に拡張されている．しかしながら，1TB 拡張仕様では，従来採用していた検証ツリーベースのメモリ保護は採用されず，保護は AES-XTS モードによる暗号化のみに変更されている．詳細は明らかではないが，Intel 自身が変更の理由を Merkle Tree 技術の拡張性の低さによると説明しており，ハードウェアキャッシュベースの完全性検証における技術的困難性の傍証としてとらえられる [88].

5. 結論

5.1. 本研究の成果

本論文ではコンピュータシステムの水平分業化を背景として生じた，データならびにプログラムからなる処理対象の情報オーナーと，実行を行うシステムの管理権限者の間の不一致問題に関連する，機密保護と完全性保護に関するシステムセキュリティ研究について述べた．

第１章では，近年のクラウドコンピューティングの普及に伴って普及が進むシステムセキュリティ技術のコンフィデンシャルコンピューティング技術について，構成要素である内部アクセス制御とメインメモリ暗号化の結合による効果は，クラウドプロバイダの権限が不正利用された場合の重要データに対する脅威対策であることを示した．この脅威が顕在化した背景には，全般的なサイバー攻撃脅威の深刻化があることは自明だが，コンピュータシステム技術全般の水平分業化という組織に関わる要素も大きいことを明らかにした．従来は組織内で垂直統合的に管理されてきた重要データが，水平分業化により他組織が管理権限を持つ経路による脅威にさらされるようになったのである．個人所有かつオープンソース OS が稼働する PC にコンテンツプロバイダの重要データであるデジタルコンテンツが配信されるケースや，社会インフラ機器の物理メンテナンスがアウトソーシングされるケースについても，重要データが他組織の管理権限による脅威は発生する．メインメモリセキュリティに関する機密性保護技術は前者のデジタルコンテンツ保護，完全性保護技術は後者の社会インフラ機器の保護に関する脅威を背景として取組んだ研究である．

第２章においては，デジタル著作権保護における課題となっていたオープンソース OS 管理下で動作するプロセスに対する制約条件と脅威を分析し，OS による資源管理の制約の下でプロセスを構成する３要素の全てに対する暗号化と整合的なプロセス保護モデルを提案した．暗号化に関わる管理を OS とは独立したプロセッサハードウェアとして実装することにより，エンドユーザがオープンソース OS を自由に改造可能な環境においても，プロセスの保護に対する干渉を防止できる．同機能の FPGA 上ハードウェア実装ならびに対応 OS の実装を行い，プロセス３要素が全て暗号化された状態におけるプロセス実行と既存 OS と整合的な提案プロセス保護モデルの妥当性を確認した．メインメモリ上のプロセス３要素を暗号化保護する先行研究は存在したが，本研究はハードウェア実装による機能実証と回路規模を明確化した点が新規の貢献である．回路オーバヘッドは共通鍵，公開鍵暗号エンジンを含め

て約 200KiG と，ベースとなる組込み CPU ならびに周辺回路とほぼ同じ規模を必要としたが， SRAM 換算では 16.7KiB であり，既存組込み CPU のキャッシュメモリと同程度の面積増加にとどまることから，ターゲットである組込み CPU に適用可能な規模であることを実証した．性能オーバヘッドについても，キャッシュヒット率 99%の場合は 10%程度となる結果が ASIC パラメータを用いたシミュレーション評価で得られており，PC 等のデジタル著作権保護に用いられるソフトウェア難読化と比較して性能オーバヘッドも低い．

また，提案方式のプロセッサにおける保護プロセス実行を，外来プログラムをターゲットシステムで実行する依頼計算としてとらえたモデルを提案した．暗号化した外来プログラムがターゲットシステムで実行される場合，外来プログラムによるターゲットシステムのプライバシ脅威が問題となる．この問題に対して，OS による資源管理と整合的な提案方式のプロセス保護モデルにおいては，プロセス保護とターゲットシステム OS の資源管理ポリシーに基づくサンドボックスと組合わせることにより，外来プログラムの保護とプライバシ保護の両立を可能とする管理モデルを提案した．

