共通鍵暗号とそのプログラムの利用

Common-Key Cryptographic and usage of the program

1　はじめに

ネットワークやコミュニケーションの安全性において，コンピュータの処理の最も重要な手段は，暗号化である．暗号には，共通鍵暗号と公開鍵暗号の 2 つがある．ここでは共通鍵暗号方式と既存のアルゴリズムの特徴について説明し，そのプログラムの利用について説明する．

1.1 暗号化

暗号化の仕組みは，一般的に、それぞれ独立した 3 つの軸で分類される．

1. **平文を暗号文に変換する際に用いる演算のタイプ**

   暗号化アルゴリズムは，換字と転置の 2 つの一般原則に基づいている．換字は，平文の各要素（文字，ビット，文字やビットのグループ）を暗号文の各要素に対応づけるもので，転置は，平文の要素が再配列されるものである．このとき，逆変換可能であることが最低条件となる．ほとんどのシステムでは，換字と転置が複数回使用されており，そのようなシステムを合成システムと呼ぶ．

2. **使用される鍵の数**

   送信者と受信者が同じ鍵を使う場合は共通鍵暗号であり，送信者と受信者がそれぞれ異なる鍵を使う場合は公開鍵暗号である．

3. **平文を処理する方法**

   平文を処理する方法として，ブロック暗号とストリーム暗号がある．ブロック暗号は，1 度に 1 つの入力ブロック要素を処理し，それぞれの入力ブロックにつき1つの出力ブロックを生成する．ストリーム暗号では，入力要素は断続的に 1bit ずつ処理され，処理に応じて 1 度に 1 つの出力要素が生成され，それが繰り返される．

2　共通鍵暗号

2.1 共通鍵暗号のモデル

図 1 は，共通鍵暗号の概念図である．平文と呼ばれる判読可能な原文は，見かけ上意味をなさない暗号文と呼ばれる文に変換される．暗号化処理は，暗号アルゴリズムと鍵で構成される．鍵は，平文とは関係のない独立した値である．アルゴリズムは，そのときに使用される鍵によって異なる暗号文を出力する．

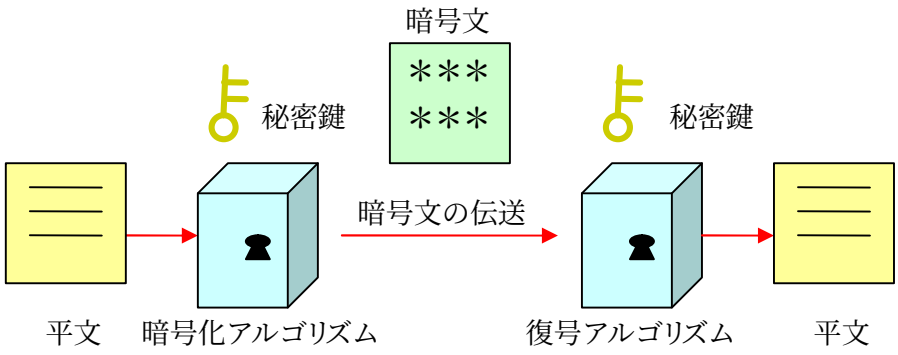


図 1 :共通鍵暗号の概念図

共通鍵暗号の強度は，アルゴリズムが秘密かどうかではなく，鍵の秘密性の高さによって決まる．要するに，暗号文と暗号化/復号アルゴリズムを知っていたとしても，暗号文を復号することはできないということである．ゆえに，暗号アルゴリズムは公開され，かつ十分に安全である必要がある．

3　既存アルゴリズムの特徴

3.1　DES・3DES

DES は IBM が開発し，1977 **年に米国商務省標準局（NBS，現 NIST）が定めた米国標準の**暗号アルゴリズムで，暗号方式としては一番有名な方式である．DES は暗号化する対象を64bit 毎に処理するブロック暗号である．鍵長は 56bit なので，現在では安全とは言えなくなった．1999 年 1 月に開催された，米 RSA の暗号解読コンテスト“DES Challenge III ”で，DES は 22 時間で解読されたのである．

DES が安全でないことから考え出されたのが，DES **の暗号/復号処理を 3 回実行して暗号化を行う 3**DES **という方式である．3**DES は DES 用に開発されている高速な LSI がそのまま利用でき，DES の大きな実績があるなどのメリットがあるので，ビジネス用途として比較的広く使用されている．しかし，処理が重くなるという欠点がある．

