オープンソースOSと共存可能なセキュリティプロセッサ技術

Security Processor Technology Compliant with Open-Source Operating Systems

橋本　幹生　■HASHIMOTO Mikio
春木　洋美　■HARUKI Hiroyoshi
川端　健　■KAWABATA Takeshi

アプリケーションプログラムをリバースエンジニアリングから保護するセキュリティプロセッサアーキテクチャL-MSP（License controlling Multiparty Secure Processor）を開発した。

L-MSPはLinux（注1）やITRONのようなオープンソースOS（基本ソフトウェア）を前提として，マルチタスク環境で動作するアプリケーションプログラムを解析・改変から保護することが可能であり，オープンシステム環境におけるデジタルコンテンツの著作権保護やソフトウェアの知的財産権保護などへの応用が可能である。

Toshiba has developed a security processor architecture called the license controlling multiparty secure processor (L-MSP).

L-MSP permits neither analysis nor modification of application programs running on open-source operating systems such as Linux or ITRON. This technology is based on embedded cryptographic hardware and access control mechanisms, and can be used for digital content protection, intellectual property protection, and other such applications.

1 まえがき

オープン化された情報インフラの普及を背景として，デジタル化された情報の知的財産権侵害やネットワークを経由した攻撃への対策が求められつつある。

仕様が公開されたオープンシステムはメーカーを超えたプラットフォームソフトウェアの共有を可能にし，その理念は広く受け入れられている。LinuxやITRONに代表されるオープンソースOSは，コンシューマ向けにも広く使われるようになった。

一方で，多様なユーザーが様々な場所で使うオープンシステムには，従来考えられていなかったセキュリティ問題が発生する。その代表がソフトウェアやデジタルコンテンツの不正コピー問題，及びプログラムの逆解析（リバースエンジニアリング）による知的財産権侵害である。その解決に必要な要素技術として，信頼できないシステム上で動作するソフトウェアを解析・改変から保護するソフトウェア保護技術が求められている。なかでも困難な課題は，オープンソースOSのようにエンドユーザーやサードパーティが自由に改造できるOSを前提としたソフトウェア保護の実現方式である。

ここでは前記課題を背景に東芝が開発した，セキュリティプロセッサアーキテクチャL-MSP [(1)]について述べる。L-MSPは，マルチタスクOSの資源管理機能と共存可能なソフトウェアのリバースエンジニアリング防止機構である。

（注1） Linuxは，Linus Torvalds氏の米国及びその他の国における登録商標。

2 オープンシステムとソフトウェア保護

2.1 ソフトウェア保護の必要性

ソフトウェア保護の必要性を，単純化したコンテンツ保護システムのモデルにそって説明する。**図1**はコンテンツ配信から端末装置における再生までの概念図である。

コンテンツ権利者は，記録媒体やネットワーク経由でコンテンツをユーザー端末に配信する。コンテンツは不正コピーなどを防止するため暗号化されている。ユーザー端末はエンドユーザーの管理下にあるオープンシステムであって，OSはLinuxのようなオープンソースOSである。

コンテンツの再生処理は，ソフトウェアベンダーが供給する，コンテンツ権利者から委託されたコンテンツ復号鍵を埋め込んだ専用の再生プログラムによって行われる。再生プログラム実行の結果，コンテンツ復号鍵によって復号されたコンテンツの再生結果が出力され，ユーザーはコンテンツを楽しむことができる。

再生プログラムはOSの管理下でほかのプログラムと並行して実行され，通常それぞれに隔離されたメモリ空間が与えられる。このようなOSの管理下で並行的に実行状態にあるプログラムの一つ一つはプロセスと呼ばれる。

ここでは，コンテンツ保護を“エンドユーザーに平文のデジタルコンテンツを取得させないこと”と定義する。ユーザーに提示する再生結果はアナログ出力に限定し，デジタルでのコピーは禁止とする。代表的な脅威は再生プログラムに埋め込まれたコンテンツ復号鍵がエンドユーザーによって