第 3 章においては，社会インフラ機器の物理メンテナンス権限が悪用される脅威，中でも不揮発メモリの完全性を担保する技術提案と試作評価を行った．提案方式は従来のハードウェアキャッシュと連携したキャッシュライン単位の改ざん検出用のツリー検証処理を，汎用 CPU の仮想化機構を利用したページ単位の検証処理と CPU 内蔵メモリに対するページング処理により実現するものである．検証ツリーを拡張ページテーブル（EPT） のトポロジーと合致させることにより，特殊なハードウェアを用いることなく未検証メモリの参照の検出が可能となる．また，ページ検証を適用するゲスト OS とページ検証なしのゲスト OS が安全に共存可能な拡張ページテーブル管理方法も提案した．

提案方式の実装を行い，実行時オーバヘッドを評価した．Linux ブートにおいては，内蔵ページバッファ 4MiB の環境において，暗号連携 DMAC を利用すればページ検証なしの場合と比較して 3.8 倍の時間を要するものの，変更なしのゲスト OS に対してメモリ参照時の完全性検証が実現できることを実証した．また，データベースアクセスのクエリ実行においては，同じく内蔵ページバッファ 4MiB の環境において，検証なしの場合と比較して応答時間は約 3000 倍の時間を要した．ただし，データベースの使用前にそのイメージ全体の完全性検証を行う場合，データベースサイズが 30MiB 以上の領域では提案方式のほうが応答完了までの時間が短い．提案方式ではプログラム領域を含むすべてのデータに対する完全性検証が行われており，不揮発性メモリを使用した重要インフラ機器のセキュリティ保護に有効性があることを実証した．

第 4 章において，本研究公表後の研究動向および商用プロセッサの技術動向をまとめた．メインメモリ暗号化機能は，2021 年時点においてほぼ全てのサーバ向け CPU アーキテクチャに導入されており，PC 向けプロセッサにおいても普及が進みつつある．

本研究はそれぞれデジタル著作権保護ならびに社会インフラ機器に対する物理攻撃対策をターゲットとした技術だが，情報オーナーと管理権限の不一致としてとらえた場合，近年クラウドにおいて導入が進むコンフィデンシャルコンピューティングと共通の課題に対する取組みと考えることができる．特にメインメモリ暗号化については，本研究の提案方式と共通の枠組みが現在の商用プロセッサに用いられている．また，メモリ完全性については，現時点でも解決されていない課題であり，提案手法が今後活用される可能性がある．

以上のように，本研究はコンピュータの利用方法の変化という背景に基づいたセキュリティ要求の変化をとらえ，先行して応用に通じる課題解決の方式を提案し，実装評価を行ったものであり，課題発見，方式提案，評価実装のそれぞれについてオリジナリティと実用性における意義を主張しうるものである．

5.2. 今後の展望

最後に半導体実装技術の動向を踏まえて，メモリセキュリティ技術の課題と展望を述べる．

5.2.1. 機密性保護とチップ内メモリ

現在，ハイエンド CPU では 3 次元マルチチップ積層技術により 64MB におよぶパッケージ内 SRAM を備えた製品がすでに出荷されている [89]．マルチチップ積層技術自体は DRAM にも適用可能であることから，全ての機密保護対象情報 (Data In Use)がチップ内に格納できれば，ストレージにおける暗号化保護 (Data at Rest)との組合わせだけで保護が完結し，メインメモリ暗号化は不要となる可能性も理論的には考えられる．

しかしながら，市場メモリ価格のダイナミクスを考えれば，著者はこのシナリオが実現する可能性は極めて低いと考える．CPU にメモリを積層するコストは低下を続けているが，単体メモリと比較すれば依然としてコストは高い．メモリの価格低下は汎用メモリのスポット市場の存在によるところが大きいため，スポット市場が存在しない専用メモリは，たとえ製造コストが低くても価格が低下しにくい．部材調達コストの観点からは，市場で大量に流通する汎用 CPU と汎用メモリをその時点の価格で調達することがコスト削減には有効である．この市場メカニズムを考慮すれば，CPU にマルチチップ積層されるベアメモリチップも，汎