3.2　MISTY

MISTY **は** 1995 **年に三菱電機が開発した共通鍵ブロック暗号である．**MISTY **は暗号化する対象を** 64bit **に処理するブロック暗号で，鍵長は** 128bit **である．**MISTY **は，同社の持つ世界最高水準の暗号強度評価技術に基づいて設計されたもので，差分解読法や線形解読法に対して，**DES **よりも十分な安全性を持つことが定量的に証明されている．**

3.3　AES

AES は，DES に代わる米国**次期標準暗号規格**の次世代共通鍵ブロック暗号であり，DES の 64bit ブロックに対し128bit 長のブロックを持ち，さらに鍵の長さを 128，192，256bit としているため，DES に比べて遥かに強固な安全性を持っている．

3.4　Camellia

Camellia は，**2000 年に** NTT と三菱電機が共同で開発した共通鍵ブロック暗号である．Camellia は 128bit 長のブロックを持ち，さらに鍵の長さを 128，192，256bit としているため，AES と同様に強固な安全性を持っている．

4　プログラムの利用

4.1　OpenSSL

OpenSSL は，SSL（Secure Sockets Layer）とTransport Layer Security（TLS）を実装した無償のライブラリで，インターネット上での暗号化したやりとりを可能にするものである．また，DES，3DES，AES を始めとした多数の暗号アルゴリズムを実装している．

4.2　利用形態

プログラムから暗号化の処理を行う関数を呼び出す

実行ファイルを使用して暗号化の処理を行う

4.3　MISTY 1サンプルコードの応用

4.3.1 はじめに

ここでは，MISTY を開発した三菱電機の 松井　充　氏が作成した MISTY 1のサンプルコードhttp://www.security.melco.co.jp/Japanese/MISTY/misty_j.pdf の概観を説明し，これを応用したプログラムについて説明する．

4.3.2 misty1 の入出力関係

暗号化/復号の本体 misty 1では，平文と秘密鍵のアドレスと，テキストのブロック数，暗号化/復号の選択番号（0:暗号化，1:復号）を入力し，暗号文/平文を出力する（図 2）．

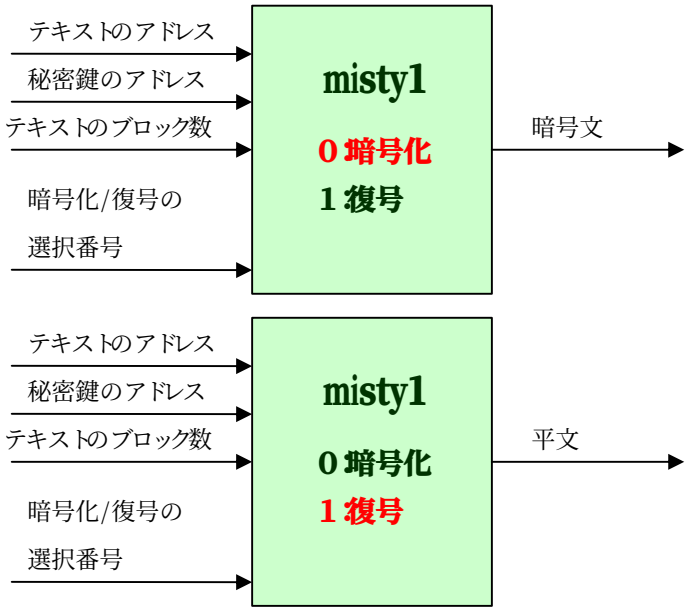


図 2 misty1 の入出力関係

テキストのアドレスを与えると，暗号文は入力したテキストのアドレスに書き込まれる．また，復号したテキストも同様のアドレスに書き込む．

4.3.3 任意文字列を入力

サンプルでは，main 関数内で 16 進 16byte の固定データをテキストとして用意しているので，任意文字列を入力して処理するように変更した．このとき，テキストのブロック数（1 ブロックは 8byte）を算出する必要がある．サンプルでは 16byte の固定データなので，ブロック数は 2と決まっているが，任意の入力の場合は，場合によって要するブロック数は異なる．例えば，18文字の半角英数字を入力した場合，要するブロック数は 3ブロックとなる．

4.3.4 ブロック内の余りの考慮

サンプルでは，16byte の固定データなので，丁度 2 ブロック分になる．そのため，ブロック内の余りが考慮されていない．しかし，任意文字列を入力して処理する場合，ブロック内の余りを考慮する必要がある（図 3）．例えば，5文字の半角英数字を入力した場合，要するブロック数は 1ブロックであるが，このブロック内に 2

byte の余りがでてしまう(文字列の終わりには null である0 が入るため，5文字なら6byte)．ここには他のプログラムのデータが入っている可能性がある．その場合，このまま暗号化してしまうと，そのプログラムに影響が出てしまう．

