**Décodage：**

1. 设置一个char型数组chainedecode[ ]，用来存放解密出的message的各字符。

为确定chainedecode[ ]的长度，先解密①CRC，②Taille du nom du fichier，③Taille de données du fichier。则chainedecode[ ]的长度为3+Taille du nom du fichier+Taille de données du fichier。

On définit un tableau (char) “chainedecode[]” pour stocker chaque caractère du message décodé. Afin de déterminer la taille de chainedecode[], on décode d'abord le “①CRC”, “②Taille du nom du fichier” et “③Taille de données du fichier”. Alors la taille de chainedecode[] est égale à “3+Taille du nom du fichier+Taille de données du fichier”.

1. ①将每个像素和0000 0001（0x01）进行与运算，提取像素的最后一位。将结果存放在二位数组newdata[i][j]中。

由加密的算法可知，依次每8个newdata[i][j]的元素可以合并成message的一个字符，即chainedecode[ ]的一个元素，定义变量k，指示chainedecode[ ]中的第k个元素。

②按每8列为一组，每一行被分成nb\_cara\_par\_ligne = (image->width)/8个组，每个组对应一个字符。

定义变量m和n，m表示每一行的第m组，n表示每一组的第n个元素。

③每一组的第n个元素左移7-n位。

④将每组的所有8个元素移位后的结果进行或运算，结果就是message的一个字符的ascii码。

①On effectue “AND” de chaque pixel (image->data[i][j]) avec 0000 0001 (0x01) pour retirer son dernier bit. Les résultats sont stockés dans un tableau à 2 dimension “newdata[i][j]”.

D’après l'algorithme de codage, nous pouvons voir que chaque 8 éléments de newdata[i][j] peut être combiné en un caractère du message, qui est un élément de chainedecode[]. On définit un variable “k” pour indiquer le kième élément dans le codechaîné[].

②Chaque 8 colonnes étant un groupe, chaque ligne est divisée en “nb\_cara\_par\_ligne = (image->width)/8” groupes, et chaque groupe correspond à un caractère.

On définit deux variables m et n. m indique le mème groupe de chaque ligne, et n représente le nième élément de chaque groupe.

③Le nième élément de chaque groupe décale à gauche de 7-n bits.

④On effecte “OR” du résultat du décalage de tous les 8 éléments de chaque groupe, ça donne un code ascii d’un caractère du message.

例：

i=0，m=0，n=0：image->data[0][0]对应的ascii码为0110 001**0**，

n=1：image->data[0][1]对应的ascii码为1110 010**1**，

n=2：image->data[0][2]对应的ascii码为1111 101**0**，

n=3：image->data[0][3]对应的ascii码为0110 110**0**，

n=4：image->data[0][4]对应的ascii码为1000 111**0**，

n=5：image->data[0][5]对应的ascii码为0101 000**0**，

n=6：image->data[0][6]对应的ascii码为1011 100**0**，

n=7：image->data[0][7]对应的ascii码为1010 000**1**，

①将每个像素和0x01与运算：n=0：newdata[0][0] = 0110 001**0** & 0000 0001 = 0000 000**0**，

n=1：newdata[0][1] = 1110 010**1** & 0000 0001 = 0000 000**1**，

n=2：newdata[0][2] = 1111 101**0** & 0000 0001 = 0000 000**0**，

n=3：newdata[0][3] = 0110 110**0** & 0000 0001 = 0000 000**0**，

n=4：newdata[0][4] = 1000 111**0** & 0000 0001 = 0000 000**0**，

n=5：newdata[0][5] = 0101 000**0** & 0000 0001 = 0000 000**0**，

n=6：newdata[0][6] = 1011 100**0** & 0000 0001 = 0000 000**0**，

n=7：newdata[0][7] = 1010 000**1** & 0000 0001 = 0000 000**1**，

②n=0：newdata[0][0]左移7-0=7位：0000 000**0** << 7 = **0**000 0000，

n=1：newdata[0][1]左移7-1=6位：0000 000**1** << 6 = 0**1**00 0000，

n=2：newdata[0][2]左移7-2=5位：0000 000**0** << 5 = 00**0**0 0000，

n=3：newdata[0][3]左移7-3=4位：0000 000**0** << 4 = 000**0** 0000，

n=4：newdata[0][4]左移7-4=3位：0000 000**0** << 3 = 0000 **0**000，

n=5：newdata[0][5]左移7-5=2位：0000 000**0** << 2 = 0000 0**0**00，

n=6：newdata[0][6]左移7-6=1位：0000 000**0** << 1 = 0000 00**0**0，

n=7：newdata[0][7]左移7-7=0位：0000 000**1** << 0 = 0000 000**1**，

③将②中的8个结果进行或运算：

(**0**000 0000) | (0**1**00 0000) | (00**0**0 0000) | (000**0** 0000) | (0000 **0**000) | (0000 0**0**00) |