用メモリと比較して高止まりする可能性は高く，外付け汎用メモリを前提としたメインメモリ暗号化技術は今後も必要と考えられる．

5.2.2. メインメモリ完全性

一方，完全性保護技術で述べた内蔵メモリに対するページングによりメインメモリ完全性検証を実現する方式に対しては，チップ内蔵メモリの大容量化は有利に働く．完全性保護技術の評価で示したようにページングによるメインメモリ完全性検証の性能ペナルティは内蔵メモリのページバッファサイズに依存するため，内蔵メモリの大容量化により性能ペナルティは低下する．ただし，完全性検証用にページバッファを占有すれば通常処理用のハードウェアキャッシュ容量が削られ，全体のパフォーマンスが低下するトレードオフが生じる．この問題に対しては，対象システムの特性に応じて CPU キャッシュとページキャッシュに対する SRAM 割当てをコンフィギュアラブルとするメカニズムが必要と思われる．

大容量化とは別に，ページキャッシュにダーティなデータがある状態の電源断によるデータ損失の問題がある．この問題自体は，ページキャッシュに関するチェックポイント - リカバリ処理を適用することで解決すべき課題である．他方，半導体の内蔵メモリとして MRAM のような不揮発メモリがすでに実用化されている [90]．チップ内蔵ページキャッシュに不揮発メモリを用いることで，上記電源断後のリカバリ処理を大幅に簡略化できる可能性があり，メインメモリ完全性保護システムの耐障害性と性能に対する向上効果が期待できる．

5.2.3. メインメモリ完全性検証のニーズ

４章で述べたように，Intel SGX の拡張においてツリー検証に代わって暗号化が用いられたことから，厳密なメインメモリ完全性検証のニーズは機密性と比較した場合，現時点では相対的に低いと考えられる．だが，制御システムにおけるいわゆる DX（Digital Transformation）と省人化が進めば制御機器が人間のオペレータに依存せず異常事態の対応を行うことが必須となることから，長期的にメインメモリ完全性検証のニーズは高まることが予想される．半導体は複雑かつ大規模な機能であっても量産によりコストが劇的に低下する一方で，複雑な機能の設計には開発コストがかかる．メインメモリ完全性検証のようにまだ普遍的なニーズと言えない機能の導入段階においては，最初から全てをハードウェア化して高パフォーマンスを得るよりシンプルなハードウェアとソフトウェア処理とを組合わせて，現時点ではまだニッチにとどまる要求に対応しながら，真に普遍的な要求を明らかにするアプローチが有用と思われる．上述のように Intel SGX に導入されたキャッシュライン単位のハードウェア完全性検証は大容量化において行詰りが明らかとなったが，メインメモリ

完全性検証のニーズは今後さらに高まると思われる．本研究で示したソフトウェアベースのアプローチはこの状況に対する解決になる可能性がある．

5.2.4. メモリセキュリティ技術の展望のまとめ

社会インフラの IT 化と水平分業化の進展に伴い，メモリセキュリティの要求は今後高まると予想されるが，対策技術はまだ完成に至っていない．この課題に対して，今後の半導体実装技術とシステムアーキテクチャの水平分業化を考慮に入れた取組みが必要である．

文献目録

[1] FBI, Internet Crime Complaint Center, "Internet Crime Report," Federal Bureau of Investigation, 2021.

[2] R. Witty, "Predicts 2020: Security and Risk Management Programs," Gartner report G00731693, 2020.

[3] Z. M. Smith, E. Lostri , J. A. Lewis, “The Hidden Costs of Cybercrime,” Center for Strategic and International Studies (CSIS), 2021.

[4] C. Singleton, “X-Force Threat Intelligence Index 2021,” IBM Corporation, 2021.

[5] S. Upadhyay and et al., "Forecast: Information Security and Risk Management, Worldwide, 2019-2025, 2Q21 Update," Gartner report G00752504, 2021.

[6] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," 2018.

[7] M. Mueller, Ruling the root, MIT Pres, 2004.

[8] J. H. Saltzer, D. P. Reed and D. D. Clark, "End-to-End Arguments in System Design," *ACM Trans. Comput. Syst.,* vol. 2, p. 277–288, November 1984.

[9] R. Anderson, "4. Access Control," in *Security Engineering (2nd eds)*, Wiley, 2008.

[10] Confidential Computing Consortium, "Confidential Computing: Hardware-Based Trusted Execution for Applications and Data," Jan. 2021. [Online]. Available: https://confidentialcomputing.io/wp-content/uploads/sites/85/2021/03/confidentialcomputing_outreach_whitepaper-8-5x11-1.pdf. [Accessed Dec. 2021].

