5C－6

モーバイル環境に適した圧縮／暗号化通信方式の WinSock API への適応方法

近藤毅*　中田幸男*　高橋泰弘*　村上弘真**

* (株) 日立製作所システム開発研究所　** (株) 日立製作所ソフトウエア開発本部

1．はじめに

モーバイル環境、特に無線通信では、盗聴を防ぐために通信データの暗号化、並びに通信速度が遅いために通信データの圧縮が必要とされる。筆者らは上記二点を満足するモーバイル環境に適した通信方式を提案[1]してきた。

上記提案方式をTCP/IP上の共通インタフェースであるWinSock APIにおいてデータ圧縮/暗号機能と通信制御機能とを備えたセキュア通信ライブラリとして既存AP無改造で実現する方法について報告する。

2．既存AP無改造方針

2.1 WinSock API取り込みの原理（図1）

既存APとWinSock APIを提供するソケットライブラリとの間の呼び出し関係を既存AP動作に影響することなく変更することで、セキュア通信ライブラリが既存APからの送信データをいったん取り込み、圧縮/暗号化処理を行った後、ソケットライブラリを呼び出すことにより既存AP無改造で通信路の圧縮/暗号化を実現する。

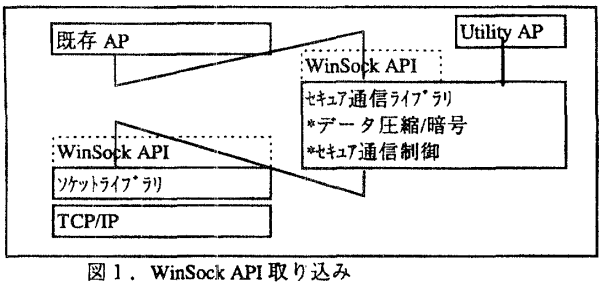

図1．WinSock API取り込み

2.2 セキュア通信ライブラリィの位置づけ

セキュア通信ライブラリは、既存APとソケットライブラリ間に組み込まれる。既存APを改造しないためにセキュア通信ライブラリはWinSockAPIを既存APに対してみせる。また、TCP/IP処理部を変更しないために、セキュア通信ライブラリはソケットライブラリが提供するWinSock APIを用いて通信を行う。

なお、既存APは圧縮や暗号に関するインタフェースを持たないため、Utility APを設けセキュア通信ライブラリとの間に圧縮/暗号のパラメタ（暗号鍵、ユーザID）等の操作をユーザに提供する。

2.3 WinSockAPI取り込みにおける要求条件

既存APのソケットライブラリへの呼び出しをセキュア通信ライブラリへの呼び出しに変える方法の検討で考慮すべき要件を表1に示す。

表1．WinSock API取り込み方法の要件

項番	項目	概要
1	高速性	AP性能劣化の少なさ
2	透明性	(1)既存APの動作不変性 (2)APのユーザ操作不変性 及び、追加操作の簡易性
3	適応性	(1)適応APの多様性 (2)適応環境の多様性 (3)異プラットホームへの移植性
4	実現容易性	(1)低開発コスト (2)保守性

3．実現方法

3.1 方法案

一般的なAPI取り込み技法として、(1)ローダ注入(2)ターゲットコード書き換え(3)ダミーDLLの3つの方法が知られている[2]。これらの方法にさらに新しい方法をも加えて、以下に示す4つ方法で我々が提案してきたモーバイル環境に適した通信方式への適応について検討した。

(1)方法1：呼び出し側書き換え方法（図2）

既存APをデバッグモードで動作させ、既存APにローダを注入した後、APを起動する。注入したローダによりセキュア通信ライブラリがロードされ、その初期化処理でソケット関数の呼び出し先をソケットライブラリからセキュア通信ライブラリへと書き換える。

(2)方法2：呼び出し先書き換え方法（図3）

ソケットライブラリ関数の先頭コードをセキュア通信ライブラリ関数へのJumpコードに書き換える。また、セキュア通信ライブラリからソケットライブラリを呼び出す場合は、関数の先頭コードを元に戻した後呼び出す。そして、ソケットライブラリ呼び出し終了時に再度先頭コードをセキュア通信ライブラリ関数へのJumpコードに書き換える。

(3)方法3：リンケージ情報書き換え方法（図4）

WinSock APIを提供するセキュア通信ライブラリを他のライブラリと重複しない適当な名称で作成しておく。

既存APのソケットライブラリとのリンケージ情報をセキュア通信ライブラリにリンクするように書き換える。

(4)方法4：名称書き換え方法（図5）

WinSock APIを提供するセキュア通信ライブラリをリネーム前のソケットライブラリ名称で作成しておく。そしてリネーム前のソケットライブラリ名称を異なる名称にリネームする。するとシステムローダにより既存APはセキュア通信ライブラリとリンクする。

Adaptation Technique of Communication Method with Data Compression and Encryption for Mobile Computing Environment to the Winsock API

Takeshi Kondo*,Yukio Nakata*,Yasuhiro Takahashi*,Hiromasa Murakami**

*Systems Development Laboratory,Hitachi,Ltd.　**Software Development Center,Hitachi,Ltd.

