実用性を重視した暗号通信方式の提案

増田　真也[†]　　渡邊　晃[†]

近年，ネットワークにおけるセキュリティ上の脅威が問題となっており，ネットワークセキュリティ技術が重要視されてきている．暗号技術を用いたセキュリティとして，IP 層での技術である IPsec は，セキュリティは強靭なものの，NA(P)T やファイアウォールとの相性が悪い，フラグメントが発生するなど多くの課題がある．そこで本論文では，既存システムに影響を与えず，またオリジナルパケットとサイズを変えないまま，本人性確認（正当な相手であることの保証）とパケットの完全性保証（パケットが改竄されていないことの保証）を行うことができる暗号通信方式について提案する．本方式では，パケット長が変化しないため十分なスループットが期待できる上，NA(P)T やファイアウォールを通過できる実用的なシステムを構築できる．

The Proposal of Practical Cipher Communication Systems

SHINYA MASUDA[†] and AKIRA WATANABE[†]

In recent years, the network security has serious problem so that people place a special emphasis on the network security technology. There is IPsec which is the technology of the IP layer as the security which cipher technology was used for. But it has a lot of subjects to solve. Those are the compatibility problem with the NA(P)T and the Fire Wall, and it occurs fragment. In this paper, we propose about the cipher communication systems. It's able to do Personal Authentication (The guarantee which is a proper companion) and Integrity Guarantee (The guarantee what the packet isn't altered) which doesn't give any influences to existed systems, and it doesn't change a packet size. This system is able to expect enough throughputs and not change a different original packet size, and it's able to build the practical systems passing through the NA(P)T and the Fire Wall.

1. はじめに

近年, ネットワークにおいて様々なセキュリティ上の脅威が問題となっており, それに伴いネットワークにおけるセキュリティが重要視されてきている．その中でも，ネットワーク自体のセキュリティを確保するネットワークセキュリティ技術は, 利用するアプリケーションを意識することなく安全を確保できることから, ネットワークの根本的なセキュリティ対策として非常に有効な手段とされている．

ネットワークにおけるセキュリティ上の脅威はインターネットだけでなく, イントラネットにも存在する．実際，企業ネットワークにおける不正アクセス被害のうち半数近くが企業内部からのものであるという報告が出されている[1)]．このようなことから，インターネットのみならずイントラネットも含めた総合的なセキュリティ対策を適用したいというニーズが最近増加している. イントラネットを視野に入れたネットワークセキュリティは，NA(P)Tやファイアウォールなどの既存システムとの相性を十分に考慮する必要があるが, これらが普及を阻害する原因となることがしばしばある．セキュリティ技術を導入するがゆえに, 既存システムに特殊な処理・設定を行ったり, 別の新たな機器を導入したりしなければならないことから, 導入を見送るというケースは少なくない．ゆえに, 既存システムに影響を与えることなく容易に導入できることが, 実用性を考える上で重要なポイントとなる．

既存のネットワークセキュリティ技術の代表として, IPパケットの暗号方式などを規定しているIPsecが挙げられる[2)~5)]. IPsecの中でも暗号通信方式について規定しているのがESPで, 盗聴を防止する暗号化以外に, なりすましを防止する本人性確認（正当な相手であることの保証）や改竄を防止する完全性保証（パケットが改竄されていないことの保証）などの機能を提供している．しかしIPsecは，セキュリティは強靭なものの, パケットの暗号化や完全性保証によってNA(P)Tやファイアウォールを通過できなくなることや, パケット長の増加によってフラグメントが発生することなどが問題とされている．

†名城大学理工学部
Faculty of Science and Technology, Meijo University

また, 既存システムに影響を与えないよう暗号化範囲を規定し, パケット長を変えずに暗号化を行う方式が提案されているが[6]（以下，従来置換方式），TCP/UDPチェックサム[7)~9)]の書き換えを行うNA(P)Tを通過できないことや，本人性確認とパケットの完全性保証を考慮していないという課題がある.

IPsec と従来置換方式以外において，ネットワークセキュリティとして暗号通信方式を規定しているものはほとんどない. そこで，本論文では従来置換方式の利点をそのまま継承し，本人性確認とパケットの完全性保証も確実に行える暗号通信方式について提案する. 本方式では, パケット長が変化しないため十分なスループットが期待できる上，NA(P)T やファイアウォールを通過できる実用的なシステムを構築できる.

提案方式の有効性を確認するために, 試作モジュールを開発し, ドライバによる動作検証と性能評価を行った. 試作モジュールの処理速度を計測した結果, 100Mbps の通信環境で問題とならない性能であることが確認できた．また，暗号化/復号などの各機能としてモジュール分割したサブモジュールの処理速度を測定することで, どの処理が全体の何割を占めているのかを調べ, 処理速度の向上に必要なものを検討した.

提案方式は全て IP 層で行う．実装方法として，IP 層の詳細な処理フローに関する情報が多い FreeBSD のカーネル内にモジュールを組み込む方法を検討した.

本論文では以下，2 章で既存の暗号通信方式とその課題，3 章で実用性を重視した暗号通信方式の要件と提案方式，4 章でモジュールの試作と実装方法，5 章で既存技術との比較検討と，試作モジュールの評価，6 章でまとめと今後の課題について述べる.

2. 既存の暗号通信方式とその課題

ネットワーク自体の安全性を確保する技術である IPsec は，IP 層のセキュリティとして盛んに研究が行われている．IPsec ESP には，トランスポートモードとトンネルモードの 2 つのモードがあり，前者は End-to-End の IPsec 通信を適用する際に利用するモードで, 後者は主にセキュリティゲートウェイ間で IPsec 通信を適用する際に利用するモードである.

しかし実際のほとんどは，ESP トンネルモードを, インターネットを利用して分散した拠点を結ぶVPN（Virtual Private Network）[10)]の構築手段として利用する場合である. これは，IPsec が複数の仕様で構成されていることが原因で起こる相互接続性の悪さや, 汎用性があり自由度が高いために起こる設定の複雑さ, そして以下で述べるNA(P)Tやファイアウォールとの相性問題などが原因で, VPNのような特定の用途以外では利用が困難であるからだと考えられる.

そこで，以下では VPN 以外に考えられる用途を例に挙げ, トランスポートモードとトンネルモードのいずれにおいても NA(P)T やファイアウォールの通過が不可欠であることを示す.

WWWを代表とするクライアント/サーバシステムに IPsec を適用する場合，サーバ側にはセキュリティゲートウェイを設置するので, 自ずとトンネルモードを利用することになる. このとき, プライベートアドレスのクライアントが外部サーバにアクセスする場合はNA(P)T やファイアウォールの通過は必須である.

近年急増化している P2P（Peer-to-Peer）ネットワークでは, 個人間の通信が主体となっており，IPsec を適用する場合は End-to-End であるトランスポートモードの利用も考えられる. このとき，NA(P)T やファイアウォールを経由することは十分にあり得る（**図 1**）.

また，イントラネットで IPsec を適用する場合は両モードの利用が考えられる. 企業ネットワークでは, 部門間にファイアウォールを設置することが多く, ファイアウォールの通過は必須である.

このように，いずれのモードにせよ NA(P)T やファイアウォールの通過は欠かせない. しかしながら，IPsec はこれら既存システムとの相性が問題とされている.

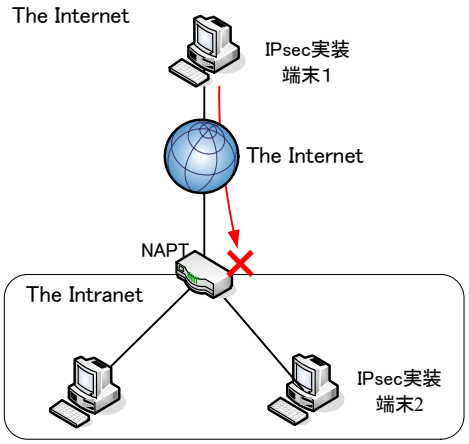


図 1 トランスポートモードで NA(P)T を経由する例
Fig. 1 Example of through the NA(P)T on transport mode.

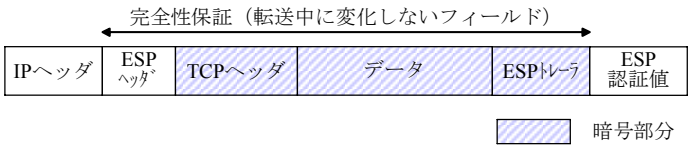


図 2　ESP トランスポートモードのパケットフォーマット（TCP の場合）
Fig. 2　Packet format of ESP transport mode (case of TCP).

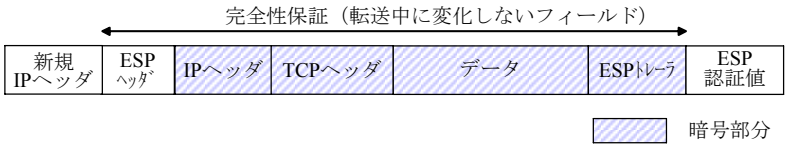


図 3　ESP トンネルモードのパケットフォーマット（TCP の場合）
Fig. 3　Packet format of ESP tunnel mode (case of TCP).

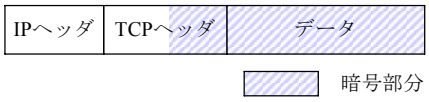


図 4　従来置換方式のパケットフォーマット（TCP の場合）
Fig. 4　Packet format of existent replace system (case of TCP).

ESP トランスポートモードの場合，**図 2** のようにポート番号が暗号化により見えなくなるため，そのパケットがどのような用途に用いられるかがファイアウォールで判別できない．その結果，ファイアウォールでは全ての IPsec の通過を禁止してしまうことが多い．また，TCP/UDP チェックサムが暗号化範囲・完全性保証の範囲に含まれているためチェックサムの書き換えを行う NA(P)T の通過はできない．この問題については NAT-Traversal として，UDP ヘッダでカプセル化することで NA(P)T を通過させる“UDP Encapsulation of IPsec Packets”がインターネットドラフトとして公開されており[11]，有効な手段として普及しつつあるが，一部の問題が未解決であり，NAT-Traversal の結果新たに生じる問題もある．その他の課題として，ヘッダ等の付加によるオーバヘッドやフラグメントの発生が挙げられる．

ESP トンネルモードの場合も，**図 3** のようにポート番号が暗号化されているため，トランスポートモードと同様に，ファイアウォールを通過できない場合が多い．このモードにおいてカプセル部分の IP アドレスを変換する NAT は，IP ペイロードにアドレスに依存したデータが存在しなければ通過できる．しかし，多くの場合が NAT ではなくポート番号の変換も行う NAPT を利用している．NAPT の場合は，変換時に TCP/UDP チェックサムの書き換えを行うのでトランスポートモードと同様に，NAPT を通過できない．更に，トンネルモードの場合はカプセル化を行うので，その分トランスポートモード以上にオーバヘッドやフラグメントが発生する懸念がある．

従来置換方式では，既存システムに影響を与えないよう暗号化範囲を規定しているが，**図 4** のように TCP/UDP ヘッダの後半部分以降を暗号化しているため，TCP/UDP チェックサムの書き換えを行う NA(P)T を通過できない．また，この方式はパケット長を変えない暗号化を実現しているが，本人性確認とパケットの完全性保証を考慮していないため，なりすましや改竄の恐れがある．

3. 実用性を重視した暗号通信方式

3.1 要　　件

2 章で述べた課題を踏まえて，実用性を重視した暗号通信方式の要件を以下のように整理する．

（1） 既存システムに影響を与えることなく容易に導入できる．
（2） セキュリティ技術の導入によって発生するオーバヘッドやフラグメントを極力抑えることができる．
（3） パケットの暗号化だけでなく，本人性確認と完全性保証も行うことができる．

3.2 提 案 方 式

実用性を考える上で最も重要なことが，3.1 で示した要件（1）である．この要件を満たすには，NA(P)T やファイアウォールの通過が欠かせない．また，高スループットを実現するためには要件（2）がポイントとなる．

本論文では従来置換方式の利点をそのまま

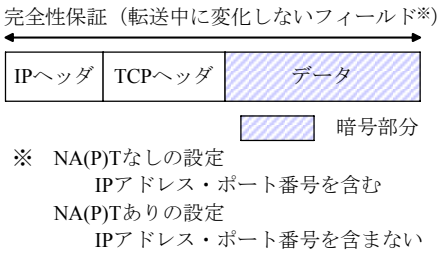


図 5 提案方式のパケットフォーマット（TCP の場合）
Fig. 5 Packet format of proposal system (case of TCP).

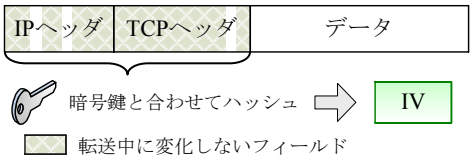


図 6 IV（Initialization Vector）の生成
Fig 6. Making of IV (Initialization Vector).

継承し，従来置換方式の課題である NA(P)T の通過を可能にしつつ, パケット長を変化させないまま, 本人性確認とパケットの完全性保証も実現する方式を提案する．本方式では，3.1 で示した要件を全て満たすことができる．尚，本方式は全て IP 層で行う．

提案方式では, **図 5** のようにユーザデータ部分のみを暗号化の対象とする．すなわち，TCP/UDP ヘッダをあえて暗号化範囲から外すことで，NA(P)T やファイアウォールを通過できるようにする．また，パケット長を変えずに暗号化するために, 任意長のデータを暗号化できるブロック暗号の CFB モードを用いる．よって, 提案方式によるフラグメントは発生しないため， 高スループットが実現できる．

提案方式では, 図 5 のように設定によってパケットの完全性保証に関わる処理を分ける．どちらの設定にするかは, 想定する環境で異なる．NA(P)T を経由しない特定の環境において暗号通信を適用したい場合は，NA(P)T なしの設定にする．この場合は，IP アドレスとポート番号を完全性保証の範囲に含める．尚，この設定では本方式を独立して使用することができる．NA(P)T を経由する場合も含めて暗号通信を適用したい場合は，NA(P)T ありの設定にする．この場合は，IP アドレスとポート番号を完全性保証の範囲から外す．IP アドレスとポート番号については, 他の技術との組み合わせで保証する方式を提案する．

① NA(P)T なしの設定

暗号化/復号の際は，暗号鍵とは別に IV（Initialization Vector）と呼ばれるブロック暗号

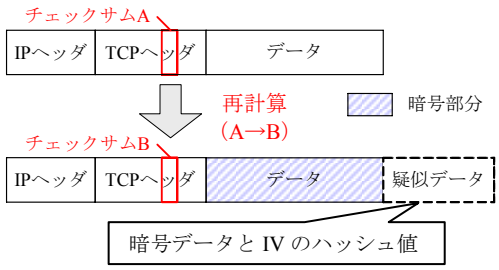


図 7 チェックサムの再計算（TCP の場合）
Fig 7. Recalculation of Checksum (case of TCP).