[11] M. Russinovich, M. Costa, C. Fournet, D. Chisnall, A. Delignat-Lavaud, S. Clebsch, K. Vaswani and V. Bhatia, "Toward Confidential Cloud Computing," *Commun. ACM,* vol. 64, p. 54–61, May 2021.

[12] W. Isaacson, "Chapter 2, The Computer," in *The Innovators: How a Group of Hackers, Geniuses, and Geeks Created the Digital Revolution*, Simon & Schuster, 2014.

[13] A. S. Tannenbaum, "1.2.2 The second generation (1955-1965) Transistor and batch system," in *Modern Operating System*, Pearson, 2009, p. 9.

[14] IEEE History Center, "IEEE Milestones:Semi-Automatic Ground Environment (SAGE) 1951-1958," 2012. [Online]. Available: https://ethw.org/Milestones:Semi-Automatic_Ground_Environment_(SAGE)_1951-1958. [Accessed Dec. 2021].

[15] Sabre Corporation, "The Sabre Story," Sabre Corporation, 2017.

[16] G. A. Blaauw and P. B. Frederick, "Section 11.4 IBM 704," in *Copmuputer Architecture, Concepts and Evolution Part II*, Addison-Wesley, 1997, p. 639.

[17] A. S. Tannenbaum, "1.2.3 The third generation (1965-1980) ICs and Multiprogramming," in *Modern Operating System*, Pearson, 2009, p. 10.

[18] T. V. Vleck, "The IBM 360/67 and CP/CMS," 2010. [Online]. Available: https://multicians.org/thvv/360-67.html. [Accessed Dec. 2021].

[19] R. Anderson, "8. Multilevel Security," in *Security Engineering (2nd eds)*, 2008.

[20] D. Bell and L. LaPadula, "Secure Computer Systems," Mitre Corporation, 1974.

[21] C. Weissman, "Security Controls in the ADEPT-50 Time-Sharing System," in *Proceedings of the November 18-20, 1969, Fall Joint Computer Conference*, New York, NY, USA, 1969.

[22] L. J. Fraim, "Scomp: A Solution to the Multilevel Security Problem," *Computer,* vol. 16, p. 26–34, July 1983.

[23] C. Weissman, "BLACKER: security for the DDN examples of A1 security engineering trades," in *Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, 1992.

[24] K. E. Hickman, "The SSL Protocol," 1995.

[25] S. Perez, "EFF: Half of web traffic is now encrypted," Feb. 2017. [Online]. Available: https://techcrunch.com/2017/02/22/eff-half-the-web-is-now-encrypted/. [Accessed Dec. 2021].

[26] M. Hoekstra, R. Lal, P. Pappachan, V. Phegade and J. del Cuvillo, "Using innovative instructions to create trustworthy software solutions," in *HASP 2013, The Second*

Workshop on Hardware and Architectural Support for Security and Privacy, Tel-Aviv, Israel, June 23-24, 2013, 2013.

[27] N. Jentzsch, "Horizontal and Vertical Analysis of Privacy and Cyber-Security Markets," 2015.

[28] R. Grant, The quest for new organizational forms: The strange case of open source software communities, 2007, p. 55–81.

[29] Google LLC, "Confidential VM and Compute Engine," [Online]. Available: https://cloud.google.com/compute/confidential-vm/docs/about-cvm. [Accessed Nov. 2021].

[30] X. Zhang, J. Schiffman, S. Gibbs, A. Kunjithapatham and S. Jeong, "Securing Elastic Applications on Mobile Devices for Cloud Computing," in *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, New York, NY, USA, 2009.

[31] R. R. Brooks, "Mobile code paradigms and security issues," *IEEE Internet Computing,* vol. 8, pp. 54-59, 2004.

[32] K. Kato and Y. Oyama, "SoftwarePot: An Encapsulated Transferable File System for Secure Software Circulation," in *Software Security – Theories and Systems, Mext-NSF-JSPS International Symposium, ISSS 2002, Tokyo, Japan, November 8-10, 2002, Revised Papers*, 2002.

