ソフトウェア保護技術

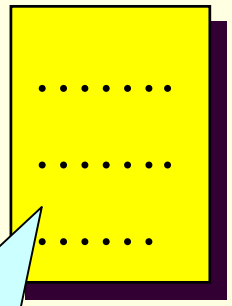
2004年12月8日

門田暁人

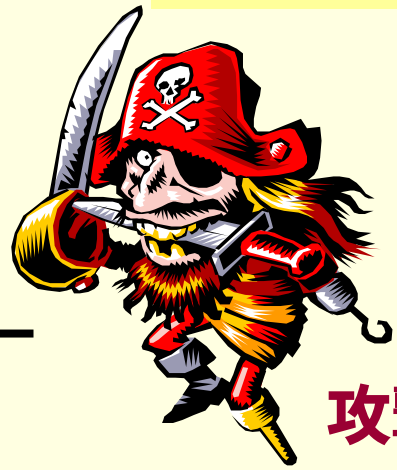
奈良先端科学技術大学院大学
情報科学研究科

ソフトウェア保護の目的

- ソフトウェア内部の秘密を隠す.
- ソフトウェアの改変を防ぐ.



解析

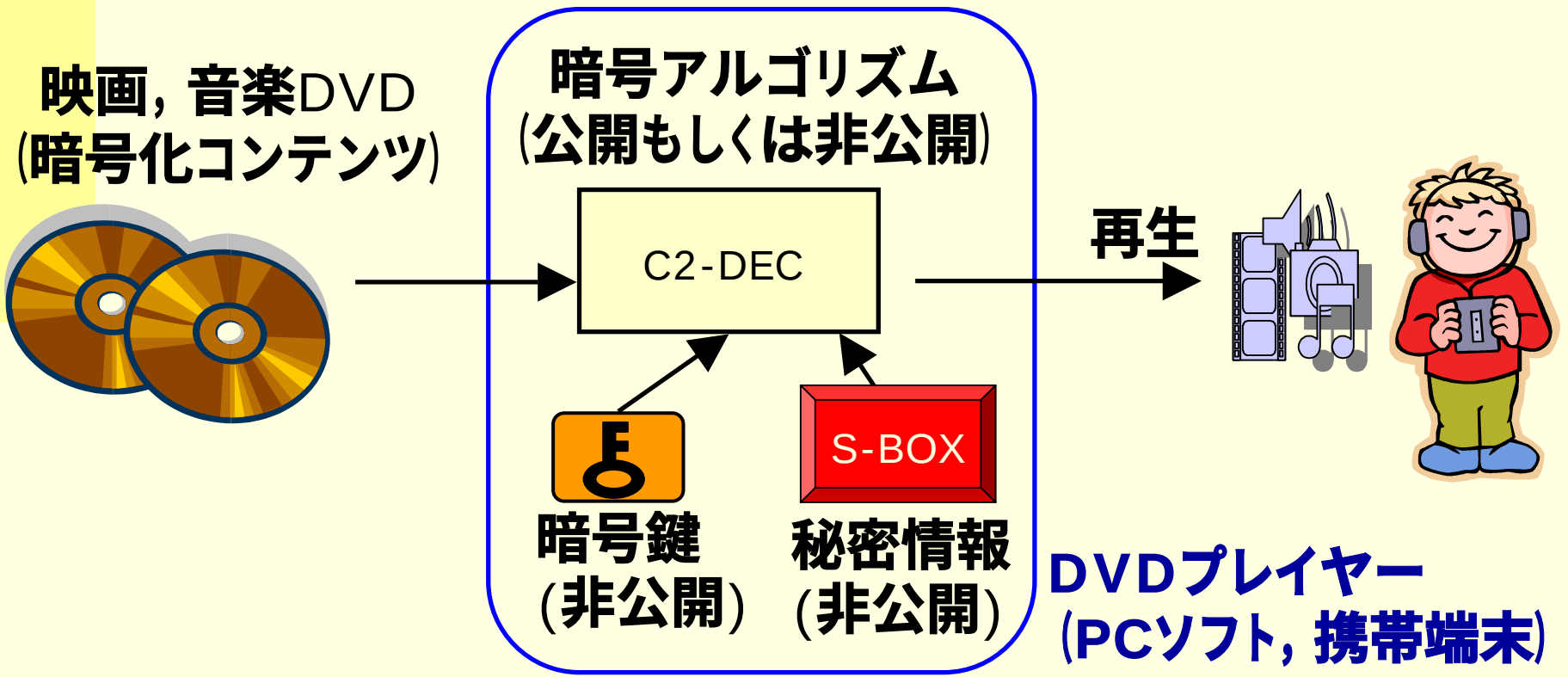


攻撃者

- **アルゴリズムやモジュール**: デジタルコンテンツの著作権管理(DRM)システムのアルゴリズムなど
- **データ (定数)** : 暗号鍵や携帯電話のデバイスIDなど
- **条件判定式**: ライセンスチェックのための分岐文
  “if (licensed) goto L”.
- **外部インタフェース**: メンテナンス用の隠しモード

保護すべきソフトウェアの例

- DVDプレイヤーソフト
  - 鍵や秘密情報がプレイヤーソフトに埋め込まれている．
  - 暗号が破られると生コンテンツがインターネットに流出する恐れがある．(P2Pソフトによる流通)



ソフトウェア保護技術

- ソフトウェア単体での保護
  - 難読化
  - 暗号化
  - 自己書き換え
- ハードウェア支援による保護
  - プログラムのASIC化 (Application Specific Integrated Circuit)
  - ワンチップ化 (CPU+ROM)，ドングル方式
  - バス暗号方式
  - IBM, Intel, Microsoft等によるTrusted Computing構想[1]

[1] The Trusted Computing Group, “TPM Main, Part 1: Design Principles,” Specification Version 1.2, Oct. 2003.

プログラムの難読化

```
int n = 52;
int i, k, p=1;

for(i=1;i<=31;i++)
{
   k = n – i + 1;
   p = p * k / i;
}
return p;
```

等価変換

```
int n=105,k,i=1,p=1;
L1: if(i <= 31){ for(;;){
k=n–2*i+2;p=(p*k-p)/2/i;
if(++i>31){k=n–2*i+2;
p=(p*k-p)/2/i++; }else
break;
p=p*(n–2*i+1)/2/i++;}
goto L1;}
return p;
```