の先頭ブロックで用いる初期値を与える必要がある．IV は暗号化/復号で同じ値であり，使用する度に異なる値である必要がある．また，第三者には分からない値を用いることが望ましい. そこで, **図 6** のように IP ヘッダ, TCP/UDP ヘッダで転送中に値の変化しないフィールド（IP アドレスとポート番号を含む．TCP/UDP チェックサムを除く．）と暗号鍵を合わせたハッシュ値を IV とすることで, 暗号化/復号で同じ値であり，パケットごとに異なる値となる IV を生成することができる．IV 生成には鍵情報を含めているため，第三者に IV が知られることはない．更にこの IV は，以下で述べる本人性確認とパケットの完全性保証の役割も果たす．

IPsec ESP ではヘッダを追加することで本人性確認とパケットの完全性保証を行っているが, 提案方式ではパケット長を変えないためヘッダの追加はせず，TCP/UDP チェックサムを用いることで本人性確認とパケットの完全性保証を行う．本来 TCP/UDP チェックサムは，データの誤り検出を行うために用いるが, ここでは IP 層に独自の処理を追加することで本人性確認とパケットの完全性保証を行う．

送信側ではデータの暗号化を行う前に，TCP/UDP チェックサムを検証して，上位層（TCP/UDP）から渡されたパケットが正しいことを確認したら，データの暗号化後，**図 7** のように暗号データと IV を合わせたハッシュ値をデータの一部と見なして（疑似データと呼ぶ）チェックサムの再計算を行う．受信側ではデータの復号を行う前に, 同様の方法で作成した疑似データを含めて計算したチェックサムを検証し, 復号後にチェックサムの再計算を行って上位層（TCP/UDP）に渡す．この方式により，暗号データと IV 生成に用いたフィールドの完全性を保証することができる（**図 8**）．

パケットを改竄した場合，改竄者は TCP/UDP チェックサムを再計算しようとするが, 正当な者しか疑似データを作ることはでき

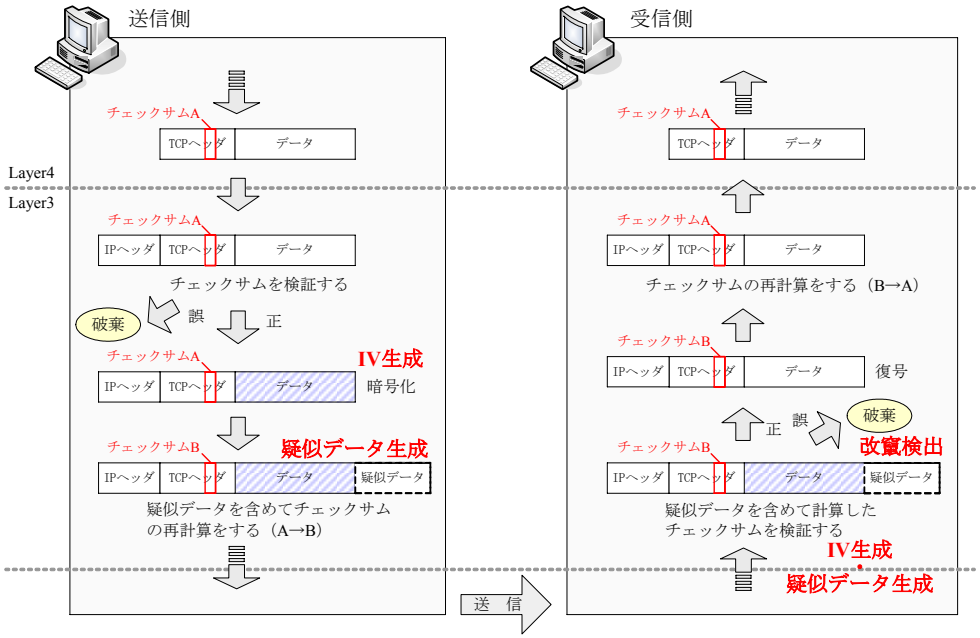


図 8　改竄検出の流れ（End-to-End の場合）
Fig. 8　Flow of alteration detection (case of End-to-End).

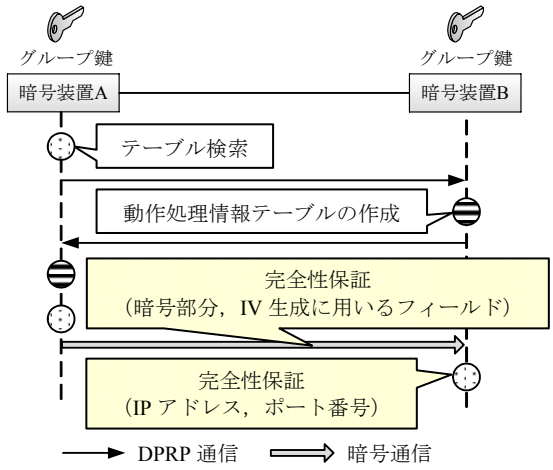


図 9　DPRP と組み合わせたパケットの完全性保証
Fig. 9　Integrity Guarantee of packet combined with DPRP.

ないので，改竄時に再計算を行うことはできない．この方式では，疑似データによって正当な相手であることが保証されるので本人性確認も実現する．

このように，疑似データを含めた TCP/UDP チェックサムの計算を行うことで，ヘッダの追加をせずに本人性確認とパケットの完全性保証が実現する．

②　NA(P)T ありの設定

NA(P)Tを経由する場合はIPアドレスとポート番号が変化するため，これらをIV生成の範囲から外す必要がある．NA(P)Tでは，チェックサムの書き換えが行われるが，変換部分の差分計算を行うだけであり受信側で行うチェックサムの検証には影響を与えない[12]．この設定では別の方法でIPアドレスとポート番号の完全性保証を行う必要がある．例えば，**図 9** のように暗号通信に先立って行うDPRP（Dynamic Process Resolution Protocol）[13] と本方式を組み合わせることで，IPアドレスとポート番号の完全性保証を実現できる．

DPRP は，暗号通信に先立って暗号化/復号などの動作処理情報を記したテーブルを作成する．暗号通信は，そのテーブルを基に行う．動作処理情報テーブルには IP アドレスとポート番号の情報が含まれており，IP アドレスとポート番号のハッシュ値を検索キーとしてテーブル検索を行う．すなわち，受信側でテーブル検索にヒットしたら IP アドレスとポート番号は改竄されていなかったことが保証される．尚，DPRP は手段のひとつであり，この考え方を用いれば他の手段も考えられる．

4.　試作モジュールの開発

提案方式の有効性を確認するために，試作モジュールを開発し，ドライバによる動作検証と性能評価を行った．本章では試作したモジュールの仕様と構成，および実際にモジュールをカーネルに組み込む方法について記述する．

4.1　試作モジュールの仕様と構成

4.1.1　仕　　様

試作したモジュールの仕様を**表 1** に示す．パケットの暗号方式は，平分と暗号文をそのまま置き換えるメッセージ置換型で，暗号アルゴリ

表 1　試作モジュールの仕様
Table 1　Specification of trial production modules.

項　目	内　　容
暗号アルゴリズム	AES（CFB モード）
暗号方式	メッセージ置換型
鍵長	256 ビット
ハッシュ関数	MD５

表 2　試作モジュールの構成と機能
Table 2　Composition and functions of trial production modules.

モジュール	機　　能
暗号処理	メインモジュール．各サブモジュールを呼び出し，一連の処理を組み立てる．
IV 生成	IP ヘッダ，TCP/UDP ヘッダで転送中に変化しないフィールドの鍵付きハッシュを生成する．
暗号化/復号	入力データをブロック暗号の CFB モードで暗号化/復号する．
疑似データ生成	暗号データと IV を合わせたハッシュを生成する．
チェックサム再計算	通常または疑似データを含めた独自の計算範囲でチェックサムの再計算を行う．
チェックサム検証	通常または疑似データを含めた独自の計算範囲でチェックサムの検証を行う．

ズムはAESのCFBモードを用いている[14]．鍵長は 256 ビットを採用した．ハッシュ関数は，AESで用いるIVが 128 ビットであることからMD5[15] を用いている．尚，暗号ライブラリとしてOpenSSLを採用した（Release version : openssl-0.9.7c）．

試作モジュールは表 1 のような仕様としてあるが，提案方式にこれらの規定はないので，暗号アルゴリズムはブロック暗号の CFB モードであれば他の共通鍵暗号アルゴリズムを用いてもよく，ハッシュ関数も同様に他のアルゴリズムを用いても構わない．但し暗号方式は，パケット長を変えないためにメッセージ置換型である必要がある．

試作モジュールは，IP 層の詳細な処理フローに関する情報が多い FreeBSD のカーネル内に組み込むことを想定して，FreeBSD で開発した（5.1 Release）．プログラムには C 言語を用いた．

4.1.2　モジュール構成

試作したモジュールの構成を**表 2** に示す．試作モジュールは，メインモジュールである暗号処理モジュールとそのサブモジュールである IV 生成モジュール，暗号化/復号モジュール，

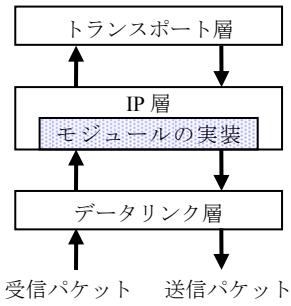


図 10　モジュールの実装位置
Fig. 10　Implementation position of modules.

疑似データ生成モジュール，チェックサム再計算モジュール，チェックサム検証モジュールから構成される．

暗号処理モジュールは，呼び出し時に与えられる，暗号化/復号などの動作内容を記した動作処理情報を基に処理を行う．動作内容が暗号化の場合は，送信側の処理として図 8 の手順で該当するサブモジュールを呼び出して処理を行う．動作内容が復号の場合は，受信側の処理として図 8 の手順で該当するサブモジュールを呼び出して処理を行う．尚，暗号鍵は予め用意されたものを用いる．実際の利用では，鍵配送方式や動作処理情報のネゴシエーションなどを考慮する必要がある．試作モジュールは，DPRP によって作成される動作処理情報テーブルを基に処理するようインタフェースを設計してある．

4.2　モジュールの実装方法

モジュールの実装方法として，FreeBSD のカーネル内にモジュールを組み込む方法を以下で述べる．

モジュールは，**図 10** のように BSD カーネル内における IP 層を改造して組み込む．送受信するパケットをデータリンク層に近い場所で横取りして処理を行い，終えたらパケットを差し戻す．図 10 が示すモジュールの実装位置には，提案方式をモジュール化した暗号処理モジュール以外に，パケットの横取り/差し戻し，テーブル検索，パケット判別，そしてこれらを取りまとめたメインシステムの機能が必要である．また，動作処理情報のネゴシエーション機能が別途必要であるが，IP 層に組み込むかソケットアプリケーションとして作成するかは任意であり，図 9 のような DPRP を用いる方法以外にも，目的に応じて様々な方法が考えられる．

動作処理情報の内容に規定はないが，宛先・送信元アドレス，宛先・送信元ポート番号，暗

表 3 既存技術との比較
Table 3 Comparison with existing technology.

	機密性	本人性確認	完全性保証	NA(P)T	ファイアウォール	フラグメント
IPsec ESP	◎	◎	◎	△	△	×
従来置換方式	○	×	×	×	○	○
提案方式	○	○	○	○	○	○

号化/復号などの動作内容，鍵の情報などである．テーブル検索は，横取りしたパケットの動作処理情報がテーブルに存在するかを検索し，ヒットしたらその情報を暗号処理モジュールに渡すものである．パケット判別は，暗号化を行ってはいけないパケットとしてRIP, OSPF, DHCPや一部のICMPパケットを暗号処理から除外するものである．

5. 評　　価

本章では既存技術との比較検討の結果をまとめる．また，ドライバによる試作モジュールの動作検証と処理時間の計測結果から，提案方式の有効性を評価する．

5.1 既存技術との比較検討

IPsec ESP と従来置換方式，提案方式を６つの項目において比較した結果を**表 3** に示す．

IPsec ESP は，TCP/UDP ヘッダを暗号化範囲に含めているが，特殊な処理・設定を行わないとNA(P)Tやファイアウォールを通過できない．また，ヘッダの追加によるオーバヘッドやフラグメントが発生する．

提案方式は，TCP/UDP ヘッダを暗号化範囲に含めない代わりに，NA(P)T やファイアウォールを通過できる．また，従来置換方式の課題である本人性確認とパケットの完全性保証も行える．パケット長が変化しないため，提案方式によるフラグメントは発生せず，高スループットを実現できる．

IPsec が強力な認証機能を提供するのに対し，提案方式は認証値としてTCP/UDPチェックサムを用いており，チェックサムのフィールド長は16ビットであるが，実用上は十分なものでありNA(P)TやファイアウォールのIP通過など実用性を重視している．

IPsec は，セキュリティは強靭だが NA(P)T やファイアウォールとの相性問題をはじめとした多くの課題がある．強力なセキュリティを要する場合は，これらの問題に注意しながら導入を検討することが重要である．提案方式は，既存システムに影響を与えないよう配慮しているため，実用性が高く比較的容易に導入できると考えられる．

5.2 試作モジュールの評価

試作したモジュールの動作検証と処理時間の計測を行うために，モジュールのドライバを作成した．ドライバは，IP パケット生成，動作処理情報の生成，パケットモニタ，処理時間の計測と表示，改竄テストの機能で構成される．

IP パケット生成は，IP ヘッダ，TCP ヘッダ，データから成る IP パケットを生成するもので，IP ヘッダと TCP ヘッダの大きさはオプションのない 20 バイト，データの大きさは目的に応じて設定する．

動作処理情報の生成では暗号処理モジュールに必要な情報として，動作内容と使用する鍵情報を設定する．ここでは使用する暗号鍵と動作内容の組み合わせとして，以下の 4 パターンを用意した．

- E-1：暗号鍵 1 で暗号化
- E-2：暗号鍵 2 で暗号化
- D-1：暗号鍵 1 で復号
- D-2：暗号鍵 2 で復号

パケットモニタは，IP ヘッダ，TCP ヘッダ，データをモニタするもので，動作検証を行うために試作モジュールの呼び出し前後でモニタする．パケット表示を有効にするかは設定で決める．

処理時間の計測と表示は，モジュールの呼び出し前後の時間の差分から処理時間を求めて結果を表示するもので，計測精度を高めるためにモジュールを 1 千万回繰り返して呼び出すのに掛かった時間を 1 千万で割った値を計測結果としてある．処理時間の計測と表示を有効にするかは設定で決める．動作検証を行うときは無効にする．

改竄テストは，数通り用意した改竄パターンからパケットの改竄を行うもので，改竄が正しく検出されるかを確認する．改竄テストを有効にするかは設定で決める．

動作検証は以下の流れで行う．まず，パケットモニタを有効にし，動作処理情報 E-1 と E-2 の選択を求め，その情報を基に送信側の処理を行う．次に，改竄テストが有効であればパケットを改竄し，動作処理情報 D-1 と D-2 の選択を求め，その情報を基に受信側の処理を行う．動作検証を行った結果，全てのパターンにおいて正しい動作結果が得られた．

400 バイトの IP パケットにおける試作モジュールの処理時間を計測した結果を**表 4** に示す．実験には 2.4GHz/Pentium4 を搭載した計算

表 4 試作モジュールの処理時間
Table 4 Processing time in trial production modules.

処理種別	処理時間 （μs）
送信側処理	16.7
受信側処理 （改竄なし）	15.4
受信側処理 （改竄あり）	3.3

表 5 サブモジュールの処理時間とその割合
Table 5 Processing time and the ratio in sub-modules.