[33] T. Sander and C. F. Tschudin, "Towards Mobile Cryptography," in *Security and Privacy - 1998 IEEE Symposium on Security and Privacy, Oakland, CA, USA, May 3-6, 1998, Proceedings*, 1998.

[34] B. Hicks, S. Rueda, L. St.Clair, T. Jaeger and P. McDaniel, "A Logical Specification and Analysis for SELinux MLS Policy," *ACM Trans. Inf. Syst. Secur.,* vol. 13, July 2010.

[35] M. Schaefer, "If A1 is the answer, what was the question? An Edgy Naif's retrospective on promulgating the trusted computer systems evaluation criteria," in *20th Annual Computer Security Applications Conference*, 2004.

[36] J. A. Bloom, I. J. Cox, T. Kalker, J.-P. M. G. Linnartz, M. L. Miller and C. B. S. Traw, "Copy protection for DVD video," *Proceedings of the IEEE,* vol. 87, pp. 1267-1276, 1999.

[37] Digital Content Protection LLC, "High-bandwidth Digital Content Protection (HDCP) Specification Revision 1.4," Jul 2009. [Online]. Available: https://www.digital-cp.com/sites/default/files/specifications/HDCP%20Specification%20Rev1_4_Secure.pdf. [Accessed Dec. 2021].

[38] A. Patrizio, "Why the DVD Hack Was a Cinch," 1999. [Online]. Available: https://www.wired.com/1999/11/why-the-dvd-hack-was-a-cinch/. [Accessed Nov. 2021].

[39] G. Joe, Game Console Hacking, Syngress Press, 2004.

[40] 電子情報通信学会, 著: *情報セキュリティハンドブック*, オーム社, 2004, p. 3.

[41] Y. Kim, R. Daly, J. S. Kim, C. Fallin, J.-H. Lee, D. Lee, C. Wilkerson, K. Lai and O. Mutlu, "RowHammer: Reliability Analysis and Security Implications," *CoRR,* vol. abs/1603.00747, 2016.

[42] A. Eskicioglu, "Protecting Intellectual Property in Digital Multimedia Networks," *Computer,* vol. 36, pp. 39-45, 2003.

[43] D. Aucsmith, "Tamper Resistant Software: An Implementation," in *Proceedings of the First International Workshop on Information Hiding*, Berlin, 1996.

[44] M. G. Kuhn, "Cipher Instruction Search Attack on the Bus-Encryption Security Microcontroller DS5002FP," *IEEE Trans. Comput.,* vol. 47, p. 1153–1157, October 1998.

[45] D. Lie, C. Thekkath, M. Mitchell, P. Lincoln, D. Boneh, J. Mitchell and a, "Architectural Support for Copy and Tamper Resistant Software," in *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems*, New York, NY, USA, 2000.

[46] T. Gilmont, J.-d. Legat and J.-j. Quisquater, "An Architecture of Security Management Unit for Safe Hosting of Multiple Agents," 1998.

[47] T. Miyamori, "A Configurable and Extensible Media Processor," in *Embedded Processor Forum*, San Jose, USA, 2002.

[48] TOPPERS プロジェクト, "TOPPERS/JSP カーネルとは," 2008. [Online]. Available: https://www.toppers.jp/project.html. [Accessed Dec. 2021].

[49] 寺本圭一, 橋本幹生, 白川健治, 尾崎哲 , 藤本謙作, “耐タンパプロセッサにおける暗号化データ領域のプロセス間共有方法,” 特開 2002-202720(特許 4074057), 2002.

[50] Keiichi Teramoto, M. Hashimoto, K. Shirakawa, S. Ozaki and K. Fujimoto, "Method for sharing encrypted data region among processes in tamper resistant processor," US 7,657,760, 2010.

[51] M. Hashimoto, H. Haruki and T. Kawabata, "Secure Processor Consistent with Both Foreign Software Protection and User Privacy Protection," in *Security Protocols, 12th International Workshop, Cambridge, UK, April 26-28, 2004. Revised Selected Papers*, 2004.

[52] P. England, B. Lampson, J. Manferdelli and B. Willman, "A trusted open platform," *Computer,* vol. 36, pp. 55-62, 2003.

[53] S. Pinto and N. Santos, "Demystifying Arm TrustZone: A Comprehensive Survey," *ACM Comput. Surv.,* vol. 51, January 2019.