元のプログラム

難読化したプログラム

・プログラムの制御構造やデータ構造を複雑にする．
- 仕様を変えない．
- 実行効率を落とさない．

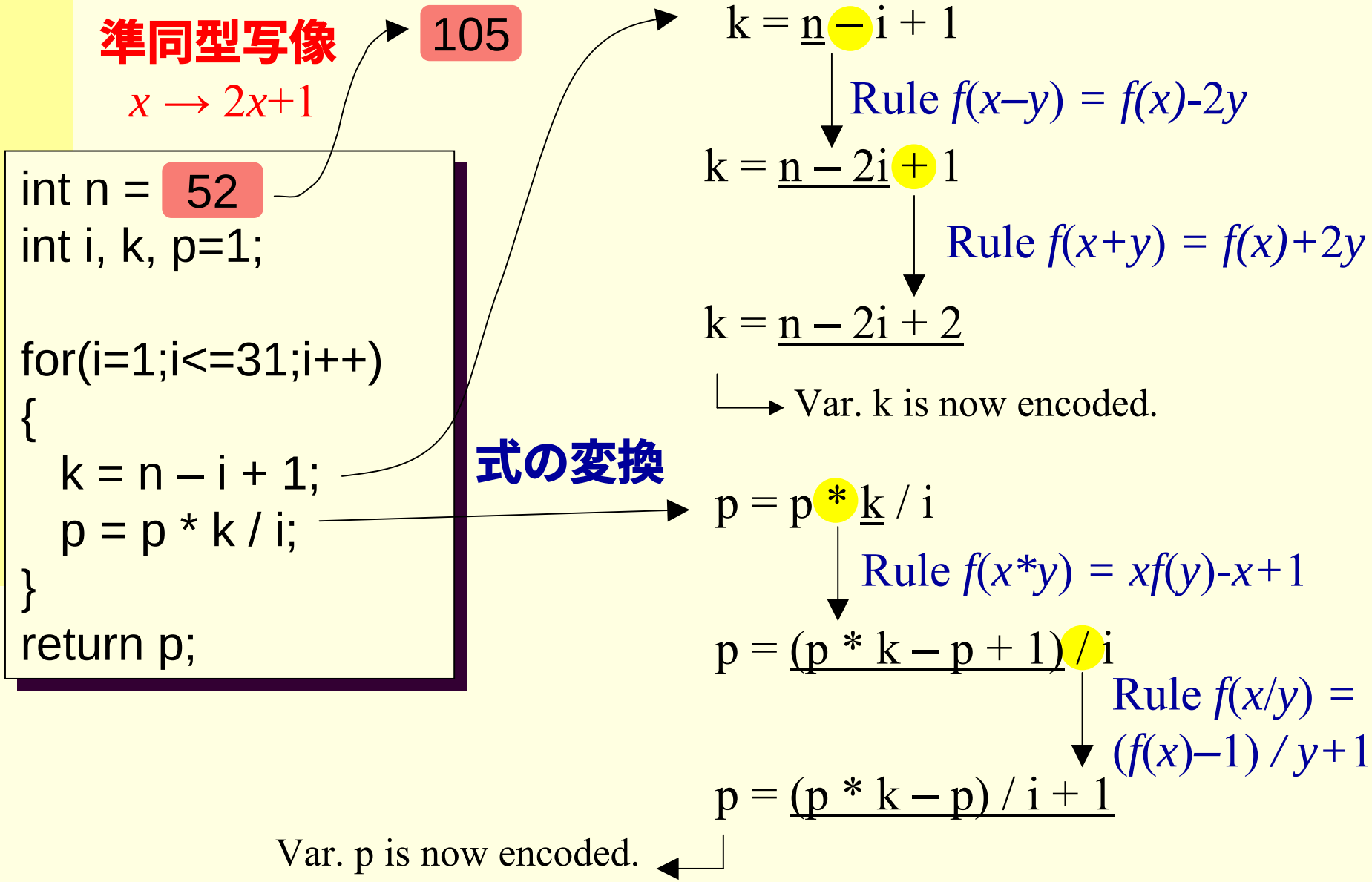
プログラムの難読化の技術動向（特許・商品）

- **変数名（識別子）の難読化**
  - **数多くの製品**（Java, .NET**用**） PreEmptive**社が有名**
- **フローチャートの難読化**
  - **数多くの論文**, Cloakware**社の特許**[1], **商品化されていない**.
- **データの難読化**
  - **奈良先端大の特許**[2], Cloakware**社の特許＆サービス**[3],
- **その他**:InterTrust[4], PreEmptive[5], Apple Computer[6]**など**

データの難読化は今後有望である.
プログラム中の暗号鍵や重要なデータを隠すのに役立つ.

[1] *United States Patent* 6,779,114, Cloakware, Aug. 2004.
[2] **門田**, **佐藤**, **神崎**, "**プログラム難読化装置，プログラム難読化プログラム及びプログラム難読化方法**," **特願**2003-202795, 29July 2003.
[3] *United States Patent* 6,594,761, Cloakware, July 2003.
[4] *United Sates Patent* 6,668,325, InterTrust, Dec. 2003.