	モジュール	処理時間 （μs）	割合 (%)
送信側	IV 生成	0.95	6
	チェックサム検証（通常）	0.77	5
	暗号化	11.85	75
	疑似データ生成	1.96	12
	チェックサム再計算（独自）	0.32	2
受信側	IV 生成	0.95	6
	疑似データ生成	1.96	12
	チェックサム検証（独自）	0.32	2
	復号	11.79	75
	チェックサム再計算（通常）	0.77	5

機を使用している（以下，同様）．改竄があった場合は，受信側で復号前にパケットを破棄するので大きな効果が得られている．処理種別の中で最も時間が掛かった送信側処理を試作モジュールの処理時間とすると，試作モジュールの処理速度は約 172Mbps である．これは，パケット長 400 バイトのうちのユーザデータ部分である 360 バイトを対象データ長として，それに掛かった処理時間から算出した．

$$360\text{byte}/16.7\,\mu\text{s} \fallingdotseq 172\text{Mbps}$$

この結果から，試作したモジュールは他処理にかかるオーバヘッドを考慮しても，100Mbps の通信環境で問題とならない性能であると言える．

試作モジュールを開発した当初は，IV の生成に HMAC を利用していたが，IV 自体を認証するわけではないため鍵情報もハッシュ関数の入力メッセージに含める方法をとり，処理パフォーマンスの向上を計らった．

サブモジュールの処理時間とその割合を解析した結果を**表 5** に示す．送信側/受信側ともに暗号化/復号が処理時間の大部分を占めている．ここでは，4.1.2 で述べたように OpenSSL ライブラリに搭載された AES の CFB モードを利用しいているが，暗号化/復号の処理時間は，利用するアルゴリズムや暗号エンジン・暗号ライブラリによって大きく異なるので，これらの工夫次第で処理速度の向上が見込める．チェックサムの検証/再計算は，通常の計算と独自の計算で処理時間が異なる．これは，通常の計算範囲が TCP/UDP 疑似ヘッダ，TCP/UDP ヘッダ，データであるのに対し，独自の計算範囲は TCP/UDP 疑似ヘッダ，TCP/UDP ヘッダ，疑似データと，疑似データを含む代わりにデータ部分を計算の範囲から外しているためである．疑似データの生成が 12%を占めているのは，ハッシュ関数の入力メッセージに暗号データが含まれているためである．ここでは，暗号データのハッシュ値を求め，その出力結果と IV を合わせたハッシュ値を疑似データとしている．これは，大きいサイズの暗号データと IV を合わせたハッシュの場合，結合に掛かる処理時間が長くなる恐れがあるためである．疑似データの生成は工夫次第で処理速度の向上が見込めそうなので，今後も検討していく．

6. むすび

実用性を重視した暗号通信方式として，既存システムに影響を与えず，またオリジナルパケットとサイズを変えないまま，本人性確認とパケットの完全性保証を行うことができる暗号通信方式を提案した．

提案方式の有効性を確認するために，ドライバを用いて試作モジュールの動作検証と性能評価を行った．試作モジュールの処理速度を計測した結果，100Mbps の通信環境で十分実用となる性能であることが確認できた．また，サブモジュールの処理速度を測定し，処理速度の向上に必要なものを検討した．

今後はモジュールをカーネルに組み込み，動作確認を行うと共に，既存技術との定量的な比較を行う予定である．また，正常なパケットを多量に送りつけるリプレイ攻撃の対策や，UDP パケットにおけるフラグメントの扱いについても検討する予定である．

参考文献

1) P. Rapalus, Computer Security Issues and Trends: CSI/FBI 2003 Computer Crime and Security Survey, CSI Press Release.
2) S. Kent, R. Atkinson.: Security Architecture for the Internet Protocol, RFC2401 (Aug. 1998).
3) R. Atkinson.: IP Authentication Header, RFC2402 (Dec. 1998)
4) R. Atkinson.: IP Encapsulation Security Payload (ESP), RFC2406 (Dec. 1998).
5) D. Harkins and D. Carrel.: The internet key exchange (IKE), RFC2409 (Dec. 1998).

6) 渡邊，厚井，井手口，横山，妹尾 “暗号技術を用いたセキュア通信グループの構築方式とその実現” 情報処理学会論文誌 Vol.38 No.04-025 (Apr. 1997).
7) R. Braden, D. Borman, C. Partridge.: Computiong the Internet Checksum, RFC1071 (Sep. 1988)
8) T. Mallory, A. Kullberg.: Incremental Updating of the Internet Checksum, RFC1141 (Jan. 1990)
9) A. Rijsinghani.: Computation of the Internet Checksum via Incremental Update, RFC1624 (May. 1994)
10) E. Rosen, Y. Rekhter.: BGP/MPLS VPNs, RFC2547(-Mar. 1999)
11) A. Huttunen, B. Swander, M. Stenberg, V. Volpe, L. Diburro.: UDP Encapsulation of IPsec Packets, Internet Draft,draft-ietf-ipsec-udp-encaps-07.txt (Oct. 2003)
12) K. Egevang, P. Francis.: The IP Network Address Translator (NAT), RFC1631 (May. 1994)
13) 渡邊，井手口，笹瀬“イントラネット閉域通信グループの物理的位置透過性を可能にする動的処理解決プロトコルの提案”電子情報通信学会論文誌 VOL.J84-D-I No.3 (Mar. 2001).
14) Daemen, J. and Rijmen, V.: AES Proposal: Rijndael. http://csrc.nist.bov/encryption/aes/rijndael/Rijndael.pdf
15) R. Rivest.: The MD5 Message-Digest Algorithm, RFC1321 (Apr. 1992)
16) B. Aboba, W. Dixon.: IPsec-NAT Compatibility Requirements, Internet Draft, draft-ietf-ipsec-nat-r-eqts-06.txt
17) T. Kivinen, A. Huttunen, B. Swander, V, Volpe.: Negotiation of NAT-Traversal in the IKE, Internet Draft, draft-ietf-ipsec-net-t-ike-08.txt (Feb. 2004)
18) UPnP FORUM, http://www.upnp.org
19) M. Oehler, R. Glenn.: HMAC-MD5 IP Authentication with Replay Prevention, RFC2085 (Feb. 1997)
20) FreeBSD:The Power To Serve, http://www.jp.freebsd.org/www.FreeBSD.org/ja/
21) The OpenSSL Project, http://www.openssl.org
22) Hiroki Yoshioka, “How to use cryptography library”, http://www.quintillion.co.jp/~yoshioka/crypto/openssl/des.html, 2000
23) “暗号利用技術ハンドブック 第 2 版”, 電子商取引実証推進協会 セキュリティ WG, 2000
24) W. Richard Stevens, “詳解 TCP/IP Vol.2 実装”, Pearson Education Japan, 2002

謝　　辞

本研究を進めるにあたり，多大なるご指導，ご鞭撻を賜りました渡邊晃教授に心から感謝いたします．とりわけ，ご多忙の中いつもお時間の許す限りご相談にのって頂きましたことに，深い感謝の念を表します．

また，本研究を進めていく上で様々なお励まし，ご助言，ご検討を頂きました渡邊研究室の市川氏，伊藤氏，加藤氏，鈴木氏，竹尾氏，竹内氏，前羽氏，保母氏，柳沢氏に深く感謝いたします．

更には，OpenSSL を利用するにあたりご質問させて頂いたメールに，迅速かつご丁寧にご返答頂きました吉岡氏と田中氏にお礼申し上げます．

最後に，研究を進めていくなか，いつも暖かく支えて頂いたご両親に心から感謝いたします．

付　録

A　ブロック暗号の CFB モードを採用した理由

CFB モードは任意長のデータを暗号化できるモードで, ストリーム暗号としての役割を果たす. ストリーム暗号には RC4 などが挙げられるが，パケットの完全性保証に IV を用いることや，実装上ブロック暗号の方が普及しており利用しやすいことから，ブロック暗号の CFB モードを採用した．尚，類似モードに OFB モードがあるが，セキュリティの面で CFB の方が強力と言われている．

B　リプレイ攻撃の問題と対策

セキュリティ対策として，パケットの盗聴には暗号化を，改竄には完全性保証を，なりすましには本人性確認を行うことで安全を確保できるが，暗号化を行う場合は復号処理負荷を狙った攻撃を考慮する必要がある．提案方式では復号前に不正パケットを破棄できるが，正常なパケットは当然ながら復号される．よって，クラッカーが正常なパケットを盗み取ってそのパケットのコピーを大量に送信するリプレイ攻撃を行った場合が問題となる．

IPsec では送信側で独自ヘッダにシーケンス番号を付与して送信し，受信側でリプレイ防御ウィンドウを使用して，受信されたパケットのシーケンス番号を確認し，重複があった場合や既に受信したパケットより大幅に小さい番号の場合はパケットを破棄する．リプレイ防御機能を有効にするかどうかは受信側で決定される．

提案方式の対策として，TCP のシーケンス番号を用いる方法が考えられるが，TCP シーケンス番号は IPsec のシーケンス番号のように 1 ずつ加算するのではなく, 送信したデータのオクテット数だけ値が加算されるため，IPsec のようなリプレイ防御ウィンドウでは対応できない．受信側でシーケンス番号を管理するテーブルを作成する方法も考えられるが，IPsec のようにすでに受信したパケットより大幅に小さい番号の場合を不正と見なすことはできない．

このように，復号処理負荷を狙ったリプレイ攻撃には何らかの対策をとる必要があるので，今後検討していく．

C　UDP におけるフラグメントの問題と対策

提案方式は，IP 層で分割されたパケットを扱うことができず，完全性を保証できない問題がある．UDP パケットの場合，分割時に IP チェックサムの再計算を行うが，**図 C-1** のように IP ペイロードには何もしない（フラグメントによって UDP ヘッダの情報が変わることはない）．よって，UDP チェックサムを再計算してしまうと，元のチェックサムには戻せなくなる．

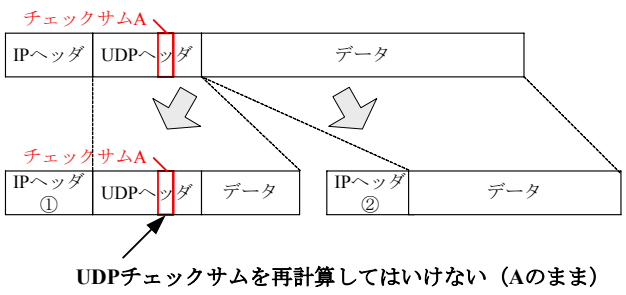


図 C-1　UDP におけるフラグメントの問題

UDP は信頼性を提供しないので，完全性も保証しないという方針も考えられるが，現時点では以下の対策が考えられる．

対策 1…フラグメントパケットは扱わない

提案方式は，必ず分割処理前に送信側処理を行い，必ずフラグメントパケットの再組み立て後に受信処理を行うことにする．この方式の場合はフラグメントを意識することはない．しかし，本方式を実装した装置が端末でなく，受信パケットを処理して転送する中継暗号装置の場合は，フラグメントパケットを一度再組み立てして処理を終えたら再び分割する必要があるので，スループットは大幅に低下すると考えられる．

対策 2…UDP については旧提案方式を用いる

旧提案方式とは，復号後にチェックサムの検証を行うことで完全性を保証する方式のことで，旧提案方式の処理内で TCP/UDP チェックサムの再計算は行わない．フラグメントパケットの再組み立て後に，上位層である UDP で OS が行うチェックサムの検証にて改竄検出ができる仕組みである．IV 生成に用いるフィールドは，UDP ヘッダを含むパケットと含まないパケットで異なるが，フラグメントパケットを識別して IV を生成し，暗号化/復号を行えばよい．しかしこの方式は，復号後でしか改竄は検出できない．

旧提案方式の原案となる資料
http://www-is.meijo-u.ac.jp/~watanabe/file/masuda/TokaiConvention_masuda.pdf

対策 3…TCP/UDP チェックサムの代わりに IP チェックサムを用いる

IP チェックサムは IP ヘッダをチェックするものであるが，IP チェックサムの計算を独自の処理に変えることで完全性保証を行う．考え方は**図 C-2** のようになる．

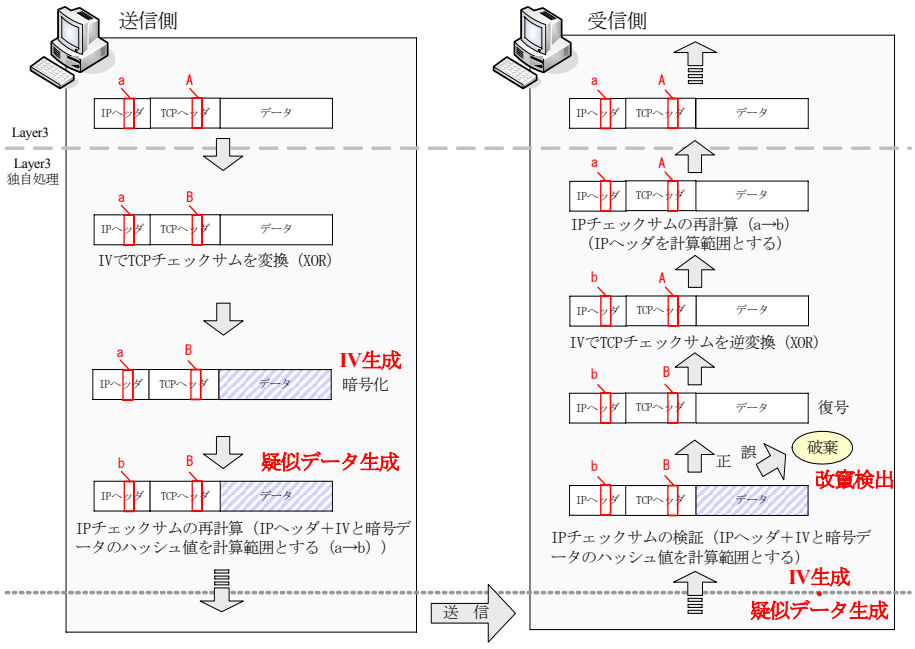


図 C-2　IP チェックサムを用いた改竄検出の流れ（TCP の場合．UDP も同様）

ルーティング中にルータや NAT によって IP ヘッダの情報が変化したら，IP チェックサムの書き換えを行うが，変換部分の差分計算であることが必須である．再計算をしてしまうと，検証で破棄されてしまう．また，この方式の場合はモジュールの呼び出し位置が変わる（受信時の処理で，ip_input の ”検証（IP チェックサムの検証など）” 前に呼び出す．ip_input 等の BSD カーネル内の処理に関しては「GSCIP 基本設計書」を参照）．また，IPv6 では IP チェックサムフィールドがないので，この方式は IPv4 でしか使えない．

D TCP/UDP チェックサムのフィールド長 16 ビットについて

IPsec で用いられる認証フィールドは任意長で，MD5-96 や SHA1-96 などのハッシュ値を 96 ビットに短縮したものがよく使われる．これは，ハッシュ値をそのままにするより，短縮した方がハッシュの中身の推測を不利にさせることができるためである．

提案方式では，TCP/UDP チェックサムが認証値となる．TCP/UDP チェックサムは 16 ビット長で，IPsec と比べると認証値としては短い．しかし，提案方式では TCP/UDP チェックサムそのものはハッシュ値ではなく，疑似データがハッシュ値であり，チェックサムの値は TCP/UDP 疑似ヘッダと TCP/UDP ヘッダ，そして疑似データを 16 ビットごとに 1 の補数で和をとった値の 1 の補数であるため，この値から中身を推測することはできない．

問題となるのは解読ではなく，チェックサムの重複である．16 ビット長なので 65,536 通りの値を表現できるが，チェックサムの値 65,536 通りのパケットを順に送りつけてゆけばやがて正しい値に辿り着く．問題は，このように大量にパケットを送りつけて，不正パケットを通過させる攻撃法が考えられるかである．ログを見る等で対処できそうだが，今後検討していく必要がある．

E TCP/UDP ヘッダが平文であることの考察

提案方式では，ユーザデータ部分のみを暗号化の対象としている．そこで，TCP/UDP ヘッダが平文であることの考察をした．

TCP ヘッダの場合，ポート番号とシーケンス番号がセキュリティ上重要な情報である．ポート番号が盗聴された場合，アプリケーションの推測の可能性があるが，実際はファイアウォールで使用できるポート番号は限定されており，それ自体がプライバシー等の問題となるとは考えにくい．また，シーケンス番号を見てコネクションを乗っ取る（TCP シーケンス番号の不整合を利用したハイジャック）行為があるが，提案方式では送信元の本人性確認を行うため，この手段は取れない．

