Journal of Software Engineering and Applications, **2012, 5, 20-22**
doi:10.4236/jsea.2012.512b004 Published Online December 2012 (http://www.SciRP.org/journal/jsea)

Discussion of New Padding Method in DES Encryption*

Liu Chengxia

Computer Science and Technology Dept. , Beijing Information and Technology University, Beijing, China.
Email: cecilia7812@163.com

Received 2012

ABSTRACT

DES is a kind of block cipher and before DES encryption the plain text be divided into the same-size blocks. But sometimes the plain text can't be divided into the exactly size. So padding step is needed to pad the space of the block. The discussion of the block padding is the emphasis of this paper. A new padding method is given and at the last part of the paper the implementation of DES using new padding method is given.

Keywords: DES; Encryption; Cipher; Block

1. Introduction

DES is the standard of data encryption, and also a kind of block cipher[1]. In block cipher the plain text be di-vided into same size block and use private key to encrypt the plain text into cipher text. In blocking when the length of the block is not as long as 64 bits we usually pad them with zero[2]. Now we will introduce a new method to pad the lacked bits in this paper. This method is not only pad zeros to them but also add the length of zero padded to the last 1 byte. So we can easy get what we pad in the block and can get rid of them fast.

2. New Padding Method

The plain text is divided into 64 bits blocks [3]. Perhaps there is not enough text to fill the last block. How to do with this? The traditional method is padding zero in the end of the block[4]. But it does not distinguish between the padding zero and the original zero. And how can we known where is the beginning of the filling zero?

To solve these problems a new padding method be given. The synopsis of the new padding method is that a "length" part be added to the end of the block.

[step1]: Divided the plain text into 8 bytes(64-bits).If the block size is less than 64 bits then goes to step2.

[step2]: Pad the lacked bits with zero but leave the last 8 bits to memorize the length of padding zeros.

[step3]: Encrypt the plain text with DES algorithm.

*Supported by Funding Project for Academic Human Resources Development in Institutions of Higher Learning Under the Jurisdiction of Beijing Municipality(PHR201008428) and Project for Excellent Talents of the Organization Department of Beijing Municipality (2010 D005007000003), Supported by the General program of science and technology development project of Beijing Municipal Education Commission under Grant No.KM201110772013.

[step4]: Decrypt the cipher text with DES algorithm.

[step5]: Delete the padding zeros according to the length field.

In the new Method there is one problem. How can we known if the last 8 bits is the length information or the original message? To solve this problem an explanation to prove the correctness of the method be given below.

For the last block, firstly, we get the last 8 bits and think it as the length data. Then according to the length data we get the whole padding field. Whether the padding field we got is full of zero? There will be two possible results:

result1: The field is full of zero. So the field is padding field. Delete it and get the real plain text.

result2: The field is not full of zero. So the last 8 bits is not the length of padding but the plain text. Preserve it and take the whole last block as plain text.

Let us see some examples to explain the results.

Ex 1: Part of the block is padded with zero.

Condition: plain text: 01010011 11000000 01111100

00000000, zero padding field: 00000000 00000000 00000000,

Length padding field: 00100000, the block is: 01010011 11000000 01111100 00000000 00000000 00000000 00000000 00100000.

Processing:

First, we get the last 8 bits of the block: 00100000. Then we presume the length of padding field is 32 bits.

Second, we get the last 32 bits data from the block.

Third, we find the 32-8=24(got rid of the length field) bits data is zero.

Forth, so the last 32 bits is padding field and we can de-lete it.

Ex 2: The information block.

Condition: plain text: 01010011 11000000 01111100 00000000 00000000 00000100 00000000 00100000, so the block is: 01010011 11000000 01111100 00000000 00000000 00000100 00000000 00100000.

Processing:

First, we get the last 8 bits of the block:00100000. Then we presume the length of padding field is 32 bits.

Second, we get the last 32 bits data from the block.

Third, we find the 32-8 = 24(got rid of the length field) bits data is not zero.

Forth, So the last 32 bits is information field.

Ex 3: The error case.

Condition: plain text: 01010011 11000000 01111100 00000000 00000000 00000000 00000000 00100000, the block is: 01010011 11000000 01111100 00000000 00000000 00000000 00000000 00100000.

Processing:

First, we get the last 8 bits of the block: 00100000. Then we presume the length of padding field is 32 bits.

Second, we get the last 32 bits data from the block.

Third, we find the 32-8=24(got rid of the length field) bits data is zero.

Forth, so the last 32 bits is padding field and we can delete it. We get error result.

If we use the new padding method to do the padding there will be some errors. But we can analyze it and we get a conclusion that it has less error than old method which be used in zero padding. Let us make a math proving.

If: The probability of every kind of block is $1/2^{64}$. The length of padding field is ”L”.

old: If the last ”L” bits are zero the original message may have $\sum_{i=1}^{L} 1/2^i$ possibilities. Then the error probability is

$$P_E = 1/2^{64} * \sum_{L=1}^{64} \sum_{i=1}^{L} 1/2^i \quad (1)$$

New: If the last 8 bits are zero the original message may have 2 possibilities. Then the error probability is

$$P_E = 1/2^{64} * \sum_{L=1}^{64} 1/2^L \quad (2)$$

So the new padding method is more exactly. Then the next section will give the implementation of DES and the new padding method is used in it.