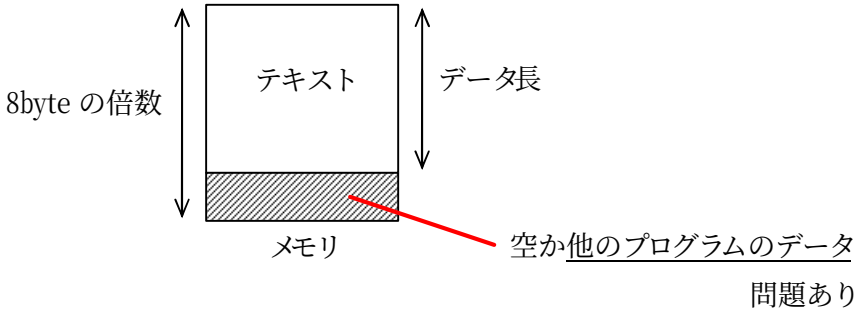
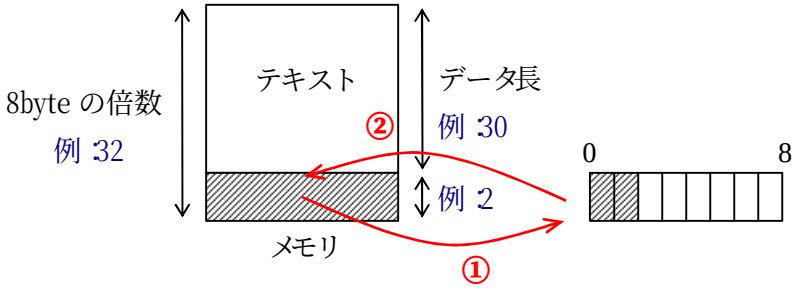