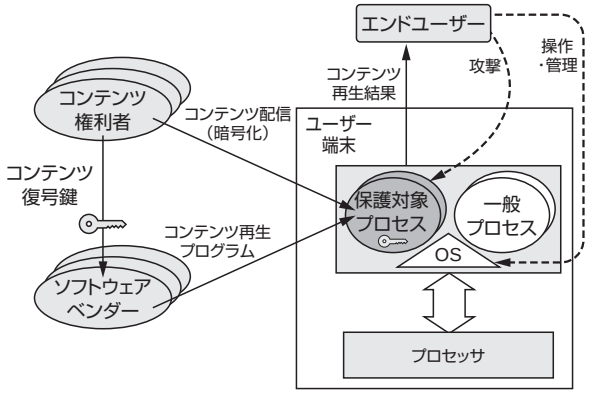

図1. コンテンツ再生システムと再生ソフトウェアへの攻撃 ―
プロセスが持つコンテンツ復号鍵がエンドユーザーの攻撃を受ける。
Content playback system and attack on playback software

取り出され，暗号化コンテンツから自由に平文コンテンツを生成されてしまうことである。実際，DVDの著作権保護システムであるCSS（Content Scramble System）方式について，あるソフトウェアベンダーの再生プログラムの不備から秘密鍵などが流出し，ホビープログラマーによりDeCSSと呼ばれるDVD 不正コピープログラムが作られた事例があった[(2)]。

このような攻撃を防止するためには，まず復号鍵を含む再生プログラムのファイル上イメージが秘匿されなければならない。また，実行中のメモリ上プロセスイメージも同様に秘匿が必要である。更に，コンテンツ復号鍵の保護に加えて，プロセスイメージに保持されているアナログ出力前の平文状態コンテンツについても秘匿が必要である。図1では保護が必要なプロセスを保護対象プロセス，それ以外のプロセスを一般プロセスとして示した。

既存システムにおいて，ソフトウェアの秘匿には難読化などの手法が使われてきた[(2)]。しかし，プロセスイメージ全体の秘匿保護は困難であり，特にオープンソースOSを前提とした場合，決定的な対策は存在しなかった。

2.2 ソフトウェア保護とOS機能との関係

既存システムの保護機構でも，プロセスごとに隔離されたメモリ空間を割り当てることで，プロセス間及びシステム上のユーザー間での不正な操作や干渉が防止されている。

しかし，このような機構はシステム管理者に信頼をおくものであって，エンドユーザーがシステム管理者でもある場合には必ずしも有効ではない。特にエンドユーザーがOSの設定を変更したり，極端な場合にはOSのソースプログラムを書き換えて保護機構の設定を変えてしまうような攻撃から，保護対象のプロセスを保護する仕組みは従来存在しなかった。

プロセスに対する攻撃手段は，以下の2種類に大別できる。

（A）外部メモリ及び二次記憶上のデータアクセス

（B）OS向け管理機能によるソフトウェアアクセス

（A）は，ICE（In Circuit Emulator）などの機材を使ってCPUバスを流れる信号を観測したり，仮想記憶機能によって磁気ディスク装置（HDD）などの二次記憶装置に一時的に保存されたプロセスイメージを観測するような攻撃である。

（B）は，プロセスの実行を中断・再開したり，割り当てるメモリのサイズを増減したり，メモリ内容を二次記憶装置に退避するような，OSに必須の特権機能を使ってプロセスイメージを観測する攻撃である。

これらの機能は抽象的に，CPU時間と記憶領域という2種類の資源をプロセスに配分するための資源管理機能とも呼ばれる。OSが背後で資源管理を行っているため，端末によってCPUの個数やメモリサイズに違いがあるにもかかわらず，プログラマーは資源管理の存在を意識することなく（透過性），アプリケーションに必要な問題解決に集中することができる。資源管理機能はソフトウェア保護の観点からは好ましくない機能だが，計算機資源を効率的に利用し，効率的にプログラムを開発するためには不可欠な機能である。

資源管理機能を利用した攻撃は，特殊な機材を必要としない。オープンシステムでは資源管理機能の仕様やソースコードが公開されているため，ある程度の技術を持つエンドユーザー（ホビープログラマーを含む）は，必要な情報を入手して攻撃用のプログラムを作成することができる。