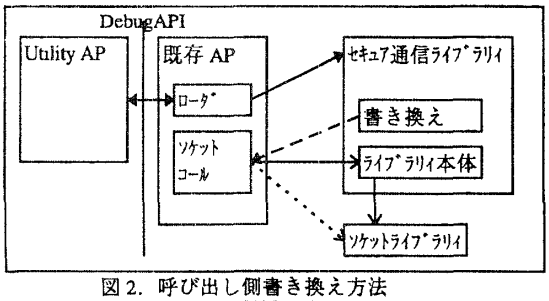

図2. 呼び出し側書き換え方法

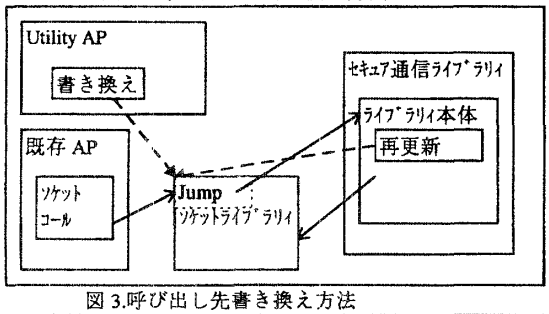

図3.呼び出し先書き換え方法

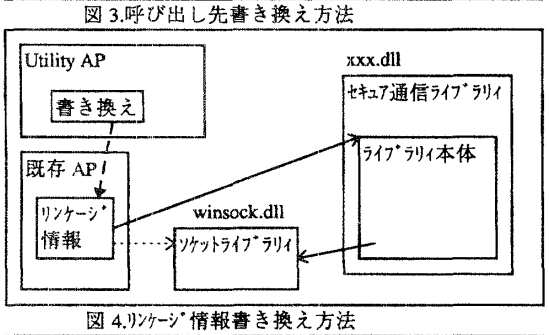

図4.リンケージ情報書き換え方法

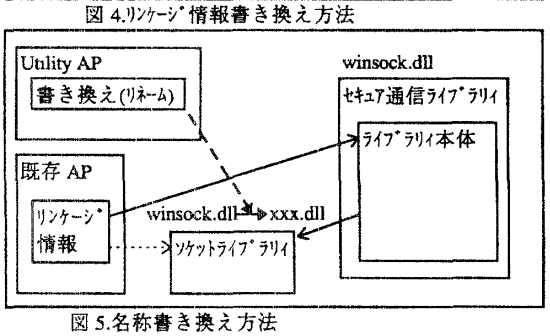

図5.名称書き換え方法

3.2 評価

要求条件毎に方法を評価する。評価結果は表2に示す。

(1)高速性

方法1では、デバッグAPIを使用し対象APを監視下に置くためにオーバヘッドが生じる。方法2では、ソケットライブラリの内容を書き換えているため排他制御が必要であり、例えば、マルチスレッド環境では権利をもったスレッドで待ちが生ずると他のスレッドが停止したままになるため性能が大きく低下するという欠点がある。
方法3、方法4は上記の様な性能劣化要因はない。

(2)透明性

AP動作の透明性に関しては、どの方法でも満足できる。ただし、ユーザ操作を考察すると方法4を除いて、次のような欠点が生ずる。AP毎に適応される1から3の方法ではユーザが既存APをインストールするときか、または起動するときにUtility APも起動するというセキュア通信ライブラリを適応させるための操作が必要になる。

(3)適応性

方法1は、デバッグインタフェースが提供されていることが前提であり適応性は高くない。また、デバッガとして動作するAPには適応できない。方法2は、書き換えるJump命令の長さ以内の関数エントリを持つものでは、次関数領域を破壊するため適応できないという欠点がある。また、AP プロセス間のガードが堅固なシステム環境には、ソケットライブラリの書き換えが拒絶されるため適応できない。方法3は、32bit/16bit環境の違いによりリンケージ情報が異なるため、書き換えロジックの個別対応が必要である。方法4は、リネーム機能を使用するため、APや環境に対して適応性がある。

(4)実現容易性

方法3、方法4ともに書き換え回数が1回で済み、かつ、既存AP実行時に書き換えないため書き換えに必要な処理も少ない。この様に簡素な方法であるため開発工数が少なくて済む。しかし、両方法ともに予め書き換える方法であるため、書き換えたファイルが他の手段によって書き換えられたり元に戻された時の対応が必要である。この対策として書き換え済みファイルの監視機構とリカバリ機構を設けるが、その開発量は多くない。特に、方法4はソケットライブラリだけを対象とするため有利である。

保守性に関しては、方法3と方法4とがソケットライブラリの変更の影響を直接受け、作り直しとなる点が欠点であるが、その頻度は多くないと考えられる。

(5)総合評価

以上により本要求条件では方法4の置き換え方法が最も適しているといえる。

表2. WinSock API取り込み実現方法比較

方法 / 項目	方法1	方法2	方法3	方法4
高速性	○	×	◎	◎
透明性	○	○	○	◎
適応性	△	△	○	◎
実現容易性	○	△	○	○
総合評価	△	×	○	◎

4. WinSock API取り込み時の共通課題と対策

ソケットライブラリ以外の関数が影響する場合が課題である。具体的には、ソケットを他のAPへ継承する関数は、ソケットライブラリが提供しない関数にも関わらず圧縮暗号通信動作に影響する。それは、ソケット毎に暗号鍵を付加するため、ソケットを継承されたAPのコンテクストで動作するセキュア通信ライブラリで暗号鍵を把握する必要があるためである。本課題の対応としてソケットへの付加情報は、異なるプロセスから共通的に参照できる機構を設けた。

5. まとめ

本稿でモーバイルに適した圧縮/暗号化通信方式を既存AP無改造でWinSockAPIに適応するには名称書き換え方法が最も適していることを示した。また、実現上の課題に関する対応策を提示した。

参考文献

[1]高橋 他、「モーバイルに適した圧縮/暗号化通信方式（その1）ﾊﾟｹｯﾄ圧縮/暗号化通信方式」情報処理第52回全国大会

[2]Matt Pietrek;"Windows 95 System Programming SECRETS" IDG BOOKS,November 1995