3. Implementation

Implementation of the DES algorithm is given below. We use the java as the programming language. The new padding method is used in it. There are an interface for DES encryption, a class for block processing and a class for implementation of the DES. And we will introduce the definition of them below.

First the **Figure 1** show the relation between different class or interface. The test class is used in testing and we will not introduce it here. The user uses the interface IDES to do the encryption. And then the IDES can be implemented in the DES class. And then the method in DES class will call the method in block class to do detailed operation.

3.1. Interface IDES

1) encrypt(byte[] src,byte[] dest,DESKey key)

The method is used to encrypt the plain text with the DES key and get the cipher text.

2) decrypt(byte[] src,byte[] dest,DESKey key)

The method is used to decrypt the cipher text with the DES key and get the plain text.

3.2. Class Block

1) The private List(Bit) bits; Used to store the bits sequence. Returns the part(from the start position to the end position) of the block.

```
Block getBlockPart(int start, int end)
{
Block newblock = new Block(0);
newblock.bits
=this.bits.subList(start, end);
return newblock;
}
```

2) Set the bits with new value at the position appointed.

```
Block setValue(byte[] value, int offset, int count)
{
Block newblock = new Block(0);
for (int i = 0; i < count;i++)
{
Block curPart = new Block(0).setValue(8,value[i+offset]);
newblock.bits.addAll(curPart.bits);
}
return newblock;
}
```

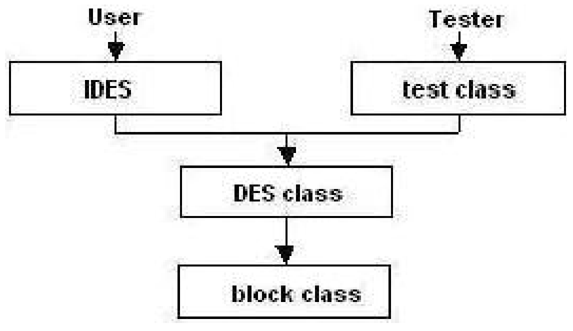

Figure 1. Main Frame of the Program.

3) Joins this and block by creating a new block with all bits of this followed by all bits of block.

```
Block joinBlock(Block block)
{
Block newblock = new Block(0);
newblock.bits.addAll(this.bits);
newblock.bits.addAll(block.bits);
return newblock;
}
```

4) Removes all bits larger than new length.

```
Block trim(int newlength)
{
Block newblock = new Block(0);
newblock.bits = this.bits.subList(0, newlength);
return newblock;
}
```

3.3. The Implementation of IDES

The class ”DES” implements the interface ”IDES”.

1) Implement the DES encryption.

```
int encrypt(byte[] src,byte[] dest,DESKey key)
{
get plain blocks.
while(not end){
encryptPadding(plain block);
for (int round=0; round<16;round++)
{encryptblock(plain block);}
}
return number of blocks.
}
```

2) The method is used to implement the Interface method.

```
int decrypt(byte[] src,byte[] dest,DESKey key)
{
get cipher blocks.
while(not end){
decryptPadding(cipher block);
for (int round=0; round<16;round++)
{decryptblock(cipher block);}
}
return number of blocks.
}
```

3) Pad the space in the block less than 64 bits.

```
Block encryptPadding(Block block)
{
if (block.getLength() == 64)
{return block; //no padding}
else if (block.getLength() == 0)
{//set length=64(0x40)
return new Block(0).setValue(64,64);
}
else
{//padding 64-block’s length bits
len=block.getLength();
pad=new Block(0).setValue(0,len,64-len)
return block.joinBlock(pad); }
}
```

4) Delete the appending bits in the block.

```
Block decryptPadding(Block block)
{
int padding= block.getBlockPart(56,64).getValue();
if (padding > 64)
{ //padding length must less than 64
throw new RuntimeException("Wrong");
}
return block.trim(64 - padding);
}
```

This is the main definition of the interface and the class. The detail of the implementation of the method is omitted. The test results will be given in the next section.

4. Conclusion

This paper is discussion of DES padding method. Although there are many problems in the new padding method we will focus on how to modify the DES algorithm to make it more effectively and more safely in future study.

5. Acknowledgements

The authors would like to thank the colleagues who give much help and also thank the students who make the testing carefully.

REFERENCES

[1] Federal Information, ”DATA ENCRYPTION STANDARD (DES)”, . http://www.itl.nist.gov/fipspubs/fip46-2.htm

[2] J. Orlin Grabbe, ”The DES Algorithm Illustrated”,. http://www.aci.net/kalliste/des.htm

[3] Alfred J. Menezes, Paul C. van Oorschot, Scott A. anstone, ”Handbook of Applied Cryptography”,. CRC Press,1996, pp250-259

[4] Brice Schneier, ”Applied Cryptography 2ND Edition: Protocols, Algorithms and Source Code in C”,. Now York: Wiley John and Sons,1996