

Protecting Cipher Programs using Light-Weight Obfuscations: A Cracker-Centric Approach

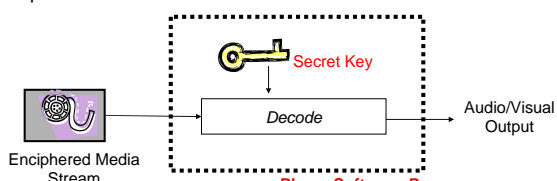
The 5th COE Postdoctoral and Doctoral Researchers Technical Presentation
August 25, 2005

Hiroki Yamauchi

Software Engineering Laboratory,
Graduate school of Information Science,
Nara Institute of Science and Technology

August 25, 2005 1

Background

- Nowadays, Digital Rights Management (DRM) software, such as iTunes and media player, became an important application for ubiquitous computing environments.
- Simplified DRM software:
 
- The manufacturer of DRM software must protect secret keys included in the player software from being spied out by a software cracker.

August 25, 2005 2

Software Obfuscation

- Various software obfuscation methods have been proposed to make it difficult for crackers to understand the software.
- Program obfuscation
 - Making expressions and procedures in a program more complex than the original[1].
- Example of program obfuscation

```
int a = 2, b = 3;
for(i=0; i<5; i++){
  a *= pow(b, i);
}
printf( "%d", a);
```

➔

```
int a[] = {2,11};
int i=1;
LOOP:
if(i!=5){a[0] *=
pow((a[1]-2)/3, i);
i++;goto LOOP;
}printf( "%d", a[0]);
```

[1] C. Collberg, C. Thomberson, "Watermarking, Tamper-Proofing, and Obfuscation -- Tools for Software Protection," IEEE Trans. Software Eng., vol.28, no.8, pp.735-746, June 2002.

August 25, 2005 3

Problem of software protection

- Unfortunately, it is unclear how effective they are in protecting a secret key inside the cipher program.
 - Difficult to understand Difficult to find a secret key.
- We focus on the "Crackers' viewpoint".
 - Obfuscation method often fell into a protector-centric approach focusing on building a "complex" program.
 - Our approach is a cracker-centric to conceal the clues of the attack.
 - It is necessary to construct a realistic crackers' model, a model of what the cracker can do (and cannot do).

August 25, 2005 4

Approach and Goals

- Goals
 - To develop the guideline for applying obfuscation methods to protect cipher programs against a skilled cracker who tries to extract security-sensitive data.
 - Our research is NOT a proposal of new obfuscation methods. Our aim is it clarify where and how to apply existing obfuscation methods.
- Approach
 - Define a realistic crackers' model.
 - Develop a guideline for obfuscation to hide clues, which might be found by the cracker.

August 25, 2005 5

Target Software

- A data decryption program using C2 algorithm
 - C2(Cryptomeria Cipher) :
 - A Feistel network-based block cipher designed for use in the area of digital entertainment content protection (CPPM/CPRM).
 - The algorithm is open to the public[2]
 - The Device Key is required to be hidden from users.
 - Written in Java
 - Using ECB (Electronic Code Book) mode

[2] 4C Entity, "Content protection for recordable media specification Introduction and common cryptographic elements," rev. 1.0, 31 pp.1.0, Jan. 2003.

August 25, 2005 6

Crackers' model

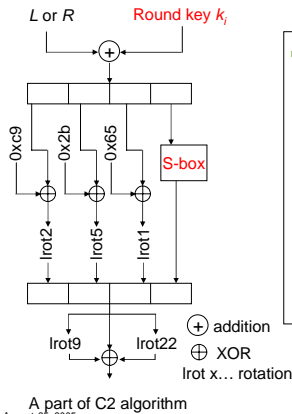
We characterized the cracker's knowledge and resources along with three dimensions.

- **Algorithm Understanding**
The cracker has full knowledge of the principles of C2 algorithm.
- **System Observation**
The cracker owns a binary file and a disassembled file of DRM software.
The cracker can observe computer memory and execution trace of software using debuggers.
- **System Control**
The cracker can execute the software with an arbitrary input.
The cracker can modify the software in any desired way.

