4Y-5

IXP425における暗号処理のオフローディング方式の実装と評価

渥美裕太† 天野桂輔† 笠原竜大‡ 村上智祐‡ 齋藤孝道†

明治大学† 明治大学大学院‡

1. はじめに

インターネットにおいてセキュリティプロトコルは広く利用されているが，その処理コストは高くシステム全体のパフォーマンスを低下させてしまう．解決策の一つとして，暗号処理に最適化されたモジュール（以下，暗号モジュールと呼ぶ）を内蔵したNP（Network Processor)の利用がある．暗号処理を暗号モジュールにオフロードすることで，システム全体のパフォーマンス向上が期待できるが，暗号モジュールが一つしかない環境化において，複数のユーザプロセスが同時に暗号モジュールにアクセスした場合，切り替えにより暗号モジュールを利用しているユーザプロセスの処理が中断され，別のユーザプロセスに制御が移ってしまうことがある．この場合，はじめに実行したユーザプロセスの暗号処理が遅延されてしまう．

そこで本論文では，OpenSSL [1]の暗号処理を，Intel社のNPであるIXP425 [2]に内蔵された暗号モジュールにオフロードし，ユーザプロセスを制御することで，効率的に暗号モジュールを利用する実装とその評価を行った．提案システムは論文[3]のものを改変したものであり，改変箇所として，暗号モジュールを利用しようとするユーザプロセスを待ち状態に遷移させるタイミングを変更した．

2. 開発環境

2.1 IXP425

本論文で使用するIXP425は，汎用CPUのXscaleコア，2個のEthenet用プロセッサ，WAN/VoIP用プロセッサ，PCIコントローラ，Queue Managerコントローラ，SDRAMコントローラなどから構成されている．

IXP425ではEthernet用の2つのNPEはNPEAとNPEBとそれぞれ呼ぶ．このうち，NPEBは暗号モジュールを内蔵し，共通暗号化方式としてDES，3DES，AESをサポートし，その暗号利用モードとして，CBCとECBがある．また，ハッシュアルゴリズムとしてSHA-1とMD5をサポートしている．

IXP425では，Intel社が提供するAPIであるAccess Libraryを利用して暗号処理機能を利用することができる．Access Libraryは，Xscaleコア，NPE，周辺装置などのデバイスにそれぞれ対応したAPIから構成されている．

2.2 OpenSSL

OpenSSLは，暗号モジュールに対応しており，アプリケーションからそれらを利用するためにENGINE APIを提供している．ENGINE APIは，暗号モジュールごとに用意されているライブラリを呼び出すことで，暗号処理を暗号モジュールへオフロードする． OpenSSLがサポートしていない暗号モジュールに対しては，それに対応するライブラリを用意することで，OpenSSLからの利用が可能となる．

OpenSSLからIXP425の暗号モジュールを利用する方法として，Intel社が提供するIXP4xxシリーズ向けのAPIであるAcces Libraryを用いる方法がある．Access Libraryは暗号モジュールを直接利用するためのAPIではなく，デバイスドライバの開発向けのAPIのみを提供する．そのため，OpenSSLからIXP425の暗号モジュールを利用するには，IXP425の暗号モジュール専用のデバイスドライバを実装する必要がある．

2.3 OCF

OCFは，OpenSSLからIXP425の暗号モジュールを含む様々なハードウェアへの透過的なアクセスを提供するAPIと，各種ハードウェアを制御するためのデバイスドライバから構成されたミドルウェアである．

OCFは，IXP425の暗号モジュール専用のデバイスドライバと，OpenSSLからそのデバイスドライバへアクセスするためのインターフェースを提供する．

An Implementation and its Evaluation of an Effective Off-loading Hardware Cryptographic Module on IXP425

†Yuta Atsumi, Keisuke Amano, Takamichi Saito

‡Ryuta Kasahara, Tomosuke Murakami

Meiji University(†), Graduate School of Meiji University(‡)

1-1-1 Higashimita, Tama-ku, Kawasaki-shi, Kanagawa, 214-8571, Japan(†)(‡)

{ee77093, ee77036}@isc.meiji.ac.jp, {ce06013, ce06039}@meiji.ac.jp, saito@cs.meiji.ac.jp

3. 提案システム

3.1 提案システム概要

複数のユーザプロセスが暗号モジュールに同時にアクセスしている状況で，ユーザプロセスの切り替えによる暗号処理の遅延をなくしたい．そこでOCFにリクエストを出した順に暗号モジュールを占有させる機能をOCFに追加した．