UDP ヘッダの場合，ポート番号がセキュリティ上重要な情報であるが，TCP ポート番号と同様の事が言える．

F IPsec における NA(P)T，ファイアウォールの対策について

NA(P)T

NA(P)T を通過させる NAT-Traversal 技術として，“UDP Encapsulation of IPsec Packets”がインターネットドラフトとして公開されている．これは，ESP パケットを UDP パケットに見せかけて NA(P)T をだます手法で，NAT-Traversal を使う IPsec ゲートウェイは，まず NAT-Traversal が必要かどうかを判定する．そして，このことを反対側の IPsec ゲートウェイにも伝える．それから IP ヘッダと ESP ヘッダの間に UDP ヘッダを挿入し，IP ヘッダ中のプロトコル番号を示すフィールドを，UDP を示す 17 番に換える．この方式により NA(P)T を通過させる技術で，有効な手段として普及しつつある．しかし，相互で対応していることが前提である．また，UDP ポート番号は 500 番の場合が多く，ファイアウォールでポートを制限している環境では利用が困難である．尚，カプセル部分は完全性保証の範囲ではない．このように，一部の解決策にはなるが，特定された利用以外は困難である．

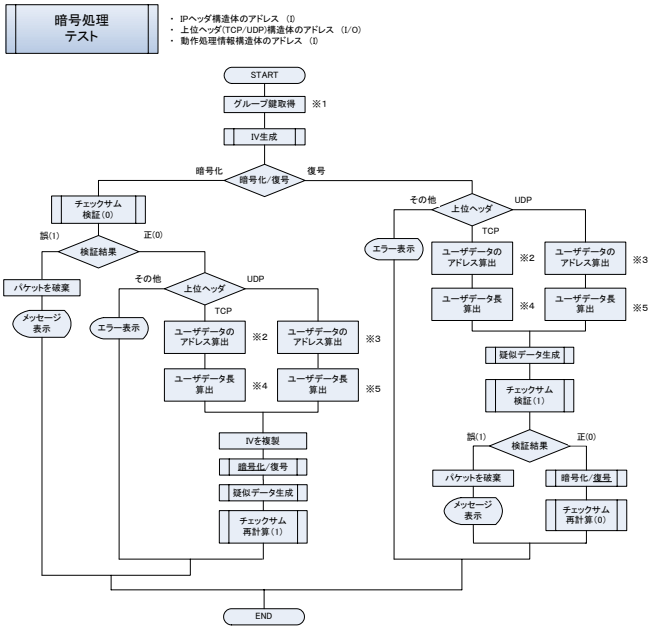
ファイアウォール

ファイアウォールと同一のゲートウェイに IPsec を実装すれば，用途に応じて経路を使い分けることになるのでルータなどによる経路設定も必要になるが，その他の問題は少なく一般的にこの配置を取る場合が多い．この配置以外は設定の複雑さなどから非常に困難である．このことから，ファイアウォールを経由する場合，End-to-End の IPsec 通信を行うことは非常に困難であることが分かる．

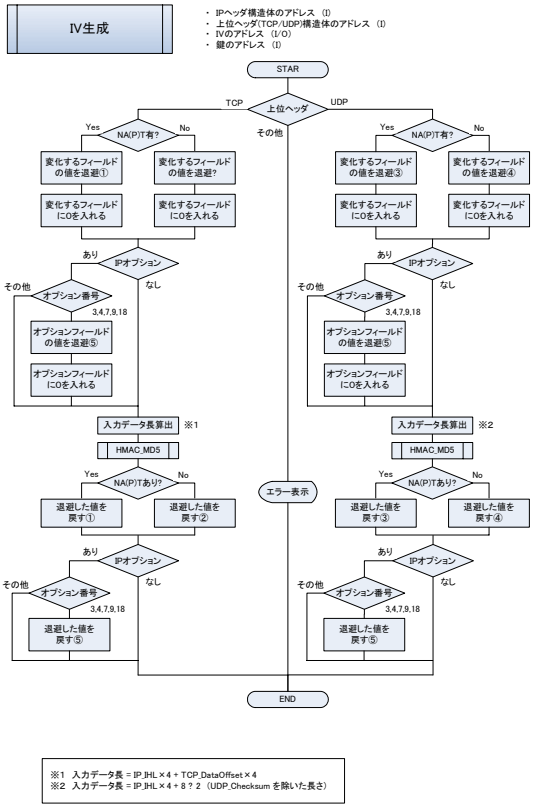
G　試作モジュール設計（フローチャート）

試作モジュールの設計（フローチャート）の一覧

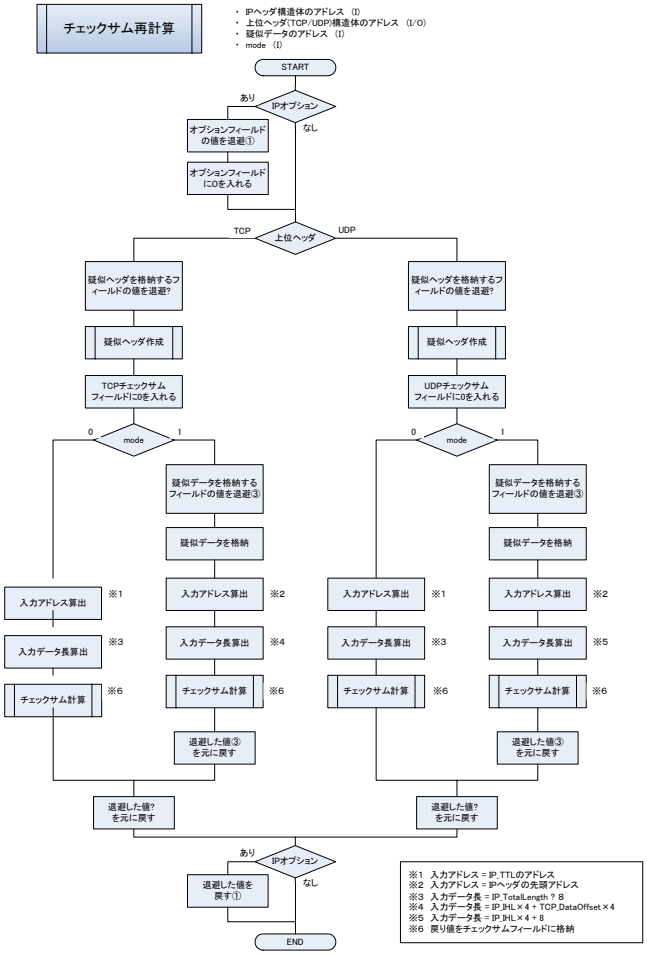
- 暗号処理テストモジュール
- IV 生成モジュール
- チェックサム再計算モジュール
- チェックサム検証モジュール
- 暗号化/復号モジュール
- 疑似データ生成モジュール
- テストドライバ

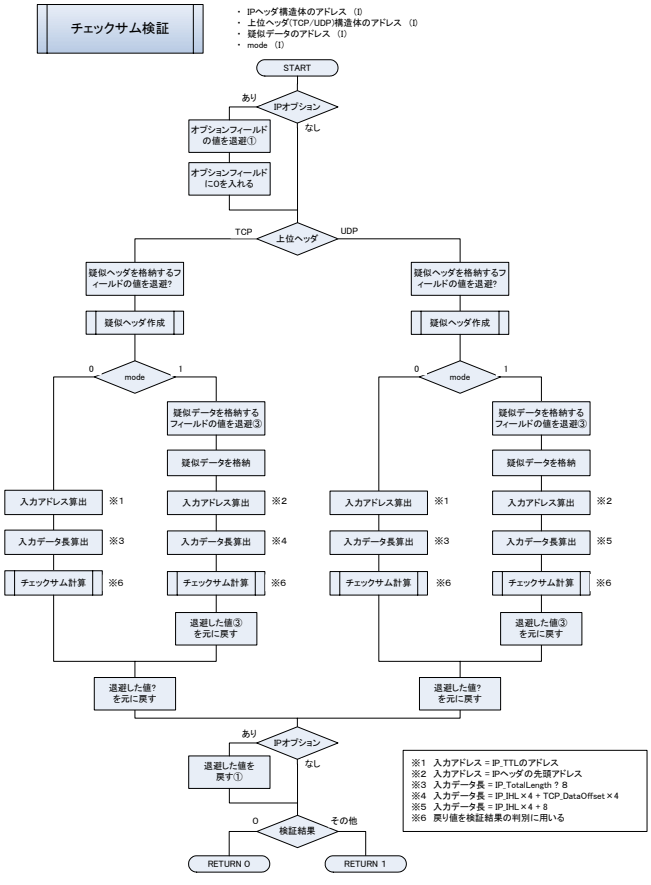


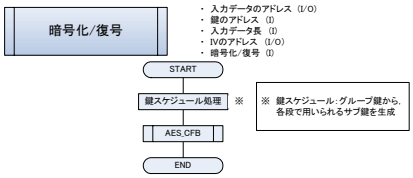
※1 グループ鍵を保持しているメモリオブジェクトから，該当グループ(動作処理情報から判断)の鍵を取り出す．
※2 ユーザデータのアドレス[番地] = A + IP_IHL × 4 + TCP_DataOffset × 4
※3 ユーザデータのアドレス[番地] = A + IP_IHL × 4 + 8
※4 暗号化/復号の入力データ長 = IP_TotalLength − IP_IHL × 4 − TCP_DataOffset × 4
※5 暗号化/復号の入力データ長 = UDP_Length ? 8
A = IPパケットの先頭アドレス

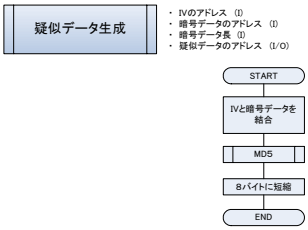


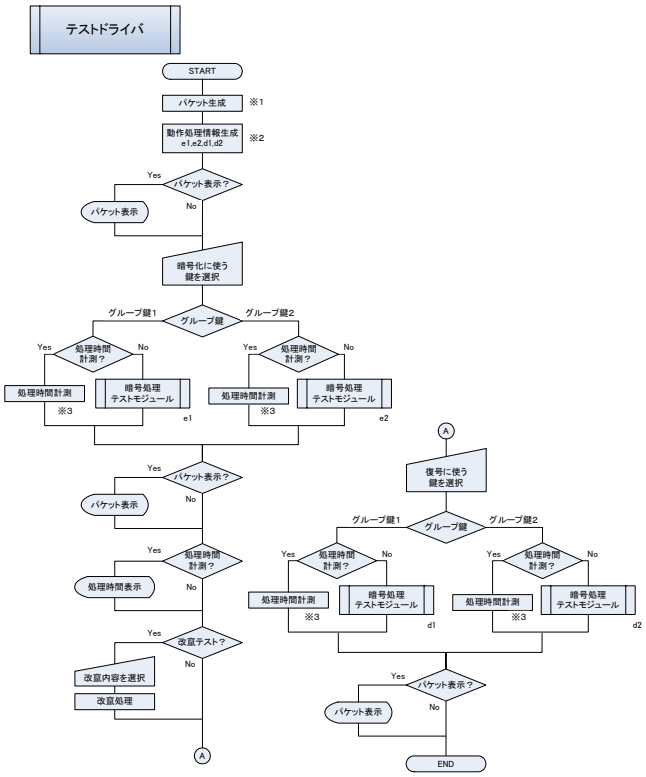
※1 入力データ長 = IP_IHL × 4 + TCP_DataOffset × 4
※2 入力データ長 = IP_IHL × 4 + 8 ? 2 （UDP_Checksum を除いた長さ）


チェックサム再計算
・ IPヘッダ構造体のアドレス （I）
・ 上位ヘッダ(TCP/UDP)構造体のアドレス （I/O）
・ 疑似データのアドレス （I）
・ mode （I）
START
IPオプション
あり
なし
オプションフィールドの値を退避①
オプションフィールドに0を入れる
上位ヘッダ
TCP
UDP
疑似ヘッダを格納するフィールドの値を退避?
疑似ヘッダ作成
TCPチェックサムフィールドに0を入れる
UDPチェックサムフィールドに0を入れる
mode
0
1
疑似データを格納するフィールドの値を退避③
疑似データを格納
入力アドレス算出
※1
※2
入力データ長算出
※3
※4
※5
チェックサム計算
※6
退避した値③を元に戻す
退避した値?を元に戻す
退避した値を戻す①
END
※1 入力アドレス = IP_TTLのアドレス
※2 入力アドレス = IPヘッダの先頭アドレス
※3 入力データ長 = IP_TotalLength ? 8
※4 入力データ長 = IP_IHL × 4 + TCP_DataOffset × 4
※5 入力データ長 = IP_IHL × 4 + 8
※6 戻り値をチェックサムフィールドに格納


チェックサム検証
・ IPヘッダ構造体のアドレス （I）
・ 上位ヘッダ(TCP/UDP)構造体のアドレス （I）
・ 疑似データのアドレス （I）
・ mode （I）
START
IPオプション
あり
なし
オプションフィールドの値を退避①
オプションフィールドに0を入れる
上位ヘッダ
TCP
UDP
疑似ヘッダを格納するフィールドの値を退避?
疑似ヘッダ作成
mode
0
1
疑似データを格納するフィールドの値を退避③
疑似データを格納
入力アドレス算出
※1
※2
入力データ長算出
※3
※4
※5
チェックサム計算
※6
退避した値③を元に戻す
退避した値?を元に戻す
退避した値を戻す①
検証結果
その他
RETURN 0
RETURN 1
※1 入力アドレス = IP_TTLのアドレス
※2 入力アドレス = IPヘッダの先頭アドレス
※3 入力データ長 = IP_TotalLength ? 8
※4 入力データ長 = IP_IHL × 4 + TCP_DataOffset × 4
※5 入力データ長 = IP_IHL × 4 + 8
※6 戻り値を検証結果の判別に用いる







※1 IPヘッダ + TCPヘッダ + データ
※2 e1…動作処理内容:暗号化, グループ番号:1
e2…動作処理内容:暗号化, グループ番号:2
d1…動作処理内容:復号, グループ番号:1
d2…動作処理内容:復号, グループ番号:2
※3 暗号処理テストモジュールを1千万回繰り返すのにかかる時間を計測して, その時間を1千万で割った値を処理時間とする

H　試作モジュール ソースコード

試作したモジュールのソースコード一覧

- CipherProcess.c
- CipherProcess.h
- param.h
- driver.c

OpenSSL （Release version : openssl-0.9.7a 以上）暗号ライブラリをインストール済みの FreeBSD で動作確認済み.

CipherProcess.c と driver.c の動的リンクライブラリを作成して，それからバイナリファイルを作成．このとき，OpenSSL ライブラリの指定が必要.