ここで解決しようとする課題は，ソフトウェア保護機能とオープンソースOSを前提とした資源管理機能との両立にある。前記の理由により，OSをソフトウェア保護の基盤とすることはできないため，代わりにCPUパッケージの安全性をソフトウェア保護の基盤とする。CPUパッケージを開封して内部の秘密情報を読み取ったり，制御回路を書き換えたりすることは，専用の機材を持たないユーザーには不可能であり，ソフトウェア保護の基盤として適切といえる。

3 L-MSPの保護メカニズム

この章では前述の課題を解決するプロセッサアーキテクチャL-MSP（License cotrolling Multiparty Secure Processor）の詳細を説明する。L-MSPは単にプログラムを秘匿するだけでなく，実行中のプロセスが持つデータなども暗号化により秘匿保護できる。これには，プログラムがサーバなどとの通信により動的にサービスの利用権，すなわちライセンスを獲得するようなシステムへの考慮が背景にあり，license controllingと名づけた理由である。

3.1 プロセスと資源管理のモデル

説明の準備として，プロセスと資源保護のモデルを**図2**に示す。プロセスの構成要素には，プログラム（命令），データ，コンテキスト（レジスタ）情報の3種類がある。

プログラム及びデータは，キャッシュラインと呼ばれる短い

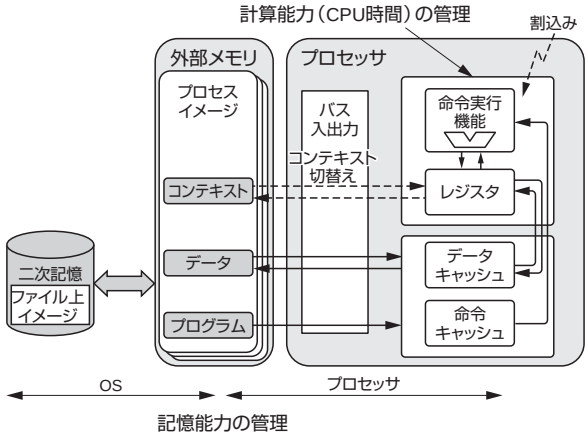

図2．OSの資源管理機能と攻撃手段への転用 ― 計算能力と記憶能力の管理機能は，ソフトウェアに対する攻撃手段としても利用できる。

Operating system resource management diverted to means of attack

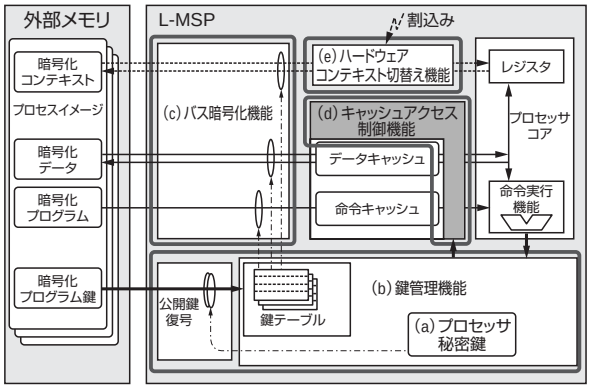

図3．L-MSPのブロック構成 ― 既存プロセッサに(a)～(e)の機能要素が追加され，動作することでプロセスの保護を行っている。

Configuration of L-MSP

単位で外部メモリから必要に応じて読み込まれてキャッシュメモリに格納され，それぞれ実行・参照される。データに書換えが発生すると，そのキャッシュラインがフラッシュされ，外部メモリに書き戻される。これらは，キャッシュメモリを制御するプロセッサハードウェアによって行われる。

一方，プログラムを二次記憶から読み出し，外部メモリに配置したり，外部メモリの容量が不足したときに一部を二次記憶に退避・復帰するのはOSの役割である。これらの機能が適切に働くことで，二次記憶に対して高速な外部メモリやキャッシュメモリを，複数のプロセスが効率的に共有して処理を進めることができる。これがメモリに関する資源管理である。

もう一方の資源管理であるCPU時間管理では，プロセスの実行中に割込み要求が発生して処理が中断すると，その時点のレジスタ値がOSによって外部メモリに一時保存される(コンテキスト切替え)。プロセス実行を再開するときには，保存したレジスタ値をOSがレジスタに復帰する。