[54] 橋本幹生, “プロセッサ、プロセッサシステム及びキャッシュ一貫性制御方法,” 特開 2005-18379(登録第 4021810 号), 2005.

[55] V. Bashun, A. Sergeev, V. Minchenkov and A. Yakovlev, "Too young to be secure: Analysis of UEFI threats and vulnerabilities," in *14th Conference of Open Innovation Association FRUCT*, 2013.

[56] H. Nakamura, T. Nakada and S. Miwa, "Normally-off computing project: Challenges and opportunities," in *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2014.

[57] K. Bailey, L. Ceze, S. D. Gribble and H. M. Levy, "Operating System Implications of Fast, Cheap, Non-volatile Memory," in *Proceedings of the 13th USENIX Conference on Hot Topics in Operating Systems*, Berkeley, CA, USA, 2011.

[58] S. Chhabra and Y. Solihin, "i-NVMM: A Secure Non-volatile Main Memory System with Incremental Encryption," in *Proceedings of the 38th Annual International Symposium on Computer Architecture*, New York, NY, USA, 2011.

[59] M. Henson and S. Taylor, "Beyond Full Disk Encryption: Protection on Security-enhanced Commodity Processors," in *Proceedings of the 11th International Conference on Applied Cryptography and Network Security*, Berlin, Heidelberg, 2013.

[60] F. Zhang, J. Chen, H. Chen and B. Zang, "CloudVisor: Retrofitting Protection of Virtual Machines in Multi-tenant Cloud with Nested Virtualization," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, New York, NY, USA, 2011.

[61] T. Muller, B. Taubmann and F. C. Freiling, "TreVisor: OS-independent Software-based Full Disk Encryption Secure Against Main Memory Attacks," in *Proceedings of the 10th International Conference on Applied Cryptography and Network Security*, Berlin, Heidelberg, 2012.

[62] P. Colp, J. Zhang, J. Gleeson, S. Suneja, E. de Lara, H. Raj, S. Saroiu and A. Wolman, "Protecting Data on Smartphones and Tablets from Memory Attacks," in *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, New York, NY, USA, 2015.

[63] H. L. Hassani, A. Bahnasse, E. Martin, C. Roland, O. Bouattane and M. E. M. Diouri, "Vulnerability and security risk assessment in a IIoT environment in compliance with standard IEC 62443," *Procedia Computer Science,* vol. 191, pp. 33-40, 2021.

[64] H. Yamada and K. Kono, "Traveling Forward in Time to Newer Operating Systems Using ShadowReboot," in *Proceedings of the 9th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, New York, NY, USA, 2013.

[65] C. Dall and J. Nieh, "KVM/ARM: The Design and Implementation of the Linux ARM Hypervisor," in *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, New York, NY, USA, 2014.

[66] Freescale Semiconductor Inc., "QorIQ LS1021A Communications Processor," 2014.

[67] Toshiba Electronic Devices & Storage Corporation, "TZ2100 Series Application Processor Lite ApP Lite," Apr. 2018. [Online]. Available: https://toshiba.semicon-

storage.com/us/semiconductor/product/microcontrollers/detail.TZ2100XBG.html. [Accessed Dec. 2021].

[68] F. McKeen, I. Alexandrovich, A. Berenzon, C. Rozas, H. Shafi, V. Shanbhogue and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution," in *{HASP} 2013, The Second Workshop on Hardware and Architectural Support*, 2013.

[69] G. E. Suh, D. Clarke, B. Gassend, M. v. Dijk and S. Devadas, "Efficient Memory Integrity Verification and Encryption for Secure Processors," in *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, Washington, DC, USA, 2003.

[70] B. Rogers, S. Chhabra, M. Prvulovic and Y. Solihin, "Using Address Independent Seed Encryption and Bonsai Merkle Trees to Make Secure Processors OS- and Performance-Friendly," in *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, Washington, DC, USA, 2007.

[71] S. Chhabra, B. Rogers, Y. Solihin and M. Prvulovic, "Making Secure Processors OS- and Performance-friendly," *ACM Trans. Archit. Code Optim.,* vol. 5, no. 4, pp. 16:1--16:35, #mar# 2009.