例）
```
$gcc –c CipherProcess.c
$gcc –c driver.c
$gcc –o CPT CipherProcess.o driver.o /usr/local/sssl/lib/libcrypto.a
$./CPT
```

CipherProcess.c

```
/* CipherProcess.c */
/*
* Copyright (C) 2003-2004
* Shinya Masuda
* Watanabe Laboratory,
* Meijo University in Japan.
* All rights reserved.
*
* This program is written by Shinya Masuda (a300j125@ccmailg.meijo-u.ac.jp).
* This program is free to use.
*
* This program includes software developed by the OpenSSL Project
* for use in the OpenSSL Toolkit (http://www.openssl.org/).
* This program includes cryptographic software written by Eric Young
* (eay@cryptsoft.com).
*
* ---- Explanation of this program ----
* Range of message authentication is part of encryption and field to
* use for a making of IV (Initialization Vector).
*
* Part of encryption and field to use for a making of IV is different
* by the existance of the NA(P)T in a network environmento to presume.
*
* This program is able to setup of the existance of the NA(P)T by the
* value of "#define NAPT".
*
* NA(P)T exist:
*   Range of encryption: TCP payload
*   Field to use for a making of IV:
*       IP header: Version,IHL,Total Length,ID,Protocol,(Option)
*       TCP header: Sequence Number,Acknowledgement Number,
*                   Data Offset,Reserved,Control Flag,Window,
*                   Urgent Pointer,(Option)
* NA(P)T is Nothing:
*   Range of encryption: TCP payload
*   Field to use for a making of IV:
*       IP header: Version,IHL,Total Length,ID,Protocol,
*                  Source Address,Destination Address,(Option)
*       TCP header : Source Port,Destination Port,Sequence Number,
*                    Ackowledgement Number,Data Offset,Reserved,
*                    Control Flag,Window,Urgent Pointer,(Option)
*
* This program take no thought to mbuf and flagment.
*
*/
#include <stdio.h>
#include <string.h>
#include "CipherProcess.h"

#include <openssl/aes.h>
#include <openssl/md5.h>

/* group key */
u_char gkey1[GKEY_SIZE]={0x31,0xC1,0xD5,0xD0,0x39,0x77,0x67,0x42,
0x11,0xB1,0xC3,0xFD,0x79,0x75,0x67,0x32,
0x02,0xA3,0xF6,0xD3,0x29,0x87,0x47,0x48,
0x31,0xD1,0xC6,0xDF,0x89,0x78,0x97,0x39};
u_char gkey2[GKEY_SIZE]={0x11,0xD3,0xDA,0xA2,0x46,0x67,0x21,0x25,
0x15,0xB2,0xD3,0xFF,0x71,0x7A,0x6A,0x33,
0x01,0xAA,0x66,0xD3,0x22,0x8A,0x42,0x42,
0x33,0xDD,0x36,0xD8,0x83,0x74,0x94,0x37};

/*********************************************************
void make_iv(struct ip *ip, void *nexthead, u_char *iv_addr)

function:
make IV module
parametor:
ip      : struct of IP header address (I)
nexthead    : struct of Next header(tcp/udp) address (I)
key_addr    : key address (I)
iv_addr     : Initialization Vector address (I/O)
return:
---
*********************************************************/
void make_iv(struct ip *ip, void *nexthead, u_char *key_addr, u_char
*iv_addr)
{
/* loop variable */
int i;
/* struct of TCP header address */
struct tcphdr *tcp;
/* struct of UDP header address */
struct udphdr *udp;
/* struct of IP_Option field address */
struct ipopt *ipopt;
/* input data length */
int length;
/* buffer */
u_char buf[48];
/* MD5 context */
MD5_CTX c;
/* reservation variable for IP hader field which change during transmission
*/
u_char  ip_tos;     /* type of service */
u_short ip_off;     /* fragment offset field */
u_char  ip_ttl;     /* time to live */
u_short ip_sum;     /* checksum */
u_long  ip_src;     /* source address */
u_long  ip_dst;     /* destination address */
/* reservation variable for TCP hader field which change during transmission
*/
u_short th_sport;   /* source port */
u_short th_dport;   /* destination port */
u_short th_sum;     /* checksum */
/* reservation variable for UDP hader field which change during transmission
*/
u_short uh_sport;   /* source port */
u_short uh_dport;   /* destination port */
/* reservation variable for IP_Optionf field */
/*????????????????????????????????????????????????????????????????????????????????
*/
////u_char optbuf[1460];
/*????????????????????????????????????????????????????????????????????????????????
*/

/* If next header is TCP then */
if(ip->ip_p == 0x06)
{
/* input the Nextheader address */
tcp = (struct tcphdr * )nexthead;

/* reserve IP header field which change during transmission (1,2) */
ip_tos = ip->ip_tos;
ip_off = ip->ip_off;
ip_ttl = ip->ip_ttl;
ip_sum = ip->ip_sum;
/* If NA(P)T exist then */
if(NAPT == 1)
{
    ip_src = ip->ip_src.s_addr;
    ip_dst = ip->ip_dst.s_addr;
    /* reserve TCP header field which change during transmission */
    th_sport = tcp->th_sport;
    th_dport = tcp->th_dport;
}
th_sum = tcp->th_sum;

/* input 0 to IP header field which change during transmission */
ip->ip_tos = 0x00;
ip->ip_off = 0x0000;
ip->ip_ttl = 0x00;
ip->ip_sum = 0x0000;
/* If NA(P)T exist then */
if(NAPT == 1)
{
    ip->ip_src.s_addr = 0x00000000;
    ip->ip_dst.s_addr = 0x00000000;
    /* input 0 to TCP header field which change during transmission */
    tcp->th_sport = 0x0000;
    tcp->th_dport = 0x0000;
}
tcp->th_sum = 0x0000;

/* If IP_Option is contained then */
if((ip->ip_hl << 2) > 20)
{
    /* set address for IP_Option 1 byte */
    ipopt = (struct ipopt *)ip + 20;
    /* If IP_Option number is 3,4,7,9,18 then */
    if( ipopt->opt_optnum == 3 ||
        ipopt->opt_optnum == 4 ||
        ipopt->opt_optnum == 7 ||
        ipopt->opt_optnum == 9 ||
        ipopt->opt_optnum == 18 )
    {
        /* reserve IP_Option field (5) */
        /*** Omission ***/
        /* input 0 to IP_Option field */
        /*** Omission ***/
    }
}
```

```
/* calculate length of input data */
length = (ip->ip_hl << 2) + (tcp->th_off << 2);

/* MD5 */
/* MD5_Init(Context) */
MD5_Init(&c);
/* MD5_Update(Context, Buffer, BufferSize) */
MD5_Update(&c, ip, length);
/* MD5_Final(Digest, Context) */
MD5_Final(buf, &c);

/* combine */
for(i=0; i<32; i++)
    buf[i + 16] = *(key_addr + i);

/* MD5 */
/* MD5_Init(Context) */
MD5_Init(&c);
/* MD5_Update(Context, Buffer, BufferSize) */
MD5_Update(&c, buf, 48);
/* MD5_Final(Digest, Context) */
MD5_Final(iv_addr, &c);

/* return the reserved variable (1,2) */
/* IP header */
ip->ip_tos = ip_tos;
ip->ip_off = ip_off;
ip->ip_ttl = ip_ttl;
ip->ip_sum = ip_sum;
/* If NA(P)T exist then */
if(NAPT == 1)
{
    ip->ip_src.s_addr = ip_src;
    ip->ip_dst.s_addr = ip_dst;
    /* TCP header */
    tcp->th_sport = th_sport;
    tcp->th_dport = th_dport;
}
tcp->th_sum = th_sum;

/* If IP_Option is contained then */
if((ip->ip_hl << 2) > 20)
{
    /* set address for IP_Option 1 byte */
    ipopt = (struct ipopt *)ip + 20;
    /* If IP_Option number is 3,4,7,9,18 then */
    if( ipopt->opt_optnum == 3 ||
        ipopt->opt_optnum == 4 ||
        ipopt->opt_optnum == 7 ||
        ipopt->opt_optnum == 9 ||
        ipopt->opt_optnum == 18 )
    {
        /* return the reserved variable (5) */
        /*** Omission ***/
    }
}
}
/* If next header is UDP then */
else if(ip->ip_p == 0x17)
{
/* input the Nextheader address */
udp = (struct udphdr * )nexthead;

/* reserve IP header field which change during transmission (3,4) */
ip_tos = ip->ip_tos;
ip_off = ip->ip_off;
ip_ttl = ip->ip_ttl;
ip_sum = ip->ip_sum;
/* If NA(P)T exist then */
if(NAPT == 1)
{
    ip_src = ip->ip_src.s_addr;
    ip_dst = ip->ip_dst.s_addr;
    /* reserve UDP header field which change during transmission */
    uh_sport = udp->uh_sport;
    uh_dport = udp->uh_dport;
}

/* input to IP header field which change during transmission */
ip->ip_tos = 0x00;
ip->ip_off = 0x0000;
ip->ip_ttl = 0x00;
ip->ip_sum = 0x0000;
/* If NA(P)T exist then */
if(NAPT == 1)
{
    ip->ip_src.s_addr = 0x00000000;
    ip->ip_dst.s_addr = 0x00000000;
    /* input to UDP header field which change during transmission */
    udp->uh_sport = 0x0000;
    udp->uh_dport = 0x0000;
}

/* If IP_Option is contained then */
if((ip->ip_hl << 2) > 20)
{
    /* set address for IP_Option 1 byte */
    ipopt = (struct ipopt *)ip + 20;

    /* If IP_Option number is 3,4,7,9,18 then */
    if( ipopt->opt_optnum == 3 ||
        ipopt->opt_optnum == 4 ||
        ipopt->opt_optnum == 7 ||
        ipopt->opt_optnum == 9 ||
        ipopt->opt_optnum == 18 )
    {
        /* reserve IP_Option field (5) */
        /*** Omission ***/
        /* input 0 to IP_Option field */
        /*** Omission ***/
    }
}

/* calculate length of input data */
length = (ip->ip_hl << 2) + 8 - 2;

/* MD5 */
/* MD5_Init(Context) */
MD5_Init(&c);
/* MD5_Update(Context, Buffer, BufferSize) */
MD5_Update(&c, ip, length);
/* MD5_Final(Digest, Context) */
MD5_Final(buf, &c);

/* combine */
for(i=0; i<32; i++)
    buf[i + 16] = *(key_addr + i);

/* MD5 */
/* MD5_Init(Context) */
MD5_Init(&c);
/* MD5_Update(Context, Buffer, BufferSize) */
MD5_Update(&c, buf, 48);
/* MD5_Final(Digest, Context) */
MD5_Final(iv_addr, &c);

/* return the reserved variable (3,4) */
/* IP header */
ip->ip_tos = ip_tos;
ip->ip_off = ip_off;
ip->ip_ttl = ip_ttl;
ip->ip_sum = ip_sum;
/* If NA(P)T exist then */
if(NAPT == 1)
{
    ip->ip_src.s_addr = ip_src;
    ip->ip_dst.s_addr = ip_dst;
    /* UDP header */
    udp->uh_sport = uh_sport;
    udp->uh_dport = uh_dport;
}

/* If IP_Option is contained then */
if((ip->ip_hl << 2) > 20)
{
    /* set address for IP_Option 1 byte */
    ipopt = (struct ipopt *)ip + 20;
    /* If IP_Option number is 3,4,7,9,18 then */
    if( ipopt->opt_optnum == 3 ||
        ipopt->opt_optnum == 4 ||
        ipopt->opt_optnum == 7 ||
        ipopt->opt_optnum == 9 ||
        ipopt->opt_optnum == 18 )
    {
        /* return the reserved variable (5) */
        /*** Omission ***/
    }
}
}
/* Exception */
else printf("Exceptional error. This packet is not TCP or UDP.¥nError message
from make_iv().¥n");

}


/*********************************************
void crypto(u_char *idat_addr, int length, u_char *key_addr, u_char *iv_addr,
```

```
int EncDec)

function:
Encryption/Desryption module (AES_CFB)
parametor:
idat_addr   : input data address (I/O)
length      : length of input data (I)
key_addr    : key address (I)
iv_addr     : Initialization Vector address (I/O)
EncDec      : AES_ENCRYPT(1):ecnryption
AES_DECRYPT(0):decryption  (I)
return:
---
*********************************************/
void crypto(u_char *idat_addr, int length, u_char *key_addr, u_char *iv_addr,
int EncDec)
{
int num = 0;
/* key schedule */
AES_KEY ks;

/* set key */
AES_set_encrypt_key(key_addr, GKEY_SIZE * 8,  &ks);
/* encryption or decryption */
AES_cfb128_encrypt(idat_addr, idat_addr, (u_long)length, &ks, iv_addr,
&num, EncDec);
}


/*********************************************************
void make_simdat(u_char *iv_addr, u_char *encdat_addr, int length, u_char
*simdat_addr)

function:
make simulated data module
parametor:
iv_addr     : Initialization Vector address (I)
encdat_addr : encrypted data address (I)
length      : length of encrypted data (I)
simdat_addr : simulated data address (I/O)

return:
---
*********************************************************/
void make_simdat(u_char *iv_addr, u_char *encdat_addr, int length, u_char
*simdat_addr)
{
/* loop variable */
int i;
/* buffer */
u_char buf1[32], buf2[16];
/* MD5 context */
MD5_CTX c;


/* initialize */
memset(buf1, 0, 32);
/* MD5 */
/* MD5_Init(Context) */
MD5_Init(&c);
/* MD5_Update(Context, Buffer, BufferSize) */
MD5_Update(&c, encdat_addr, length);
/* MD5_Final(Digest, Context) */
MD5_Final(buf1, &c);

/* combine */
for(i=0; i<16; i++)
    buf1[i + 16] = *(iv_addr + i);

/* MD5 */
/* MD5_Init(Context) */
MD5_Init(&c);
/* MD5_Update(Context, Buffer, BufferSize) */
MD5_Update(&c, buf1, 32);
/* MD5_Final(Digest, Context) */
MD5_Final(buf2, &c);

memcpy(simdat_addr, buf2, 8);
}


/************************************************
u_short checksum(u_short *idat_addr, int length);

function:
calculate checksum sub module
parametor:
idat_addr   : input data address (I)
length      : length of input data (I)

return:
calculated value (u_short)
************************************************/
u_short checksum(u_short *idat_addr, int length)
{
long sum = 0;   /* checksum */

for(; length>1; length-=2)
{
    sum += *idat_addr++;
    if(sum & 0x80000000)
        sum = (sum & 0xffff) + (sum >> 16);
}

if(length == 1)
{
    u_short i = 0;
    *(u_char*)(&i) = *(u_char *)idat_addr;
    sum += i;
}

while(sum >> 16)
    sum = (sum & 0xffff) + (sum >> 16);

return ~sum;
}


/*********************************************************
void recalc_cksum(struct ip *ip, void *nexthead, u_char *simdat_addr, int
mode)

function:
recalculate checksum module
parametor:
ip      : struct of IP header address (I)
nexthead    : struct of Next header(tcp/udp) address (I/O)
simdat_addr : simulated data address (I)
mode        : (0):nomal
        (1):contain simulated data (I)
        return:
        ---
*********************************************************/
void recalc_cksum(struct ip *ip, void *nexthead, u_char *simdat_addr, int
mode)
{
/* struct of TCP header address */
struct tcphdr *tcp;
/* struct of UDP header address */
struct udphdr *udp;
/* reservation variable for IP hader field */
u_char iph_buf[8];  /* top 8 byte buffer of IP header*/
u_char  ip_ttl;     /* time to live */
u_short ip_sum;     /* checksum */
/* input data address */
u_short *idat_addr;
/* input data length */
int length;


/* If IP_Option is contained then */
if((ip->ip_hl << 2) > 20)
{
    /* reserve IP_Option field (1) */
    /*** Omission ***/
    /* input 0 to IP_Option field */
    /*** Omission ***/
}

/* If next header is TCP then */
if(ip->ip_p == 0x06)
{
    /* input the Nextheader address */
    tcp = (struct tcphdr * )nexthead;

    /* reserve IP header field which input simulated header (2) */
    ip_ttl = ip->ip_ttl;
    ip_sum = ip->ip_sum;

    /* make simulated header */
    ip->ip_ttl = 0;
    ip->ip_sum = (ip->ip_len >> 2) - ip->ip_hl;

    /* input 0 to TCP checksum field */
    tcp->th_sum = 0x0000;

    /* If mode is 0 then */
    if(mode == 0)
    {
```

```
        /* calculate address of input data */
        idat_addr = (u_short *)ip + 8;

        /* calculate length of input data */
        length = ip->ip_len - 8;

        /* calculate TCP checksum */
        tcp->th_sum = checksum(idat_addr, length);
    }
    else if(mode == 1)
    {
        /* reserve IP header field which input simulated data (3) */
        memcpy(iph_buf, (u_char *)ip, 8);

        /* input simulated data to IP header field */
        memcpy((u_char *)ip, simdat_addr, 8);

        /* calculate address of input data */
        idat_addr = (u_short *)ip;

        /* calculate length of input data */
        length = (ip->ip_hl << 2) + (tcp->th_off << 2);

        /* calculate TCP checksum */
        tcp->th_sum = checksum(idat_addr, length);

        /* return the reserved variable (3) */
        memcpy((u_char *)ip, iph_buf, 8);
    }
    /* Exception */
    else printf("Exceptional error. This mode value isn't 0 or 1.¥nError
message from recalc_cksum().¥n");

    /* return the reserved variable (2) */
    ip->ip_ttl = ip_ttl;
    ip->ip_sum = ip_sum;

}
/* If next header is UDP then */
else if(ip->ip_p == 0x17)
{
    /* input the Nextheader address */
    udp = (struct udphdr * )nexthead;

    /* reserve IP header field which input simulated header (2) */
    ip_ttl = ip->ip_ttl;
    ip_sum = ip->ip_sum;

    /* make simulated header */
    ip->ip_ttl = 0;
    ip->ip_sum = (ip->ip_len >> 2) - ip->ip_hl;

    /* input 0 to UDP checksum field */
    udp->uh_sum = 0x0000;

    /* If mode is 0 then */
    if(mode == 0)
    {
        /* calculate address of input data */
        idat_addr = (u_short *)ip + 8;

        /* calculate length of input data */
        length = ip->ip_len - 8;

        /* calculate UDP checksum */
        udp->uh_sum = checksum(idat_addr, length);
    }
    else if(mode == 1)
    {
        /* reserve IP header field which input simulated data (3) */
        memcpy(iph_buf, (u_char *)ip, 8);

        /* input simulated data to IP header field */
        memcpy((u_char *)ip, simdat_addr, 8);

        /* calculate address of input data */
        idat_addr = (u_short *)ip;

        /* calculate length of input data */
        length = (ip->ip_hl << 2) + 8;

        /* calculate UDP checksum */
        udp->uh_sum = checksum(idat_addr, length);

        /* return the reserved variable (3) */
        memcpy((u_char *)ip, iph_buf, 8);
    }
    /* Exception */
    else printf("Exceptional error. This mode value isn't 0 or 1.¥nError
message from recalc_cksum().¥n");

    /* return the reserved variable (2) */
    ip->ip_ttl = ip_ttl;
    ip->ip_sum = ip_sum;

}

/* If IP_Option is contained then */
if((ip->ip_hl << 2) > 20)
{
    /* return the reserved variable (1) */
    /*** Omission ***/
}
}


/*********************************************************
int verify_cksum(struct ip *ip, void *nexthead, u_char *simdat_addr, int
mode)

function:
verify checksum module
parametor:
ip       : struct of IP header address (I)
nexthead    : struct of Next header(tcp/udp) address (I)
simdat_addr : simulated data address (I)
mode        : (0):nomal
(1):contain simulated data (I)
return:
true(0) or false(1) (int)
*********************************************************/
int verify_cksum(struct ip *ip, void *nexthead, u_char *simdat_addr, int
mode)
{
/* verification result */
u_short result;
/* struct of TCP header address */
struct tcphdr *tcp;
/* struct of UDP header address */
struct udphdr *udp;
/* reservation variable for IP hader field */
u_char iph_buf[8];  /* top 8 byte buffer of IP header*/
u_char  ip_ttl;     /* time to live */
u_short ip_sum;     /* checksum */
/* input data address */
u_short *idat_addr;
/* input data length */
int length;


/* If IP_Option is contained then */
if((ip->ip_hl << 2) > 20)
{
    /* reserve IP_Option field (1) */
    /*** Omission ***/
    /* input 0 to IP_Option field */
    /*** Omission ***/
}

/* If next header is TCP then */
if(ip->ip_p == 0x06)
{
    /* input the Nextheader address */
    tcp = (struct tcphdr * )nexthead;

    /* reserve IP header field which input simulated header (2) */
    ip_ttl = ip->ip_ttl;
    ip_sum = ip->ip_sum;

    /* make simulated header */
    ip->ip_ttl = 0;
    ip->ip_sum = (ip->ip_len >> 2) - ip->ip_hl;

    /* If mode is 0 then */
    if(mode == 0)
    {
        /* calculate address of input data */
        idat_addr = (u_short *)ip + 8;

        /* calculate length of input data */
        length = ip->ip_len - 8;

        /* calculate TCP checksum */
        result = checksum(idat_addr, length);
    }
    else if(mode == 1)
    {
        /* reserve IP header field which input simulated data (3) */
```

```
        memcpy(iph_buf, (u_char *)ip, 8);

        /* input simulated data to IP header field */
        memcpy((u_char *)ip, simdat_addr, 8);

        /* calculate address of input data */
        idat_addr = (u_short *)ip;

        /* calculate length of input data */
        length = (ip->ip_hl << 2) + (tcp->th_off << 2);

        /* calculate TCP checksum */
        result = checksum(idat_addr, length);

        /* return the reserved variable (3) */
        memcpy((u_char *)ip, iph_buf, 8);
    }
    /* Exception */
    else printf("Exceptional error. This mode value isn't 0 or 1.¥nError
message from recalc_cksum().¥n");

    /* return the reserved variable (2) */
    ip->ip_ttl = ip_ttl;
    ip->ip_sum = ip_sum;

}
/* If next header is UDP then */
else if(ip->ip_p == 0x17)
{
    /* input the Nextheader address */
    udp = (struct udphdr * )nexthead;

    /* reserve IP header field which input simulated header (2) */
    ip_ttl = ip->ip_ttl;
    ip_sum = ip->ip_sum;

    /* make simulated header */
    ip->ip_ttl = 0;
    ip->ip_sum = (ip->ip_len >> 2) - ip->ip_hl;

    /* If mode is 0 then */
    if(mode == 0)
    {
        /* calculate address of input data */
        idat_addr = (u_short *)ip + 8;

        /* calculate length of input data */
        length = ip->ip_len - 8;

        /* calculate UDP checksum */
        result =  checksum(idat_addr, length);
    }
    else if(mode == 1)
    {
        /* reserve IP header field which input simulated data (3) */
        memcpy(iph_buf, (u_char *)ip, 8);

        /* input simulated data to IP header field */
        memcpy((u_char *)ip, simdat_addr, 8);

        /* calculate address of input data */
        idat_addr = (u_short *)ip;

        /* calculate length of input data */
        length = (ip->ip_hl << 2) + 8;

        /* calculate UDP checksum */
        result = checksum(idat_addr, length);

        /* return the reserved variable (3) */
        memcpy((u_char *)ip, iph_buf, 8);
    }
    /* Exception */
    else printf("Exceptional error. This mode value isn't 0 or 1.¥nError
message from recalc_cksum().¥n");

    /* return the reserved variable (2) */
    ip->ip_ttl = ip_ttl;
    ip->ip_sum = ip_sum;

}

/* If IP_Option is contained then */
if((ip->ip_hl << 2) > 20)
{
    /* return the reserved variable (1) */
    /*** Omission ***/
}

/* If verification result is 0 then */
if(result == 0)
    return 0;
else
    return 1;
}


/*********************************************************
void cip_proc_t(struct ip *ip, void *nexthead, PID *PID_addr)

function:
cipher process test module
parametor:
ip      : struct of IP header address (I)
nexthead    : struct of Next header(tcp/udp) address (I)
PID_addr    : struct of process infomation data (I)
return:
---
*********************************************************/
void cip_proc_t(struct ip *ip, void *nexthead, PID *PID_addr)
{
/* struct of TCP header address */
struct tcphdr *tcp;
/* struct of UDP header address */
struct udphdr *udp;
/* user data address */
u_char *udat_addr;
/* user data length */
int length;
/* gkey address*/
u_char *key_addr;
/* Initialization Vector */
u_char iv[IV_SIZE];
/* reservation variable for IV */
u_char res_iv[IV_SIZE];
/* simulated data */
u_char simdat[8];
/* verification result */
int result;


/*******************************************
tcp = (struct tcphdr * )nexthead;
printf("tcp_sum :%x¥n", tcp->th_sum);
recalc_cksum(ip, nexthead, simdat, 0);
printf("tcp_sum :%x¥n", tcp->th_sum);
*******************************************/

/****** get group key (test) ******/
if(PID_addr->g_no == 1) key_addr = gkey1;
else if(PID_addr->g_no == 2) key_addr = gkey2;

/* make IV */
make_iv(ip, nexthead, key_addr, iv);

/* If process infomation contents c is Enc then */
if(PID_addr->proc_c == 1)
{
    /* verify checksum (0) */
    result = verify_cksum(ip, nexthead, simdat, 0);

    /* If verification result is true then */
    if(result == 0)
    {
        /* If next header is TCP then */
        if(ip->ip_p == 0x06)
        {
            /* input the Nextheader address */
            tcp = (struct tcphdr * )nexthead;

            /* calculate address of user data */
            udat_addr = (u_char *)ip + (ip->ip_hl << 2) + (tcp->th_off <<
2);

            /* calculate length of user data */
            length = ip->ip_len - (ip->ip_hl << 2) - (tcp->th_off << 2);
        }
        /* If next header is UDP then */
        else if(ip->ip_p == 0x17)
        {
            /* input the Nextheader address */
            udp = (struct udphdr * )nexthead;

            /* calculate address of user data */
            udat_addr = (u_char *)ip + (ip->ip_hl << 2) + 8;
            /* calculate length of user data */
            length = udp->uh_ulen - 8;
        }
```

```
        /* If next header is TCP or UDP then */
        if(ip->ip_p == 0x06 || ip->ip_p == 0x17)
        {
            /* copy the IV */
            memcpy(res_iv, iv, IV_SIZE);

            /* encrypto */
            crypto(udat_addr, length, key_addr, res_iv, 1);

            /* make simulated data */
            make_simdat(iv, udat_addr, length, simdat);

            /* recalculate checksum (1) */
            recalc_cksum(ip, nexthead, simdat, 1);
        }
        /* Exception */
        else printf("Exceptional error. This packet is not TCP or
UDP.¥nError message from cip_proc_t().¥n");
    }
    /* If verification result is false then */
    else if(result == 1)
    {
        /* distruct the packet */
        /*** Omssion ***/

        /* print to the message */
        printf("This packet has the possibility that it breaks.This packet
was distructed.¥n");
    }
}
/* If process infomation contents c is Dec then */
else if(PID_addr->proc_c == 2)
{
    /* If next header is TCP then */
    if(ip->ip_p == 0x06)
    {
        /* input the Nextheader address */
        tcp = (struct tcphdr * )nexthead;

        /* calculate address of user data */
        udat_addr = (u_char *)ip + (ip->ip_hl << 2) + (tcp->th_off << 2);

        /* calculate length of user data */
        length = ip->ip_len - (ip->ip_hl << 2) - (tcp->th_off << 2);
    }
    /* If next header is UDP then */
    else if(ip->ip_p == 0x17)
    {
        /* input the Nextheader address */
        udp = (struct udphdr * )nexthead;

        /* calculate address of user data */
        udat_addr = (u_char *)ip + (ip->ip_hl << 2) + 8;
        /* calculate length of user data */
        length = udp->uh_ulen - 8;
    }

    /* If next header is TCP or UDP then */
    if(ip->ip_p == 0x06 || ip->ip_p == 0x17)
    {
        /* make simulated data */
        make_simdat(iv, udat_addr, length, simdat);

        /* verify checksum (1) */
        result = verify_cksum(ip, nexthead, simdat, 1);

        /* If verification result is true then */
        if(result == 0)
        {
            /* decrypto */
            crypto(udat_addr, length, key_addr, iv, 0);

            /* recalculate checksum (0) */
            recalc_cksum(ip, nexthead, simdat, 0);
        }
        /* If verification result is false then */
        else if(result == 1)
        {
            /* distruct the packet */
            /*** Omssion ***/

            /* print to the message */
            printf("This packet has the possibility that it breaks or it was
altered.This packet was distructed.¥n");
        }
    }
    /* Exception */
    else printf("Exceptional error. This packet is not TCP or UDP.¥nError
message from cip_proc_t().¥n");
}
}
```