このように，既存システムではOSがプロセスイメージに含まれる秘密情報，例えば暗号鍵や平文コンテンツを自由に読み取ることができる。たとえプロセスイメージに対してソフトウェア処理による暗号化などの対策を施したとしても，レジスタに保持されている処理の中間値は秘匿できない。

3.2 L-MSP アーキテクチャ

L-MSPアーキテクチャの構成を**図3**に示す。前章の攻撃(A)，(B)への対策と資源保護の両立のため，既存プロセッサに対して以下の追加機能を備えている。

- (a) チップに埋め込まれた秘密鍵
- (b) 暗号化とアクセス制御とを対応づける鍵管理
- (c) バス暗号化
- (d) パッケージ内部の平文プロセス情報のアクセス制御
- (e) ハードウェアコンテキスト切替え

3.2.1 ソフトウェアの暗号化と配布

ソフトウェアベンダーは共通鍵Kxを一つ選び，配布するプログラムの命令を暗号化し，更に公開鍵Kpでプログラム対応の共通鍵Kxを暗号化する(暗号化プログラム鍵)。配布先のプロセッサには製造時に，公開鍵Kpに対する秘密鍵Ksと公開鍵復号機能が組み込まれている。暗号化されたプログラムと暗号化プログラム鍵がユーザー端末に配布されるが，暗号化プログラム鍵を復号して共通鍵Kxを取り出せるのは対応するプロセッサに限られ，システムを管理するユーザーやOSであっても共通鍵Kxは取得できない。

3.2.2 暗号化プログラム鍵の読込みとバス暗号処理

実行時は配布と逆の手順がプロセッサ内部のハードウェアで行われ，動的に共通鍵Kxが読み込まれる。L-MSPの鍵管理機能は，保護対象プロセスと共通鍵Kxを関連付けてプロセッサ内部のテーブルに保持管理する。

共通鍵Kxに対応付けられたプロセスの実行によって命令のフェッチが始まると，外部メモリからキャッシュラインへの命令読込み時に共通鍵Kxによる復号が自動的に行われ，平文の命令がキャッシュメモリに読み込まれる。

プロセスがデータ暗号化機能を利用するときは，アドレス範囲とデータ用暗号鍵を指定する。アドレス範囲とデータ用暗号鍵もプロセス単位に管理される。データ処理についてはメモリへの暗号化書込みが追加されるだけで基本的には命令の処理と同様である。

3.2.3 アクセス制御

命令やデータの秘匿には，プロセッサ内部の平文情報に対するアクセス制御も必要である。L-MSPでは，キャッシュへの命令・データ読込み時にそのキャッシュラインの読込みに使われたプロセスの識別子がキャッシュタグに書き込まれる。そのキャッシュラインへのアクセス時には，現在実行中のプロセスの識別子とキャッシュタグに保持された識別子が照合される。不一致の場合は

キャッシュのミスヒットとして，外部メモリから再度対応するアドレスの内容が，共通鍵Kxによる復号処理なしに読み込まれる。

例えばOSが保護対象プロセスのメモリ領域にアクセスした場合，キャッシュ上にある平文情報にはアクセスできないが，外部メモリ上の暗号化された情報には常にアクセスが可能である。この機構により，OSはたとえ保護対象プロセスであっても外部メモリ上にある情報にはすべてアクセスが可能であり仮想記憶などの資源管理が可能である。同時に，その情報は暗号化されているので秘匿と資源管理が両立する。

直感的に言い表せば，プロセッサ外部の情報はプロセスごとに異なる鍵による暗号の壁で守られており，内部の情報はアクセス制御による壁で守られているといえる。

3.2.4　ハードウェアコンテキスト切替え　更に，従来OSで行われていた割込み発生時のコンテキスト切替えは，L-MSPではハードウェア機構により行われる。割込みが発生すると，レジスタの内容はプロセスごとに対応づけられた鍵により暗号化されて外部メモリに書き出される。この鍵はプロセッサ内部に保持され，外部に出力されることはない。実行の再開時には保存されたコンテキスト情報が復号され，レジスタに読み込まれて処理が継続される。OSはレジスタ値を直接操作することはなく，プロセスを指定して処理再開の特殊命令を発行するだけである。プロセスに対する操作がプロセッサによりカプセル化されているといえる。