[72] H. Shacham, M. Page, B. Pfaff, E.-J. Goh, N. Modadugu and D. Boneh, "On the Effectiveness of Address-Space Randomization," in *Proceedings of the 11th ACM Conference on Computer and Communications Security*, New York, NY, USA, 2004.

[73] D. Gruss, C. Maurice, A. Fogh, M. Lipp and S. Mangard, "Prefetch Side-Channel Attacks: Bypassing SMAP and Kernel ASLR," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 2016.

[74] X. Chen, T. Garfinkel, E. C. Lewis, P. Subrahmanyam, C. A. Waldspurger, D. Boneh, J. Dwoskin and D. R. K. Ports, "Overshadow: a virtualization-based approach to retrofitting protection in commodity operating systems," *ACM SIGOPS Operating Systems Review,* vol. 42, p. 2–13, 2008.

[75] A. F. Voellm, "byte-unixbench," Jan. 2018. [Online]. Available: https://github.com/kdlucas/byte-unixbench. [Accessed Dec. 2021].

[76] The SQLite Consortium, "What Is SQLite?," May. 2000. [Online]. Available: https://www.sqlite.org/index.html. [Accessed Dec. 2021].

[77] S. Chhabra, B. Rogers, Y. Solihin and M. Prvulovic, "SecureME: A Hardware-software Approach to Full System Security," in *Proceedings of the International Conference on Supercomputing*, New York, NY, USA, 2011.

[78] Y. Xia, Y. Liu, H. Guan, Y. Chen, T. Chen, B. Zang and H. Chen, "Secure Outsourcing of Virtual Appliance," *IEEE Transactions on Cloud Computing,* vol. 5, pp. 390-404, 2017.

[79] G. Andrade, D. Lee, D. Kohlbrenner, K. Asanovic and D. Song, "Software-Based Off-Chip Memory Protection," Fourth Workshop on Computer Architecture Research with RISC-V (CARRV 2020), 2020.

[80] Intel Corporation, "Intel Software Guard Extensions (Document Number 332680-001)," 2015.

[81] D. Kaplan, J. Powell and T. Wolle, "AMD MEMORY ENCRYPTION," Apr. 2016. [Online]. Available: http://developer.amd.com/wordpress/media/2013/12/AMD_Memory_Encryption_Whitepaper_v7-Public.pdf. [Accessed Dec. 2021].

[82] M. Morbitzer, S. Proskurin, M. Radev, M. Dorfhuber and E. Q. Salas, "SEVerity: Code Injection Attacks against Encrypted Virtual Machines," in *2021 IEEE Security and Privacy Workshops (SPW)*, 2021.

[83] Advanced Micro Devices, "AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More," Jan. 2020. [Online]. Available: https://www.amd.com/system/files/TechDocs/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf. [Accessed Dec. 2021].

[84] Inrtel Corporation, "Intel Architecture Memory Encryption Technologies Speficification (Document Number 336907-003US)," 2021.

[85] Intel Corporation, "Intel Trust Domain Extensions whitepaper (Document Number 343961-002)," 2021.

[86] W. Starke and B. Thompto, "IBM's POWER10 Processor," in *2020 IEEE Hot Chips 32 Symposium (HCS)*, 2020.

[87] Arm Limited, Architecture & Technology Group, "Arm CCA Security Model 1.0," Aug. 2021. [Online]. Available: https://developer.arm.com/documentation/DEN0096/latest. [Accessed Dec. 2021].

[88] Intel Corporation, "What Technology Change Enables 1 Terabyte (TB) Enclave Page Cache (EPC) size in 3rd Generation Intel® Xeon® Scalable Processor Platforms?," Jul. 2021. [Online]. Available: https://www.intel.com/content/www/us/en/support/articles/000059614/software/intel-security-products.html. [Accessed Dec. 2021].

[89] M. Evers, L. Barnes and M. Clark, "Next Generation "Zen 3" Core," in *2021 IEEE Hot Chips 33 Symposium (HCS)*, 2021.

