3W-2

Android 端末における暗号モジュールの利用と評価

金子洋平† 天野桂輔‡ 齋藤孝道†
明治大学† 明治大学大学院‡

1. はじめに

SSL (Secure Socket Layer) /TLS (Transport Layer Security) を用いた通信では，暗号処理やハッシュ処理などを伴う．これらの処理は汎用プロセッサにとって負荷が高く，システム全体のパフォーマンス低下や通信のスループット低下などの原因となる．これらの負荷を軽減する方法として，暗号処理に最適化されたハードウェアモジュール(以下，暗号モジュールと呼ぶ)を搭載したプロセッサを利用し，暗号処理をオフロードする方法がある．

Android 端末に搭載される CPU は，一般的なサーバ等に搭載される CPU よりも性能が低く，SSL/TLS のボトルネックとなる．Android で，SSL/TLS の通信のために利用される OpenSSL は，様々なハードウェアアクセラレータに対応しているが，直接アクセス出来ない暗号モジュールも存在する．そこで，OpenSSL から暗号モジュールにアクセスするために暗号ミドルウェアを利用する方法がある．

本論文では，暗号モジュールを搭載したプロセッサとして TI 社の AM3358 プロセッサ[1]を利用し，暗号ミドルウェアを介して，Android の標準ブラウザの SSL/TLS 通信に伴う暗号処理のオフロードを行うことを試みた．暗号処理をオフロードすることにより，SSL/TLS 通信の負荷を低減し，スループットの向上を目指す．また，HTTP リクエスト/レスポンス時間の測定をすることで，その評価をした．

2. 計測環境

2.1. AM3358 プロセッサ

AM3358 プロセッサは，TI 社の開発した ARM Cortex-A8 アーキテクチャベースのシングルコアプロセッサである．稼働させることが可能な OS は，Linux，Android，Windows Embedded CE である．本論文では，Android を利用した．

暗号モジュールは，AM3358 プロセッサに１つ搭載されていて，共通鍵暗号化方式やハッシュ関数に対応しており，AES，SHA，MD5 のアルゴリズムと RNG (Random Number Generator) を利用可能である．

2.2. Android

Android は，Linux カーネル，DlvikVM やアプリケーションフレームワークなどのミドルウェア，Web ブラウザなどのアプリケーションから構成される．

Android の Linux カーネルでは，通常のハードウェア制御に加えて電源管理，メモリ管理などを行う．また， Linux 用のデバイスドライバは，そのまま Android でも利用可能である．したがって，Linux で動作するハードウェアを Android で動作させることは容易である．また，Android は様々なライブラリを含み，SSL 通信のために OpenSSL ライブラリを含んでいる．

2.3. OCF と OpenSSL

AM3358 プロセッサの暗号モジュールで暗号処理を行うため，OpenSSL が提供する ENGINE API を利用する．ENGINE API は，AM3358 プロセッサの暗号モジュールに直接アクセス出来ないため，OCF(OpenBSD Cryptographic Framework)[2]を利用して暗号モジュールへのオフロードを行った．

OCF は，ハードウェアアクセスラレータが提供する暗号処理機能を利用するための共通のインターフェースを提供する API と，ハードウェアアクセラレータ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．

Android に含まれる OpenSSL ライブラリは，ENGINEAPI や OCF を利用できない設定になっている．このため，これらの機能を利用するために OpenSSL の実行環境を再構築した．

ブラウザから OpenSSL や OCF を通して暗号モジュールにアクセスするまでの関係を図 1 に示

An Utilization of Cryptographic Module of Android Device and its Evaluation
†Yohei Kaneko, Takamichi Saito
‡Keisuke Amano
Meiji University(†), Graduate School of Meiji University(‡)
1-1-1 Higashimita, Tama-ku, Kawasaki-shi, Kanagawa, 214-8571, Japan(†)(‡), ee97040@isc.meiji.ac.jp, ce16002@meiji.ac.jp, saito@cs.meiji.ac.jp

す.