CipherProcess.h

```
/* CipherProcess.h */
/*
* Copyright (C) 2003-2004
* Shinya Masuda
* Watanabe Laboratory,
* Meijo University in Japan.
* All rights reserved.
*
* This program is written by Shinya Masuda (a3003j125@ccmailg.meijo-u.ac.jp).
* This program is free to use.
*
* This program includes software developed by the OpenSSL Project
* for use in the OpenSSL Toolkit (http://www.openssl.org/).
* This program includes cryptographic software written by Eric Young
* (eay@cryptsoft.com).
*
* ---- Explanation of this program ----
* Range of message authentication is part of encryption and field to
* use for a making of IV (Initialization Vector).
*
* Part of encryption and field to use for a making of IV is different
* by the existance of the NA(P)T in a network environmento to presume.
*
* This program is able to setup of the existance of the NA(P)T by the
* value of "#define NAPT".
*
* NA(P)T exist:
*   Range of encryption: TCP payload
*   Field to use for a making of IV:
*       IP header: Version,IHL,Total Length,ID,Protocol,(Option)
*       TCP header: Sequence Number,Acknowledgement Number,
*                   Data Offset,Reserved,Control Flag,Window,
*                   Urgent Pointer,(Option)
* NA(P)T is Nothing:
*   Range of encryption: TCP payload
*   Field to use for a making of IV:
*       IP header: Version,IHL,Total Length,ID,Protocol,
*                  Source Address,Destination Address,(Option)
*       TCP header : Source Port,Destination Port,Sequence Number,
*                    Ackowledgement Number,Data Offset,Reserved,
*                    Control Flag,Window,Urgent Pointer,(Option)
*
* This program take no thought to mbuf and flagment.
*
*/


#include <sys/param.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in_systm.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <netinet/udp.h>
#include "param.h"

#define IV_SIZE 16      /* IV size (byte) */
#define GKEY_SIZE 32        /* groupe key size (byte) */

#define NAPT 0          /* NA(P)T  1:exist 0:nothing */

/*************************************************************
void make_iv(struct ip *ip, void *nexthead, u_char *iv_addr)

function:
make IV module
parametor:
ip      : struct of IP header address (I)
nexthead    : struct of Next header(tcp/udp) address (I)
key_addr    : key address (I)
iv_addr     : Initialization Vector address (I/O)
return:
    ---
*************************************************************/
void make_iv(
struct ip *ip,
void *nexthead,
    u_char *key_addr,
u_char *iv_addr
    );
```

```
/*************************************************
    void crypto(u_char *idat_addr, int length, u_char *key_addr, u_char
*iv_addr, int EncDec)

    function:
    Encryption/Desryption module (AES_CFB)
    parametor:
    idat_addr   : input data address (I/O)
    length      : length of input data (I)
    key_addr    : key address (I)
    iv_addr     : Initialization Vector address (I/O)
    EncDec      : AES_ENCRYPT(1):ecnryption
                  AES_DECRYPT(0):decryption  (I)
    return:
    ---
*************************************************/
void crypto(
    u_char *idat_addr,
    int length,
    u_char *key_addr,
    u_char *iv_addr,
    int EncDec
    );


/*********************************************************
    void make_simdat(u_char *iv_addr, u_char *encdat_addr, int length,
u_char *simdat_addr)

    function:
    make simulated data module
    parametor:
    iv_addr     : Initialization Vector address (I)
    encdat_addr : encrypted data address (I)
    length      : length of encrypted data (I)
    simdat_addr : simulated data address (I/O)

    return:
    ---
*********************************************************/
void make_simdat(
    u_char *iv_addr,
    u_char *encdat_addr,
    int length,
    u_char *simdat_addr
    );


/*********************************************************
    void recalc_cksum(struct ip *ip, void *nexthead, u_char *simdat_addr,
int mode)

    function:
    recalculate checksum module
    parametor:
    ip      : struct of IP header address (I)
    nexthead    : struct of Next header(tcp/udp) address (I/O)
    simdat_addr : simulated data address (I)
    mode        : (0):nomal
                  (1):contain simulated data (I)
    return:
    ---
*********************************************************/
void recalc_cksum(
    struct ip *ip,
        void *nexthead,
            u_char *simdat_addr,
        int mode
            );


/*********************************************************
    int verify_cksum(struct ip *ip, void *nexthead, u_char *simdat_addr, int
mode)

    function:
    verify checksum module
    parametor:
    ip      : struct of IP header address (I)
    nexthead    : struct of Next header(tcp/udp) address (I)
    simdat_addr : simulated data address (I)
    mode        : (0):nomal
                  (1):contain simulated data (I)
    return:
    true(0) or false(1) (int)
*********************************************************/
int verify_cksum(
    struct ip *ip,
    void *nexthead,
        u_char *simdat_addr,
    int mode
);


/*********************************************************
    void cip_proc_t(struct ip *ip, void *nexthead, PID *PID_addr)

    function:
    cipher process test module
    parametor:
    ip      : struct of IP header address (I)
    nexthead    : struct of Next header(tcp/udp) address (I)
    PID_addr    : struct of process infomation data (I)
    return:
    ---
 *********************************************************/
 void cip_proc_t(
     struct ip *ip,
     void *nexthead,
     PID *PID_addr
);


/* struct of IP_Option 1 byte */
struct ipopt
{
#if BYTE_ORDER == LITTLE_ENDIAN
    unsigned int    opt_optnum:5,   /* option number */
    opt_cls:2,  /* class */
    opt_c:1;    /* copy */
#endif
#if BYTE_ORDER == BIG_ENDIAN
    unsigned int    opt_c:1,    /* copy */
    opt_cls:2,  /* class */
    opt_optnum:5;   /* option number */
#endif
};
```