ユーザプロセスを制御するための構造体（以下，制御構造体）をユーザプロセスごとに生成する．OCF は各ユーザプロセスの制御構造体のメンバである list_head 構造体をつなぎ合わせることで，暗号モジュールを利用するユーザプロセスを管理する．また，制御構造体は wait キューをメンバに持ち，待ち状態に遷移するユーザプロセスをこれに繋いでいく．

論文［3］ではリクエストを出した直後にユーザプロセスを待ち状態に遷移させている．本論文では，他のユーザプロセスが暗号モジュールにオフローディングする直前でユーザプロセスを待ち状態に遷移させることによって，他のユーザプロセスが暗号モジュールを占有している間，自ユーザプロセスが暗号処理の直前までの処理を行えるようにした．

3.2 提案システム動作例

ここでは，提案システムに従ってOpenSSLの暗号処理をOCF経由で暗号モジュールへオフローディングする際の動作例を，図1中の番号と対応させて説明する．

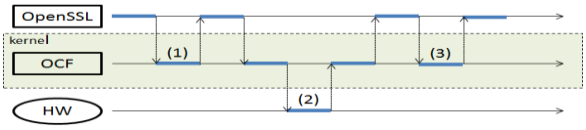

図1：提案システムの概要

(1) OpenSSLとOCFの間でセッションを確立する．すなわち，セッション識別子，暗号方式・暗号モード，暗号化鍵を共有する．セッションの確立時にリストが空ならば，ユーザプロセスをリストにつなぎ次の処理に移る．リストが空でなければ，ユーザプロセスをリストにつないだ後，waitキューにつなぎ，待ち状態に遷移させる．

(2) 暗号モジュールで暗号処理を行い，処理結果をOpenSSLへ返す．

(3) セッションの解放を行う．その際，自ユーザプロセスをリストから削除し，リストの先頭のユーザプロセスを実行可能状態に遷移させる．

4. 評価

4.1 評価方法

提案システムの実装の性能評価にはEVP APIを使用した計測用コードを用いた．ここで，計測用コードとは，本論文用に作成したもので，fork()により複数のユーザプロセスを生成し，IXP425に過負荷をかけるソフトウェアである．各ユーザプロセスは，平文データを3DESのCBCモードでN回暗号化する．ここで平文のデータサイズは，512byte，1024byte，4096byte及び8192byteの4種類とした．今回は，ユーザプロセス数8と20の場合で，Nを10から1000まで10ずつ値を増やしていき，それぞれの処理回数における処理速度を計測した．また，計測にはgettimeofday()を使用し，ユーザプロセスの生成から終了までを処理時間とした．

4.2 計測結果

それぞれの計測結果を図2に示す．

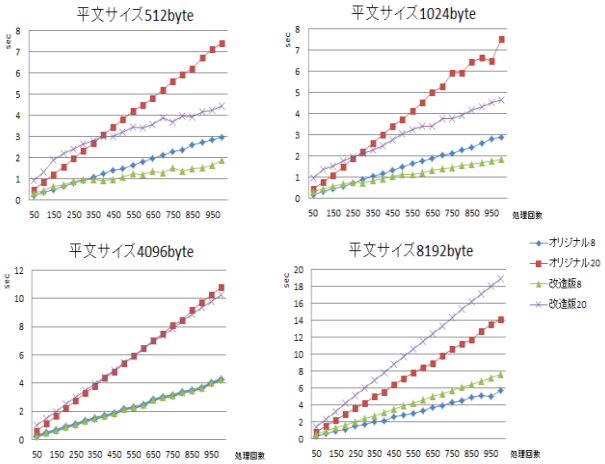

図2：計測結果

データサイズが大きい場合，提案システムの処理時間はオリジナルの OCF に比べて長くなった．データサイズが大きくなることで，スラッシングが 起きているのが原因だと考えられる．

論文［3］のシステムの改変を行ったが，処理速度の向上は見られなかった．

5. まとめ

データサイズが小さいときはオリジナルに比べて提案システムのほうが処理時間は短くなったが，データサイズが大きくなるとオリジナルより処理時間が長くなる結果となった．

謝辞

本研究の成果の一部は，科学研究費補助金（課題番号22700086）の助成を受けたものである．

参考文献

[1] http://www.openssl.org.

[2] http://www.intel.com/design/network/.

[3] 齋藤，大釜，羅，杉浦， IXP425 における暗号処理の効率的なオフロード方式の実装と評価，情報処理学会論文誌， Vol.51 No.9 (2010)，pp1530-1541.