[90] Y.-D. Chih, Y.-C. Shih, C.-F. Lee, Y.-A. Chang, P.-H. Lee, H.-J. Lin, Y.-L. Chen, C.-P. Lo, M.-C. Shih, K.-H. Shen, H. Chuang and T.-Y. J. Chang, "13.3 A 22nm 32Mb Embedded STT-MRAM with 10ns Read Speed, 1M Cycle Write Endurance, 10 Years Retention at 150°C and High Immunity to Magnetic Field Interference," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2020.

謝辞

学位審査にあたりご多忙かつコロナ禍の中快く審査をお引き受けくださいました筑波大学加藤和彦教授，朴泰祐教授，中田秀基教授，大山恵弘准教授，面和成准教授，阿部洋丈准教授に深謝いたします．多面的な視点から，貴重なアドバイス，ご指導を賜りました．特に主査を引き受けてくださいました加藤和彦教授には，入学前から親身に相談にのっていただき暖かいアドバイスと的確なご指導をいただきました．また，OSSS 研究室の阿部洋丈准教授には多くのディスカッションの時間と配慮に満ちたご指導をいただき，お礼申し上げます．

本研究は東芝の優秀な共同研究者の協力がなければなしえないものでした．特に春木洋美氏，川端健氏，金井遵氏，山田菜穂子氏にはディスカッションと実装で大きな協力をいただいており，感謝いたします．また，それまでのソフトウェア畑からハードウェアの研究に着手した際には，東芝のオリジナル RISC プロセッサ MeP 開発の中心メンバである宮森高氏，玉井孝典氏との議論により多くの課題が明確化できました．白川健治氏，藤本謙作氏，寺本圭一氏，横山淳氏にも多くの技術アドバイスをいただきました．着手時点では多くのリスクがあった研究活動に理解と支援をいただいた上司の山田尚志氏，釜谷幸男氏，川村新一氏，井上敦氏，新保淳氏，雨宮治郎氏にお礼申し上げます．

東京大学坂井修一教授，入江英嗣准教授にはセキュリティアーキテクチャ研究の目指すべき方向性につき高い視点から貴重なアドバイスをいただきました．お礼申し上げます．フェローとして東芝を訪問中のケンブリッジ大　Frank Stajano 教授からは国際学会への投稿を勧めていただきました．

著者が学生時代の専門分野とは異なるテーマで研究活動をはじめ，デジタル著作権保護に取組むきっかけは慶応義塾大学との R3 プロジェクトでした．機会を与えていただいた現 NICT 徳田英幸先生，青山学院大学戸辺義人教授，慶応義塾大学中澤仁教授，東芝の岡本利夫氏，斎藤健氏，高畠由彰氏，正畑康郎氏にこの場をお借りして感謝の意を表します．

コロナ禍においてリモートで行われることになりましたが，OSSS 研究室の学生の皆様との技術ディスカッションも本論文をまとめるにあたり大変有益でした．最後に，職場の同僚，そして家族の協力に感謝いたします．

本研究に関する発表論文

本論文の内容に関する発表論文を以下に挙げる．著者によるこれら論文の博士論文に関する再利用は，情報処理学会，Springer-Verlag, IEEE の各著作権者の許諾と厚意によるものであり，ここに謝意を表す.

第２章　機密性保護技術について

論文誌（トランザクション）

- 橋本 幹生 , 春木 洋美: 敵対的な OS からソフトウェアを保護するプロセッサアーキテクチャ,情報処理学会論文誌コンピューティングシステム（ACS） 45(SIG03(ACS5)), (2004), 1-10

国際学会

- Mikio Hashimoto, Takeshi Kawabata, Hiroyoshi Haruki: Secure Processor Consistent with Both Foreign Software Protection and User Privacy Protection, "Proceedings of Security Protocols Workshop (Cambridge University) (2004), LNCS-3957: 276-286.

第３章　完全性保護技術について

国際学会

- Mikio Hashimoto, Naoko Yamada, Jun Kanai: TREBIVE: A TREe Based Integrity Verification Environment for Non-volatile Memory System., Proceedings of IEEE PRDC 2015 (2015) p.279-289.

その他

国際学会併設チュートリアル講演資料

- M. Hashimoto, "Overview of Memory Security Technologies," *2021 International Symposium on VLSI Technology, Systems and Applications (VLSI-TSA)*, 2021, pp. 1-2, doi: 10.1109/VLSI-TSA51926.2021.9440133.