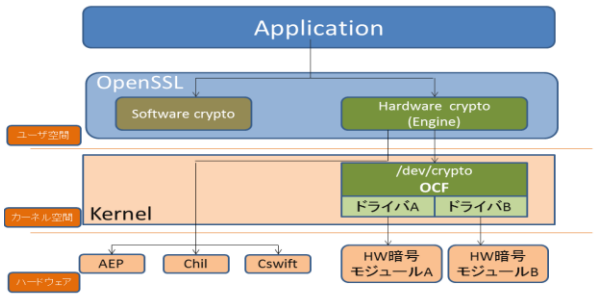

図 1:オフロード概要

3. 評価

3.1. 評価方法

評価は，Android 標準ブラウザからファイルを SSL サーバにアップロードし，その処理時間を計測することで行った．アップロードするデータは，70 バイトから 18,948,662 バイトまでのビットマップファイルである．

時間の計測には，Navigation Timing API[3]を利用した．Navigation Timing API は，ブラウザ Web サイトの画面を表示するまでに，各処理の開始時間と終了時間を取得することの出来る JavaScript の API である．本論文では，アップロード時間を計測するために，http リクエストからレスポンスまでの時間を測る計測コードを作成した．

計測は，ブラウザからファイルのアップロードページにアクセスし，ファイルのアップロードを行った後，計測コードを実行し時間を計測した． また，計測は各ファイルにつき 5 回行いその平均の値をとった．

3.2. 計測結果

図 2 で示す計測結果について表 1 に示す．計測結果より，Android 標準ブラウザの SSL 通信に伴う暗号処理を暗号モジュールにオフロードすることで高速化できた．

表 1 ：アップロード時間

ファイルサイズ（バイト）	オフロードあり（ms）	オフロードなし（ms）
70	402	362
57,654	442	436
2,359,350	1,206	2,592
3,240,054	1,480	3,086
6,350,454	2,512	5,320
14,288,454	5,581	10,564
15,046,694	6,331	11,597
18,207,342	6,704	13,293
18,948,662	7,784	13,848

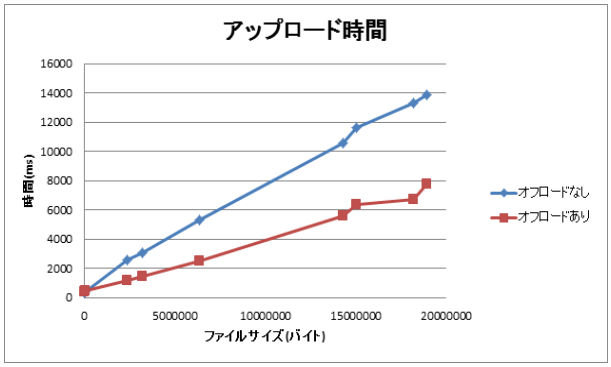

図２：アップロード時間

しかし，ファイルサイズが小さい場合には，オフロードによる処理速度の低下を引き起こす．これは，ファイルサイズが小さい分，暗号処理の高速化の影響が少ないので，オフロードによって生じるオーバヘッドの割合が通信時間において高くなってしまうためである．

処理時間の差は，ファイルサイズが大きくなるにつれて大きくなっており，最終的にファイルサイズが 18,948,662 バイトの時には，暗号モジュールにオフロードしていない場合に比べ約 44%の通信速度の高速化に成功している．

4. まとめ

本論文では，Android 端末に搭載されている暗号モジュールに Android 標準ブラウザの SSL/TLS 通信に伴う暗号処理のオフロードを行った．その評価を行った結果，暗号モジュールに暗号処理をオフロードしない場合に比べ高速化することができた．

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] AM3358 Processor, http://www.ti.com/product/am3358

[2] OCF, http://ocf-linux.sourceforge.net/

[3] Navigation Timing API, https://dvcs.w3.org/hg/webperf/raw-file/tip/specs/NavigationTiming/Overview.html