param.h

```
/* param.h */

/* struct of process infomation data */
typedef struct proc_info_data
{
    u_long  saddr;      /* source address */
    u_long  daddr;      /* dest address */
#if BYTE_ORDER == LITTLE_ENDIAN
    u_char  proc_b:4,   /* process infomation contents */
    proc_a:4;   /* process infomation contents */
#endif
#if BYTE_ORDER == BIG_ENDIAN
    u_char  proc_a:4,   /* process infomation contents */
    proc_b:4;   /* process infomation contents */
#endif
    u_char  proc_c;     /* process infomation contents */
    u_short g_no;       /* group number */
    u_short key_ver;    /* group key version */
    u_char  timer;      /* timer value */
} PID;

/*
proc_c
5: (unused)
4: CREATE
3: FWD
2: DEC
1: ENC
g_no
1: gkey1
2: gkey2
*/
```

driver.c

```
/* driver.c */
/*
 * Copyright (C) 2003-2004
 * Shinya Masuda
 * Watanabe Laboratory,
 * Meijo University in Japan.
 * All rights reserved.
 *
 * This program is written by Shinya Masuda
(a300j125@ccmailg.meijo-u.ac.jp).
```

```
 * This program is free to use.
 *
 * ---- Explanation of this program ----
 * This program is test driver for "cip_proc_t()".
 * This program print to the state of a packet and processing speed,
 * when the subroutine "cip_proc_t()" was made to run to process the
 * encryption and the message authentication of a packet.
 *
 * This program make a packet, and it process the packet.
 * But, the packet does not send. And, this program omits the UDP packet.
 * This program take no thought to mbuf.
 * Furhtermore, this program in order to simplify processing, the
 * IP/TCP/UDP header is created uniquely.
 *
 */

#include <stdio.h>
#include <time.h>
#include <sys/socket.h> /* to call ntohs(), inet_ntoa */
#include "CipherProcess.h"

#define PRINT_PACKET 1  /* print to the packet      1:yes 0:no */
#define ALTER 0     /* alteration test      1:yes 0:no */
#define TIME 0      /* print to the processing time 1:yes 0:no */


/*********************************************************
 char *tos_str(int num)

 function:
    convert int into char for IP TOS
 parametor:
    num : IP tos (I)
 return:
    char *
*********************************************************/
char *tos_str(int num)
{
    /* converted string */
    static char str[9];
    /* loop variable */
    int i;


    str[8] = '¥0';
    /* binary number of 8 figures */
    for(i = 0; i < 8; i++){
        if(i != 0 && num == 0)
            str[7 - i] = '0';
        if(num % 2 == 0)
            str[7 - i] = '0';
        else
            str[7 - i] = '1';
        num /= 2;
    }


    return str;
}


/*********************************************************
 char *flags_str(int num)

 function:
    convert int into char for IP flags
 parametor:
    num : IP flags (I)
 return:
    char *
*********************************************************/
char *flags_str(int num)
{
    /* converted string */
    static char str[4];
    /* loop variable */
    int i;


    str[3] = '¥0';
    /* binary number of 3 figures */
    for(i = 0; i < 3; i++){
        if(i != 0 && num == 0)
            str[2 - i] = '0';
        if(num % 2 == 0)
            str[2 - i] = '0';
        else
            str[2 - i] = '1';
        num /= 2;
    }


    return str;
}



/*********************************************************
 void print_ip(struct ip *ip)

 function:
    print to the IP header
 parametor:
    ip  : struct of IP header address (I)
 return:
    ---
*********************************************************/
void print_ip(struct ip *ip)
{
    printf(" IP header¥n");

printf("+-------+------+-------------+---------------------------+¥n"
);
    printf("| Ver: %u| IHL:%2u| TOS: %8s| Total Length:    %10u|¥n",
        ip->ip_v, ip->ip_hl, tos_str(ip->ip_tos), ip->ip_len);

printf("+-------+------+-------------+-------+-------------------+¥n"
);
    printf("| Identication:           %5u|Flg:%3s| FO:           %5u|¥n",
        ntohs(ip->ip_id), flags_str(ntohs(ip->ip_off)),
        ntohs(ip->ip_off) & IP_OFFMASK);

printf("+---------------+-------------+-------+-------------------+¥n"
);
    printf("| TTL:       %3u| Protocol: %3u| Header
Checksum:      %5u|¥n",
        ip->ip_ttl, ip->ip_p, ntohs(ip->ip_sum));

printf("+---------------+-------------+---------------------------+¥n"
);
    printf("| Source Address:                           %15s|¥n",
        inet_ntoa(*(struct in_addr *)&(ip->ip_src)));

printf("+-----------------------------------------------------------+¥n"
);
    printf("| Destination Address:                      %15s|¥n",
        inet_ntoa(*(struct in_addr *)&(ip->ip_dst)));

printf("+-----------------------------------------------------------+¥n"
);

}


/*********************************************************
 char *cf_str(int num)

 function:
    convert int into char for IP CF
 parametor:
    num : IP CF (I)
 return:
    char *
*********************************************************/
char *cf_str(int num)
{
    /* converted string */
    static char str[7];
    /* loop variable */
    int i;


    str[6] = '¥0';
    /* binary number of 6 figures */
    for(i = 0; i < 6; i++){
        if(i != 0 && num == 0)
            str[5 - i] = '0';
        if(num % 2 == 0)
            str[5 - i] = '0';
        else
            str[5 - i] = '1';
        num /= 2;
    }


    return str;
}
```

```
/*********************************************************
 void print_tcp(struct tcphdr *tcp)

 function:
    print to the TCP header
 parametor:
    tcp : struct of TCP header address (I)
 return:
    ---
*********************************************************/
void print_tcp(struct tcphdr *tcp)
{
    printf(" TCP header¥n");

printf("+-------------------------------+-------------------------------+¥n"
);
    printf("| Source Port:             %5u| Destination Port:      %5u|¥n",
        ntohs(tcp->th_sport), ntohs(tcp->th_dport));

printf("+-------------------------------+-------------------------------+¥n"
);
    printf("| Sequence Number:                                %10lu|¥n",
        (u_long)ntohl(tcp->th_seq));

printf("+---------------------------------------------------------------+¥n"
);
    printf("| Acknowledgement Number:                         %10lu|¥n",
        (u_long)ntohl(tcp->th_ack));

printf("+-------+-------+-------+-------------------------------+¥n"
);
    printf("| DO: %2u| Reserved | CF: %6s| Window:                %5u|¥n",
        tcp->th_off, cf_str(tcp->th_flags), ntohs(tcp->th_win));

printf("+-------+-------+-------+-------------------------------+¥n"
);
    printf("| Checksum:               %5u| Urgent Pointer:        %5u|¥n",
        ntohs(tcp->th_sum), ntohs(tcp->th_urp));

printf("+-------------------------------+-------------------------------+¥n"
);

}


/*********************************************************
 void print_tcppay(u_char *tcppay_addr, int length)

 function:
    print to the TCP payload
 parametor:
    tcppay_addr : TCP payload address (I)
    length      : TCP payload length (I)
 return
    ---
*********************************************************/
void print_tcppay(u_char *tcppay_addr, int length)
{
    /* loop variable */
    int i;
    /* quotient */
    int quot;


    /* calculate quotient of "length / 19" */
    quot = (int)(length / 19);

    printf( "TCP payload¥n" );

printf("+---------------------------------------------------------------+¥n|
");
        for( i=0; i<length; i++ ){
                if ((i% 19 == 0) && (i != 0))
            printf("
|¥n+---------------------------------------------------------------+¥n|");
                printf( " %02x", *(tcppay_addr + i));
        }
        for( i=0; i<(19*(quot+1)-length); i++ ) printf("   ");
        printf("
|¥n+---------------------------------------------------------------+¥n");

}


/*********************************************
 int main(int argc, char *argv[])
 function:
    Test Driver
 return:
    0
*********************************************/
int main(int argc, char *argv[])
{
    /* loop variable */
    long i;
    /* start time, end time */
    clock_t start, end;
    /* packet buffer */
    u_char buff[2048];
    /* struct of IP header address */
    struct ip *ip;
    /* struct of TCP header adress */
    struct tcphdr *tcp;
    /* TCP payload address */
    u_char *tcppay;
    /* struct of process infomation data */
    PID PID_e1, PID_e2, PID_d1, PID_d2;
    /* payload length */
    int length;
    /* select key number */
    int keyn;


    /* set address for IP header */
    ip = (struct ip *)buff;
    /* set address for TCP header */
    tcp = (struct tcphdr *)(buff + 20);
    /* set address for TCP payload */
    tcppay = (buff + 40);

    /* input value of packet*/
    /* IP header */
    ip->ip_v = 0x4;     /* header length */
    ip->ip_hl = 0x5;            /* version */
    ip->ip_tos = 0x00;          /* type of service */
    ip->ip_len = 0x0190;        /* total length */
    /* 64:0x0040, 128:0x0080, 256:0x0100, 512:0x0200, 1024:0x0400 */
    /* 200:0x00c8, 400:0x0190, 600:0x0258, 800:0x0320, 1000:0x03e8 */
    ip->ip_id = 0x2684;     /* identification */
    ip->ip_off = 0x0000;        /* fragment offset field */
    ip->ip_ttl = 0x80;          /* time to live */
    ip->ip_p = 0x06;        /* protocol */
    ip->ip_sum = 0x6078;            /* checksum */
    ip->ip_src.s_addr = 0x0e01a8c0;         /* source address */
    ip->ip_dst.s_addr = 0xf6208dcb;         /* destination address */
    /* TCP header*/
    tcp->th_sport = 0xd212;         /* source port */
    tcp->th_dport = 0x5000;         /* destination port */
    tcp->th_seq = 0xcaf7b6af;       /* sequence number */
    tcp->th_ack = 0x86fa5007;       /* acknowledgement number */
    tcp->th_off = 0x5;              /* data offset */
    tcp->th_x2 = 0x0;               /* (unused) */
    tcp->th_flags = 0x02;           /* flag */
    tcp->th_win = 0xffff;           /* window */
    tcp->th_sum = 0xf025;           /* checksum */
    tcp->th_urp = 0x0000;           /* urgent pointer */
    /* TCP payload */
    /* calculate length of TCP payload */
    length = ip->ip_len - (ip->ip_hl << 2) - (tcp->th_off << 2);
    for(i=0; i<length; i++) *(tcppay + i) = 0x77;

    /* input value of PID */
    PID_e1.proc_c = 1;  /* Enc */
    PID_e1.g_no = 1;    /* gkey1 */
    PID_e2.proc_c = 1;  /* Enc */
    PID_e2.g_no = 2;    /* gkey2 */
    PID_d1.proc_c = 2;  /* Dec */
    PID_d1.g_no = 1;    /* gkey1 */
    PID_d2.proc_c = 2;  /* Dec */
    PID_d2.g_no = 2;    /* gkey2 */

    printf("¥nNAPT=%d (0:nothing, 1:exist)¥n", NAPT);
    printf("Packet length:%d¥n", ip->ip_len);

#if PRINT_PACKET
    printf( "------------------------------ plain packet
------------------------------¥n" );
    /* print to the IP header */
    print_ip(ip);
    /* print to the TCP header */
    print_tcp(tcp);
    /* print to the TCP payload */
    print_tcppay(tcppay, length);
#endif

    /* select key number */
    printf("Please select key number (1:key1  2:key2) :");
```

```
    scanf("%d", &keyn);

    switch(keyn)
    {
        case 1:
#if TIME
            printf("processing now ...¥n");
            /* start timer */
            start = clock();
            for(i=0; i<10000000; i++)
            {
#endif
                /* call cip_proc_t() - enc gkey1 - */
                cip_proc_t(ip, tcp, &PID_e1);
#if TIME
            }
            /* end timer */
            end = clock();
#endif
            break;
        case 2:
#if TIME
            printf("processing now ...¥n");
            /* start timer */
            start = clock();
            for(i=0; i<10000000; i++)
            {
#endif
                /* call cip_proc_t() - enc gkey2 - */
                cip_proc_t(ip, tcp, &PID_e2);
#if TIME
            }
            /* end timer */
            end = clock();
#endif

            break;
        default:
            break;
    }

#if PRINT_PACKET
    printf( "---------------------------- encrypted packet
----------------------------¥n" );
    /* print to the IP header */
    print_ip(ip);
    /* print to the TCP header */
    print_tcp(tcp);
    /* print to the TCP payload */
    print_tcppay(tcppay, length);

#endif
#if TIME
    /* print to the processing time */
    printf("processing time : %f ms¥n", (double)(end - start) /
CLOCKS_PER_SEC * 1000 / 10000000);
#endif

#if ALTER
    /* alteration test */
    int yn;
    int select;
    printf("Alter? (yes:1,no:2) :");
    scanf("%d", &yn);
    printf("¥n");
    if(yn==1)
    {
        printf(" 1:IP Identifier    (authen)        : 33830 -> 29734¥n");
        printf(" 2:IP Source IP Address (NAPT=0 authen)     : 192.168.1.14
-> 192.168.1.10¥n");
        printf(" 3:TCP DataOffset   (authen)        : 5 -> 6¥n");
        printf(" 4:TCP Source Port  (NAPT=0 authen)     : 4818 -> 5074¥n");
        printf("authen : authentication range¥n");
        printf("Please select a number :");
        scanf("%d", &select);
        printf("¥nThis packet was altered!¥n");
        switch(select)
        {
            case 1:
                ip->ip_id = 0x2674;     /* 0x2684 */
                break;
            case 2:
                ip->ip_src.s_addr = 0x0a01a8c0;     /* 0x0e01a8c0 */
                break;
            case 3:
                tcp->th_off = 0x6;      /* 0x5 */
                break;
            case 4:
                tcp->th_sport = 0xd213 ;    /* 0xd212 */
                break;
            default:
                break;
        }
    }
#endif
    /* select key number */
    printf("Please select key number (1:key1  2:key2) :");
    scanf("%d", &keyn);

    switch(keyn)
    {
        case 1:
#if TIME
            printf("processing now ...¥n");
            /* start timer */
            start = clock();
            for(i=0; i<10000000; i++)
            {
#endif
                /* call cip_proc_t() - dec gkey1 - */
                cip_proc_t(ip, tcp, &PID_d1);
#if TIME
            }
            /* end timer */
            end = clock();
#endif
            break;
        case 2:
#if TIME
            printf("processing now ...¥n");
            /* start timer */
            start = clock();
            for(i=0; i<10000000; i++)
            {
#endif
                /* call cip_proc_t() - dec gkey2 - */
                cip_proc_t(ip, tcp, &PID_d2);
#if TIME
            }
            /* end timer */
            end = clock();
#endif

            break;
        default:
            break;
    }

#if PRINT_PACKET
    printf( "---------------------------- decrypted packet
----------------------------¥n" );
    /* print to the IP header */
    print_ip(ip);
    /* print to the TCP header */
    print_tcp(tcp);
    /* print to the TCP payload */
    print_tcppay(tcppay, length);
#endif
#if TIME
    /* print to the processing time */
    printf("processing time : %f ms¥n", (double)(end - start) /
CLOCKS_PER_SEC * 1000 / 10000000);
#endif

    return 0;
```