(0000 00**0**0) | (0000 000**1)** = **0100 0001**

因此，chainedecode[0]的ascii码为0100 0001，则chainedecode[0]=’A’。

1. 根据chainedecode[ ]的长度3+Taille du nom du fichier+Taille de données du fichier可以算出需要进行解码的像素的个数。

我们按照“先行后列”的规则进行加密，因此解密也是行优先进行的。设每行的组的个数为nb\_cara\_par\_ligne，设置如下两个常量：

I=(3+taille\_nom+taille\_data)/nb\_cara\_par\_ligne;

M=(3+taille\_nom+taille\_data)%nb\_cara\_par\_ligne;

将解密分为两个部分进行：

（1）0≤i<I，0≤m<nb\_cara\_par\_ligne

1. i=I，0≤m<M

对于像素image->data[i][j]，列(j)与组(m)以及每组的元素标号(n)的关系为：j=8\*m+n

则像素和chainedecode[k]的对应关系为：k=nb\_cara\_par\_ligne\*i+m

Selon la taille de codechaîne[] (3+Taille du nom du fichier+Taille de données du fichier), le nombre de pixels à décoder peut être calculé.

Le codage suit la règle "Priorité de ligne", donc le décodage suivi aussi cette règle. Suppose que le nombre des groupes par ligne est nb\_cara\_par\_ligne, on définit les deux constantes suivantes:

I=(3+taille\_nom+taille\_data)/nb\_cara\_par\_ligne;

M=(3+taille\_nom+taille\_data)%nb\_cara\_par\_ligne;

Le décodage est divisé en deux parties:

（1）0≤i<I，0≤m<nb\_cara\_par\_ligne

（2）i=I，0≤m<M

Pour chaque pixel image->data[i][j], la relation entre l’indice de la colonne (j), du groupe (m) et du élément dans chaque groupe (n) est: j=8\*m+n

Donc, le relation entre le pixel et chainedecode[k] est: k=nb\_cara\_par\_ligne\*i+m

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i=0 | m=0 | | | | | | | | | |
| n=0 | n=1 | n=2 | n=3 | n=4 | n=5 | n=6 | n=7 | | |
| k=0=nb\_cara\_par\_ligne\*0+0 | | | | | | | | | | | |
|  | | | | | | | | | | | |
| i=0 | m=1 | | | | | | | | | |
| n=0 | n=1 | n=2 | n=3 | n=4 | n=5 | n=6 | n=7 | | |
| k=1=nb\_cara\_par\_ligne\*0+1 | | | | | | | | | | | |
| ... ... | | | | | | | | | | | |
| i=1 | m=0 | | | | | | | | | |
| n=0 | n=1 | n=2 | n=3 | n=4 | n=5 | n=6 | | | n=7 |
| k=nb\_cara\_par\_ligne\*1+0 | | | | | | | | | | | |
|  | | | | | | | | | | | |
| i=1 | m=1 | | | | | | | | | |
| n=0 | n=1 | n=2 | n=3 | n=4 | n=5 | n=6 | | n=7 | |
| k=nb\_cara\_par\_ligne\*1+1 | | | | | | | | | | | |

代码：

for(i=0;i<I;i++)

{

for(m=0;m<nb\_cara\_par\_ligne;m++)

{

for(n=0;n<8;n++)

{

newdata[i][8\*m+n]=newdata[i][8\*m+n]<<(7-n); chainedecode[nb\_cara\_par\_ligne\*i+m]=chainedecode[nb\_cara\_par\_ligne\*i+m]|

newdata[i][8\*m+n];

}

}

}

for(m=0;m<M;m++)

{

for(n=0;n<8;n++)

{

newdata[I][8\*m+n]=newdata[I][8\*m+n]<<(7-n);

chainedecode[nb\_cara\_par\_ligne\*I+m]=chainedecode[nb\_cara\_par\_ligne\*I+m]|

newdata[I][8\*m+n];

}

}