August 25, 2005

7

Algorithm understanding



August 25, 2005

8

- **Cracker's knowledge**
Round keys are 32-bit length constants.
There are 10 round keys.
S-box is a set of 256 8-bit values.
...

Guideline for applying obfuscation

- In order to achieve the security goal, we need to hide the following information in the C2 program.
 - A) Round keys (secret keys)
 - B) S-box (secret data table)
 - C) Feistel function
 - D) Distinctive opcodes and operands
 - E) Obfuscation itself
- Currently, we started to develop a guideline of obfuscation that can hide above information.
- Today, I explain about hiding A) and B).

August 25, 2005

9

A) Round keys

```
rkey[0] = 0x789ac6ee;
rkey[1] = 0x79bc3398;
rkey[2] = 0x48d15d62;
....
```

Source code for defining Round keys

```
17:  iconst_0
18:  ldc   #6; //int 2023409390
20:  iastore
21:  aload  8
23:  iconst_1
24:  ldc   #7; //int 2042377112
26:  iastore
27:  aload  8
29:  iconst_2
30:  ldc   #8; //int 1221680482
```

Disassembled code (by javap -c)

Round keys can be easily found from disassembly code.

Because they are large (32-bit) constants

August 25, 2005

10

Obfuscation for Hiding Round keys

- Changing the encoding domain of keys with homomorphism
 $rkey[0] = 0x789ac6ee;$ → $rkey[0] = 0x123232143e;$
 Calculations using the key must be done in the changed domain.
- Inserting dummy keys
 $rkey[0] = 0x789ac6ee;$
 $rkey[1] = 0x79bc3398;$
 ... → $rkey[0] = 0x789ac6ee;$
 $rkey[1] = 0x6bc32e55;$ ← dummy constant
 $rkey[2] = 0x79bc3398;$
 $rkey[3] = 0x1d8aa4f5;$ ← dummy constant
 ...
- Dividing keys into smaller sub-keys.
 $rkey[0] = 0x789ac6ee;$ → $B_rkey[0][0] = 0x78;$
 $B_rkey[0][1] = 0x9a;$
 $B_rkey[0][2] = 0x6c;$
 $B_rkey[0][3] = 0xee;$
 Sub-keys must not be concatenated back to the original key throughout the calculation.

August 25, 2005

11

B) S-box

```
static byte SecretConstant[] = { (byte)0xB6, (byte)0xAA, ...
```

Source code for defining S-box

```
static {};
Code:
0:  sipush 256
3:  newarray byte
5:  dup
6:  iconst_0
7:  bipush -74
9:  bastore
10: dup
11: iconst_1
12: bipush -86
14: bastore
```

Disassembled code (by javap -c)

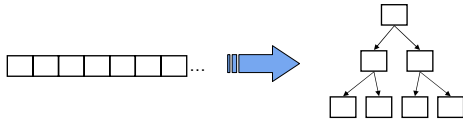
Arrays are easily recognizable in disassembled code.

August 25, 2005

12

Obfuscation for hiding S-Box

- Using more complex data structure, such as trees and lists, instead of using a simple array.



- Merge, Split, Fold and/or Interleave the array.

```
(1) int A[10]
(2) A[i] = ...;
```

```
(3) int B[10], C[20];
(4) B[i] = ...;
(5) C[i] = ...;
```



```
(1) int A1[5], A2[5];
(2) if((i%2)==0) A1[i/2]=...;
    else A2[i/2]=...;
(3) int BC[20];
(4) BC[3*i] = ...;
(5) BC[i/2*3+1+i%2] = ...;
```

August 25, 2005

13

Summary

- Defined a cracker's model, a model of what a cracker is able (and not able) to do.
- Developed a part of an obfuscation guideline to hide clues, which might be found by the cracker.
 - Hiding round keys and S-Box
- Future Plan
 - Complete the guideline.
 - Evaluation by a super hacker.

August 25, 2005

14