L-MSPではこのようにして，命令，データ，コンテキストのプロセス3要素はすべてOSが知りえない暗号鍵により保護され，ブラックボックス化された状態でプロセスを実行できる。

3.3　試作検証

L-MSPはキャッシュメモリを持つプロセッサ適用可能な方式である。提案方式の妥当性を検証するため，当社の32ビットRISC（Reduced Instruction Set Computer）CPUコアMeP（Media embedded Processor）[3]を用いたテストベッドを構築し，機能を実証した。概略構成を**図4**に示す。

CPUコアのMePを中心に 公開鍵暗号（RSA1024），共通鍵暗号（AES128），鍵管理機能などをVerilog HDL（Hardware Description Language）で記述し，CPUとして機能するFPGA（Field Programmable Gate Array）環境を構築した。

この環境にプロセス切替え機能をL-MSP向けに修正したTOPPERS（Toyohashi OPen Platform for Embedded Real-time Systems）/JSP（Just Standard Profile）μITRONカーネル[4]を移植，カーネル上で複数の保護対象プロセスと非保護プロセスを並列実行させ，前述の保護機能がITRONのプロセス管理機能と共存可能であることを確認した。OSのプロセス切替え機能やキャッシュ管理機能の一部を変更することで，ITRON以外のLinuxやその他のオープンソースOSに対してもL-MSPの適用が可能である。

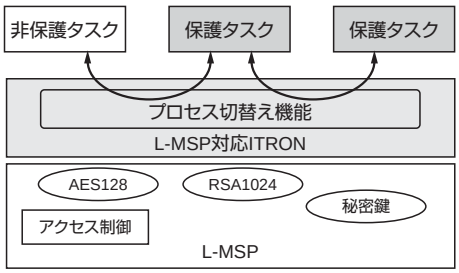

図4．試作システムのハードウェア／ソフトウェア構成 ― プロセス切替え機能に変更を加えることでITRONをL-MSPに適応させることができた。

Hardware and software diagram of trial system

4　あとがき

オープンソースOSの資源管理機能と共存可能なソフトウェア保護機能L-MSPを提案し，テストベッドを構築して既存オープンソースOSとの整合性を検証した。

ソフトウェア保護は著作権保護に限らず，今後，ソフトウェアの知的財産権保護，ネットワークエージェントやグリッドコンピューティングなど，様々なシステムの安全性を高めるために必要とされる技術である。L-MSPはソフトウェア保護の基盤を与える技術だが，実用的なシステムの構築には，高級言語からL-MSP上で動作する安全なソフトウェアを生成する技術や，安全性の検証技術も必要である。今後，これらの技術にも取り組んでいきたい。

文　献

(1) 橋本幹生，ほか．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．**45**，SIG3（ACS5），2004，p.1－10.
(2) 山田尚志，ほか．デジタルコンテンツ保護の現状と課題．東芝レビュー．**58**，6，2003，p.2－7.
(3) （株）東芝．MeP. <http://www.mepcore.com/>，（参照2005-04-12）.
(4) TOPPERSプロジェクト．<http://www.toppers.jp/>，（参照2005-04-12）.

橋本　幹生　HASHIMOTO Mikio

研究開発センター コンピュータ・ネットワークラボラトリー研究主務。ホームネットワーク，ソフトウェア保護技術に関する研究・開発に従事。電子情報通信学会，ACM会員。
Computer & Network Systems Lab.

春木　洋美　HARUKI Hiroyoshi

研究開発センター コンピュータ・ネットワークラボラトリー。ソフトウェア保護技術に関する研究・開発に従事。情報処理学会会員。
Computer & Network Systems Lab.

川端　　健　KAWABATA Takeshi

研究開発センター コンピュータ・ネットワークラボラトリー。ソフトウェア保護技術に関する研究・開発に従事。電子情報通信学会会員。
Computer & Network Systems Lab.