Flash Decryption

From Archive.org

What is the flash image?

Inside the firmware there are typicaly between 2 to 3 images.
1. The Apple flash update image.
2. The Apple OS.
3. The resources image (only from 5G, nano and above.)
For more info visit Firmware page.

While images number 2 and 3 are raw, the Apple flash update image is encrypted.
Note: On iPods 1G-3G this image is not encrypted.
Late discoveries by BadBlox find out how to decrypt this image.
I will describe how it's done here. In addition, in the end of this page you will find the source code in C++ and Java for how to do it.

Decryption

Apple decided to encrypt the flash image in RC4.
Luckily for us, the key is hidden in the firmware itself.
An image consists of a 512 (might be 2048, depends on the disk sector size, for me details, read about Volume offset in the firmware directory in Firmware page) random bytes and after that the data itself.
According to Firmware page, the checksum starts not at the real offset written in the image directory table but after these 512 bytes.
These 512 bytes are called the security block.
They hold a 32bit key for the decryption of the data of the image.
The security block contains 8 "markers". These markers can be enable or disable.
If all the markers are disabled, then the image is unprotected.
If one marker is enable then the image is protected and you have to extract the key in order to decrypt it.

The markers are 32 bits values at precises locations in the block. Here is the word offset for the 8 markers in the security block:

int[] offset={0x5,0x25,0x6f,0x69,0x15,0x4d,0x40,0x34};

To get the actual offset in bytes in the block, you have to multiply by 4.

To know more about markers properties and RC4 key extraction, please refer to the source code example.

After we got the key, we just use an RC4 decryptor to do the work of decrypting the image.
NOTE: The key is big endian and you need to flip it in order for the decryption to work.

Source Code

Now for the source code.
BadBlox and Kingstone made a source code for this one.

BadBlox's Java code:

```
package Ipod.Firmware;

public class SecurityBlock {

        byte[] data;
        private int[] offset={0x5,0x25,0x6f,0x69,0x15,0x4d,0x40,0x34};
        public int key;
        public boolean fileIsProtected=false;

        public SecurityBlock(byte[] rawData){
                int constant = 0x54c3a298;
                int key=0;
                data=rawData;
                int aMarker=0;
                int pos=0;
                for (int c=0;c<8;c++){
                        pos =offset[c]*4;
                        aMarker=readWord(rawData,pos);
                        boolean result=testMarker(aMarker);
                        //System.out.println("Marker ="+Integer.toHexString(aMarker)+" "+result);
                        if (result){ // This marker is enable
                                fileIsProtected=true;
                                // pos of nextblock
                                pos =(offset[c+1]*4)+4;
                                key=0;
                                int temp1=aMarker;
                                for (int count=0;count<2;count++){
                                        int word=readWord(data,pos);
                                        temp1=aMarker;
                                        temp1=temp1^word;
                                        temp1=temp1^constant;
                                        key=temp1;
                                        pos=pos+4;
                                }
                                int r1=0x6f;
                                int r2=0;
                                int r12;
                                int r14;
                                for (int count=2;count<128;count=count+2){
                                        r2=readWord(data,count*4);
                                        r12=readWord(data,(count*4)+4);
                                        r14=r2 | (r12>>>16);
                                        r2=r2&0xffff;
                                        r2=r2 | r12;
                                        r1=r1^r14;
                                        r1=r1+r2;
                                }
                                key=key^r1;
                                // Invert key, little endian
                                this.key = ((key&0xff)<<24)|((key&0xff00)<<8)|((key&0xff0000)>>>8)|((key&0xff000000)>>>24);
                        }
                }
        }

        public static int readWord(byte[] buffer,int pos){
                int p1=buffer[pos];
                int p2=buffer[pos+1];
                int p3=buffer[pos+2];
                int p4=buffer[pos+3];
                if (p1<0)p1=p1+256;
                if (p2<0)p2=p2+256;
                if (p3<0)p3=p3+256;
                if (p4<0)p4=p4+256;
                return p1+(p2<<8)+(p3<<16)+(p4<<24);
        }

        public boolean testMarker(int marker){
                int mask = (marker&0xff)|((marker&0xff)<<8)|((marker&0xff)<<16)|((marker&0xff)<<24);
                int decrypt = marker ^ mask;
                int temp1=decrypt>>>24;
                int temp2=decrypt<<8;
                if (temp1==0) return false;
                temp2=temp2>>>24;
                decrypt=decrypt<<16;
                decrypt=decrypt>>>24;
                if ((temp1<temp2)&&(temp2<decrypt)){
                        temp1=temp1&0xf;
                        temp2=temp2&0xf;
                        decrypt=decrypt&0xf;
                        if ((temp1>temp2)&&(temp2>decrypt)){
                                if (decrypt!=0) return true; // This marker is enable!
                        }
                }
                return false;
        }
}
```

Kingstone's C++ code:

AUPD Decrypter (Dead Link)

Your input file has to be a firmware file. The program will find the aupd image and decrypt it.

It saves two files: A complete firmware file with decrypted aupd image and a single decrypted aupd image.

Retrieved from "http://ipodlinux.org/wiki/Flash_Decryption (2010)"