図３：ブロック内の余りの問題

これを回避する方法として，余りのデータをバッファへ退避する方法がある．暗号化後に退避したデータを元に戻すことで，この問題は解決する（図４）．



① 暗号化の前にバッファへ退避
② 暗号化して暗号データを出力後，退避したデータを元に戻す

図４：バッファへ退避することによる解決

具体例の図を以下に示す（図５･６）．

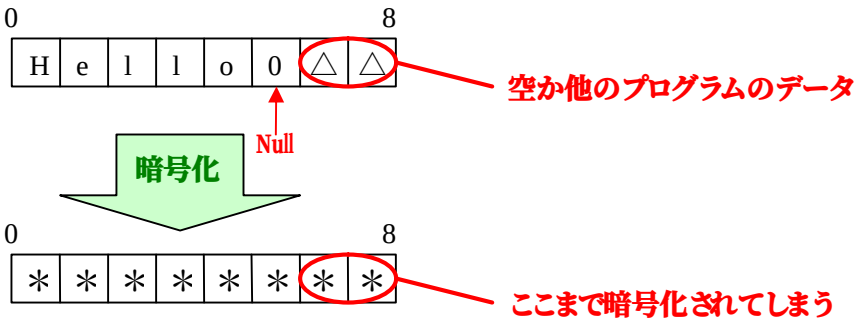


図５：ブロック内の余りの問題の例

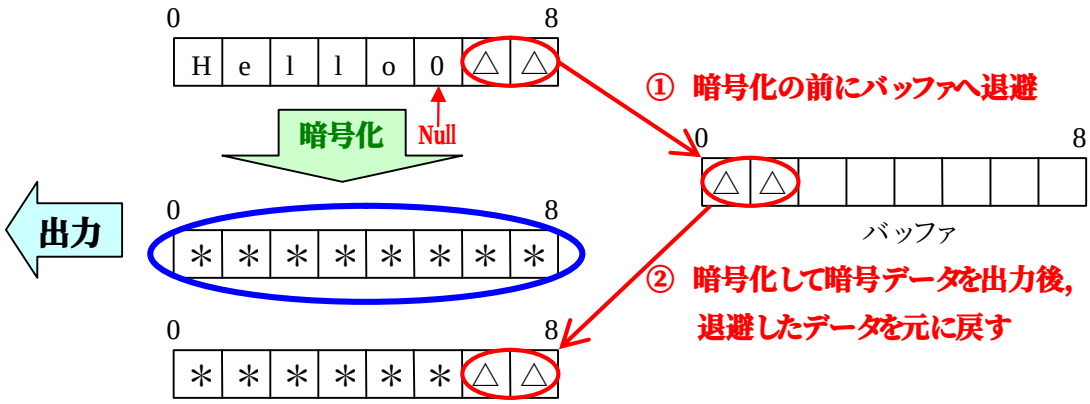


図６バッファへ退避することによる解決の例

4.3.5 変更した main 関数のコード

今回変更した main 関数のコードを以下に示す 図 7).

```
main() {
        /* secret key */
        static uchar key[16] = {0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77,
                0x88,0x99,0xaa,0xbb,0xcc,0xdd,0xee,0xff};
        char text[128];         /* input string */
        int length;             /* string length */
        int block;              /* number of string blocks */
        int i;                  /* loop */
        char *ptext;            /* text pointer */
        int rest;               /* rest of blocks */
        char buf[8];            /* buffer for rest */

        /* input string */
        printf("input string :");
        scanf("%s", text);

        ptext = &(text[0]);
        /* string length */
        length = strlen(text);
        printf("string length :%d\n", length);
        /* number of string blocks */
        block = (int)ceil((double)length / 8);
        printf("block :%d\n", block);
```

```
		/* rest of blocks  length+1's "1" is  a "1" of nulls which indicate the end of the string */
		rest = 8*block - (length+1);
		/* input a data into the rest */
		for( i=0; i<rest; i++ )		*(ptext + length + i + 1) = 'r';
		/* Evacuate a rest data into a buffer*/
		for( i=0; i<rest; i++ ) buf[i] = *(ptext + length + i + 1);

		/* print to the secret key*/
		printf( "secret key :" );
		for( i=0; i<16; i++ ) printf( "%02x ", key[i] );
		printf( "\n" );
		/* print to the plain text*/
		printf( "plain text :" );
		/* for( i=0; i<16; i++ ) printf( "%02x ", text[i] ); */
		for( i=0; i<length; i++ ) printf( "%c", text[i] );
		printf( "\n" );

		/* print to the address of the text and the text */
		for( i=0; i<length; i++) printf("top address of the text ＋%d address and the data:%d / %c\n", i, (int)&*(ptext + i), *(ptext + i));
		/* print to the address of the rest and the rest */
		for( i=0; i<rest+1; i++) printf("end address of the text ＋%d address and the data :%d / %c\n" ,i ,(int)&*(ptext + length + i) , *(ptext + length + i));

		/* call misty1 encryption */
		misty1( text, key, block, 0 );
		/* extended key */
		printf( "extended tey :" );
		for( i=0; i<8; i++ ) printf( "%02x %02x ", (uchar)(EXTKEY[1][i]>>8), (uchar)(EXTKEY[1][i]&0xff) );
		printf( "\n" );
		/* encrypted text*/
		printf( "encrypted text :" );
		/* for( i=0; i<16; i++ ) printf( "%02x ", text[i] ); */
		for( i=0; i<length; i++ ) printf( "%c", text[i] );
		printf( "\n" );
```

```
		/* return the data of buffer */
		for( i=0; i<rest; i++ ) *(ptext + length + i + 1) = buf[i];

		/* print to the address of the text and the text */
		for( i=0; i<length; i++) printf("top address of the text ＋%d address and the data:%d
/ %c\n", i, (int)&*(ptext + i), *(ptext + i));
		/* print to the address of the rest and the rest */
		for( i=0; i<rest+1; i++) printf("end address of the text ＋%d address and the data :%d
/ %c\n" ,i ,(int)&*(ptext + length + i) , *(ptext + length + i));

		/* call misty1　Decryption */
		misty1( text, key, block, 1 );

		/* decrypted text */
		printf( "decrypted text :" );
		/* for( i=0; i<16; i++ ) printf( "%02x ", text[i] ); */
		for( i=0; i<length; i++ ) printf( "%c", text[i] );
		printf( "\n" );

		/* print to the address of the text and the text */
		for( i=0; i<length; i++) printf("top address of the text ＋%d address and the data:%d
/ %c\n", i, (int)&*(ptext + i), *(ptext + i));
		/* print to the address of the rest and the rest */
		for( i=0; i<rest+1; i++) printf("end address of the text ＋%d address and the data :%d
/ %c\n" ,i ,(int)&*(ptext + length + i) , *(ptext + length + i));
}
```

5　最後に

今回，共通鍵暗号の基礎を学び，そのプログラムの簡単な利用を試みた．これにより，今後の目標，課題が見えてきた．

まず，OpenSSL での暗号ライブラリの利用を調査し，MISTY 1プログラムと差し替え可能な入出力関係であるかを考える．これは，一般的な暗号ライブラリの入出力関係に合わせる必要がある．また，パケットの暗号化を考慮したプログラムが必要である．そして，実際にパケットの暗号化・復号を試みる．

6　参考文献

・情報セキュリティ技術　http://www.security.melco.co.jp/SecWWW/

・　暗号とネットワークセキュリティ　理論と実際」
著：W・スターリングス　出版：㈱ピアソン・エデュケーション