I　OpenSSL 暗号ライブラリ DES・AES テストプログラム

DES・AES のテストプログラム一覧

- des_ecb.c
- des_cbc.c
- des_cfb.c

- aes_ecb.c
- aes_cbc.c
- aes_cfb.c

OpenSSL （Release version : openssl-0.9.7a 以上）暗号ライブラリをインストール済みの Linux，FreeBSD で動作確認済み．環境を整えれば Windows での使用も可能．

gcc コンパイラの場合，コンパイル時に OpenSSL ライブラリの指定が必要．

例）
$gcc des_ecb.c /usr/local/ssl/lib/libcrypto.a

des_ecb.c

```
#include <stdio.h>
#include <string.h>
#include <openssl/des.h>

#define LENGTH 5

static unsigned char
key_data1[8]={0x01,0x23,0x45,0x67,0x89,0xAB,0xCD,0xEF};
static unsigned char
key_data2[8]={0x11,0x33,0xD5,0x55,0xA9,0xAa,0xDD,0xFF};

int main(int argc, char *argv[])
{
    int i;
    DES_cblock in,out,outin;
    DES_key_schedule ks;

    DES_set_key_unchecked(&key_data1,&ks);
    memset(in,0x77,LENGTH);
    memset(out,0,LENGTH);
    memset(outin,0,LENGTH);
    DES_ecb_encrypt(&in,&out,&ks,DES_ENCRYPT);
    DES_set_key_unchecked(&key_data1,&ks);
    DES_ecb_encrypt(&out,&outin,&ks,DES_DECRYPT);

    printf("Plain text:   ");
    for (i = 0; i < LENGTH; i++)
            printf(" %02x", in[i]);
    putchar('¥n');

    printf("Encrypted:");
    for (i = 0; i < LENGTH; i++)
            printf(" %02x", out[i]);
    putchar('¥n');

    printf("Decrypted:");
    for (i = 0; i < LENGTH; i++)
            printf(" %02x", outin[i]);
    putchar('¥n');
    return 0;
}
```

des_cbc.c

```
#include <stdio.h>
#include <string.h>
#include <openssl/des.h>

#define LENGTH 5

unsigned char key_data1[8]={0x01,0x23,0x45,0x67,0x89,0xAB,0xCD,0xEF};
unsigned char key_data2[8]={0x11,0x33,0x55,0x77,0x99,0xAA,0xCC,0xEE};
unsigned char iv_data1[8]={0x34,0x11,0xA1,0xE9,0x01,0x58,0xB1,0x35};
unsigned char iv_data2[8]={0xd4,0x31,0xA5,0xE3,0x71,0x78,0xB2,0x45};

int main(int argc, char *argv[])
{
    int i;

    unsigned char in[LENGTH];
    unsigned char out[LENGTH];
    unsigned char outin[LENGTH];

    memset(in,0x77,LENGTH);
    memset(out,0,LENGTH);
    memset(outin,0,LENGTH);

    DES_cblock ivec;
    DES_key_schedule ks;

    printf("Plain text:   ");
    for (i = 0; i < LENGTH; i++)
            printf(" %02x", in[i]);
    putchar('¥n');

    memcpy(ivec,iv_data1,8);

    printf("IV:");
    for (i = 0; i < 8; i++)
        printf("%02x", ivec[i]);
    putchar('¥n');

    DES_set_key_unchecked(&key_data1,&ks);
    printf("*** Enc! ***¥n");
    DES_cbc_encrypt(in,out,LENGTH,&ks,&ivec,DES_ENCRYPT);

    printf("Plain text:   ");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", in[i]);
    putchar('¥n');
    printf("Encrypted:");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", out[i]);
    putchar('¥n');

    printf("IV before :");
    for (i = 0; i < 8; i++)
        printf("%02x", ivec[i]);
    putchar('¥n');

    memcpy(ivec,iv_data1,8);

    printf("IV after:");
    for (i = 0; i < 8; i++)
        printf("%02x", ivec[i]);
    putchar('¥n');

    DES_set_key_unchecked(&key_data1,&ks);
    printf("*** Dec! ***¥n");
    DES_cbc_encrypt(out,outin,LENGTH,&ks,&ivec,DES_DECRYPT);

    printf("Plain text:   ");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", in[i]);
    putchar('¥n');
    printf("Encrypted:");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", out[i]);
    putchar('¥n');
    printf("Decrypted:");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", outin[i]);
    putchar('¥n');
    printf("IV:");
    for (i = 0; i < 8; i++)
        printf("%02x", ivec[i]);
    putchar('¥n');

    return 0;
}
```

des_cfb.c

```
#include <stdio.h>
#include <string.h>
#include <openssl/des.h>

#define LENGTH 17

unsigned char key_data1[8]={0x01,0x23,0x45,0x67,0x89,0xAB,0xCD,0xEF};
unsigned char key_data2[8]={0x11,0x33,0x55,0x77,0x99,0xAA,0xCC,0xEE};
unsigned char iv_data1[8]={0x34,0x11,0xA1,0xE9,0x01,0x58,0xB1,0x35};
unsigned char iv_data2[8]={0xd4,0x31,0xA5,0xE3,0x71,0x78,0xB2,0x45};

int main(int argc, char *argv[])
{
    int i;

    unsigned char in[LENGTH];
    unsigned char out[LENGTH];
    unsigned char outin[LENGTH];

    int numbits = 8;

    memset(in,0x77,LENGTH);
    memset(out,0,LENGTH);
    memset(outin,0,LENGTH);


    DES_cblock ivec;
    DES_key_schedule ks;

    printf("Plain text:   ");
    for (i = 0; i < LENGTH; i++)
            printf(" %02x", in[i]);
    putchar('¥n');

    memcpy(ivec,iv_data1,8);

    printf("IV:");
```

```
    for (i = 0; i < 8; i++)
        printf("%02x", ivec[i]);
    putchar('¥n');

    DES_set_key_unchecked(&key_data1,&ks);
    printf("*** Enc! ***¥n");
    DES_cfb_encrypt(in,out,numbits,LENGTH,&ks,&ivec,DES_ENCRYPT);

    printf("Plain text:   ");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", in[i]);
    putchar('¥n');
    printf("Encrypted:");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", out[i]);
    putchar('¥n');

    printf("IV before :");
    for (i = 0; i < 8; i++)
        printf("%02x", ivec[i]);
    putchar('¥n');

    memcpy(ivec,iv_data1,8);

    printf("IV after:");
    for (i = 0; i < 8; i++)
        printf("%02x", ivec[i]);
    putchar('¥n');

    DES_set_key_unchecked(&key_data1,&ks);
    printf("*** Dec! ***¥n");
    DES_cfb_encrypt(out,outin,numbits,LENGTH,&ks,&ivec,DES_DECRYPT);

    printf("Plain text:   ");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", in[i]);
    putchar('¥n');
    printf("Encrypted:");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", out[i]);
    putchar('¥n');
    printf("Decrypted:");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", outin[i]);
    putchar('¥n');
    printf("IV:");
    for (i = 0; i < 8; i++)
        printf("%02x", ivec[i]);
    putchar('¥n');

    return 0;
}
```

aes_ecb.c

```
#include <stdio.h>
#include <string.h>
#include <openssl/aes.h>
#define LENGTH 5

const unsigned char
key_data1[16]={0x01,0x23,0x45,0x67,0x89,0xAB,0xCD,0xEF,
               0x01,0x23,0x45,0x67,0x89,0xAB,0xCD,0xEF};
const unsigned char
key_data2[16]={0x11,0x33,0xA5,0x77,0x99,0xCB,0x1D,0xFF,
               0x11,0x2D,0x35,0x65,0x99,0x1B,0x2D,0x3F};

int main(int argc, char *argv[])
{
    int i;
    unsigned char in[LENGTH];
    unsigned char out[LENGTH];
    unsigned char outin[LENGTH];
    AES_KEY ks;

    memset(in,0x77,LENGTH);
    memset(out,0,LENGTH);
    memset(outin,0,LENGTH);

    printf("Plain text:   ");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", in[i]);
    putchar('¥n');

    AES_set_encrypt_key(key_data1,128,&ks);
    printf("*** Enc! ***¥n");
    AES_ecb_encrypt(in,out,&ks,AES_ENCRYPT);

    printf("Plain text:   ");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", in[i]);
    putchar('¥n');
    printf("Encrypted:");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", out[i]);
    putchar('¥n');

    AES_set_decrypt_key(key_data1,128,&ks);
    printf("*** Dec!  ***¥n");
    AES_ecb_encrypt(out,outin,&ks,AES_DECRYPT);

    printf("Plain text:   ");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", in[i]);
    putchar('¥n');
    printf("Encrypted:");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", out[i]);
    putchar('¥n');
    printf("Decrypted:");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", outin[i]);
    putchar('¥n');

    return 0;
}
```

aes_cbc.c

```
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <openssl/aes.h>

#define LENGTH 64

static unsigned char
key_data1[16]={0x01,0x23,0x45,0x67,0x89,0xAB,0xCD,0xEF,
               0x11,0x33,0x35,0x77,0x99,0xA1,0xCC,0xFF};
static unsigned char
key_data2[16]={0x11,0x24,0x55,0x37,0x39,0xAA,0xDD,0xEE,
               0x10,0x30,0x31,0x72,0x96,0xA2,0xCF,0xFD};
static unsigned char
iv_data[16]={0x21,0x25,0xD5,0xA7,0x88,0xAB,0xCD,0xEF,

0x11,0x33,0x35,0x77,0x99,0xD1,0xFC,0xF1};

int main(int argc, char *argv[])
{
    int i;
    unsigned char in[LENGTH];
    unsigned char out[LENGTH];
    unsigned char outin[LENGTH];
    unsigned char ivec[16];
    AES_KEY ks;
    clock_t start, end;

    memset(in,0x77,LENGTH);
    memset(out,0,LENGTH);
    memset(outin,0,LENGTH);

    printf("Plain text:   ");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", in[i]);
    putchar('¥n');

    memcpy(ivec,iv_data,16);

    printf("IV:");
        for(i=0;i<16;i++) printf("%02x", ivec[i]);
    printf("¥n");

    AES_set_encrypt_key(key_data1,128,&ks);
    printf("*** Enc! ***¥n");

    AES_cbc_encrypt(in,out,(unsigned long)LENGTH,&ks,ivec,AES_ENCRYPT);

    printf("Plain text:   ");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", in[i]);
    putchar('¥n');
    printf("Encrypted:");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", out[i]);
    putchar('¥n');
```

```
    printf("IV before:");
    for(i=0;i<16;i++) printf("%02x", ivec[i]);
    printf("¥n");

    memcpy(ivec, iv_data, 16);

    printf("IV after:");
    for(i=0;i<16;i++) printf("%02x", ivec[i]);
    printf("¥n");

    AES_set_decrypt_key(key_data1, 128, &ks);
    printf("*** Dec! ***¥n");
    AES_cbc_encrypt(out, outin, (unsigned
long)LENGTH, &ks, ivec, AES_DECRYPT);

    printf("Plain text:   ");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", in[i]);
    putchar('¥n');
    printf("Encrypted:");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", out[i]);
    putchar('¥n');
    printf("Decrypted:");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", outin[i]);
    putchar('¥n');
    printf("IV:");
    for(i=0;i<16;i++) printf("%02x", ivec[i]);
    printf("¥n");

    return 0;
}
```

aes_cfb.c

```
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <openssl/aes.h>

#define LENGTH 64
#define DEFF_KEY 0 /* 0:same key, 1:defferent key */
static unsigned char
key_data1[16]={0x01, 0x23, 0x45, 0x67, 0x89, 0xAB, 0xCD, 0xEF,
               0x11, 0x33, 0x35, 0x77, 0x99, 0xA1, 0xCC, 0xFF};
static unsigned char
key_data2[16]={0x11, 0x24, 0x55, 0x37, 0x39, 0xAA, 0xDD, 0xEE,
               0x10, 0x30, 0x31, 0x72, 0x96, 0xA2, 0xCF, 0xFD};
static unsigned char
iv_data1[16]={0x21, 0x25, 0xD5, 0xA7, 0x88, 0xAB, 0xCD, 0xEF,

0x11, 0x33, 0x35, 0x77, 0x99, 0xD1, 0xFC, 0xF1};
static unsigned char
iv_data2[16]={0x31, 0x55, 0xB5, 0xA4, 0x98, 0xA1, 0xFD, 0xEF,

0x11, 0x33, 0xD5, 0xAA, 0x59, 0x3C, 0x4C, 0xF2};


int main(int argc, char *argv[])
{
    int i;
    unsigned char in[LENGTH];
    unsigned char out[LENGTH];
    unsigned char outin[LENGTH];
    unsigned char iv[16];
    int num;
    AES_KEY ks;


    memset(in, 0x77, LENGTH);
    memset(out, 0, LENGTH);
    memset(outin, 0, LENGTH);

    printf("LENGTH:%d¥n", LENGTH);
    printf("DEFF_KEY:%d¥n", DEFF_KEY);

    printf("*** Plain text ***¥n");
    printf("Plain text:   ");
        for (i = 0; i < LENGTH; i++)
                printf(" %02x", in[i]);
        putchar('¥n');

    memcpy(iv, iv_data1, 16);
    num = 0;
    printf("num:%d¥n", num);

    printf("IV:");
    for(i=0;i<16;i++) printf("%02x", iv[i]);
    printf("¥n");

    AES_set_encrypt_key(key_data1, 128, &ks);
    printf("*** Enc! ***¥n");
    AES_cfb128_encrypt(in, out, (const unsigned
long)LENGTH, &ks, iv, &num, AES_ENCRYPT);

    printf("Plain text:   ");
        for (i = 0; i < LENGTH; i++)
                printf(" %02x", in[i]);
        putchar('¥n');
    printf("Encrypted:");
        for (i = 0; i < LENGTH; i++)
                printf(" %02x", out[i]);
        putchar('¥n');

    printf("IV before:");
        for(i=0;i<16;i++) printf("%02x", iv[i]);
        printf("¥n");

    memcpy(iv, iv_data1, 16);
    num=0;
    printf("num:%d¥n", num);

    printf("IV after:");
        for(i=0;i<16;i++) printf("%02x", iv[i]);
        printf("¥n");

#if DEFF_KEY
    AES_set_encrypt_key(key_data2, 128, &ks);
#else
    AES_set_encrypt_key(key_data1, 128, &ks);
#endif
    printf("*** Dec! ***¥n");
    AES_cfb128_encrypt(out, outin, (unsigned
long)LENGTH, &ks, iv, &num, AES_DECRYPT);

    printf("Plain text:   ");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", in[i]);
    putchar('¥n');
    printf("Encrypted:");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", out[i]);
    putchar('¥n');
    printf("Decrypted:");
    for (i = 0; i < LENGTH; i++)
        printf(" %02x", outin[i]);
    putchar('¥n');
    printf("IV:");
    for(i=0;i<16;i++) printf("%02x", iv[i]);
    printf("¥n");

    printf("num:%d¥n", num);

    return 0;
}
```