[5] *United States Patent* 6,102,966, PreEmptive Solutions, Aug. 2000.
[6] *United States Patent* 6,694,435, Apple Computer, 17 Feb. 2004.

データの難読化の例


準同型写像
$x \rightarrow 2x+1$
105
int n = 52
int i, k, p=1;
for(i=1;i<=31;i++)
{
k = n – i + 1;
p = p * k / i;
}
return p;
式の変換
k = n – i + 1
Rule $f(x–y) = f(x)-2y$
k = n – 2i + 1
Rule $f(x+y) = f(x)+2y$
k = n – 2i + 2
Var. k is now encoded.
p = p * k / i
Rule $f(x*y) = xf(y)-x+1$
p = (p * k – p + 1) / i
Rule $f(x/y) =$
$(f(x)–1) / y+1$
p = (p * k – p) / i + 1
Var. p is now encoded.

プログラムの暗号化

```
int n = 52;
int i, k, p=1;

for(i=1;i<=31;i++)
{
   k = n – i + 1;
   p = p * k / i;
}
return p;
```

元のプログラム

暗号化

```
0J9F0E7FK3JADJEIOJ3945
7ASJSI438JCI4AWIDJO45F
OLPO9HI7FF8LE43D2LIJ5J
SIO2119JASHKBMSJDU921
AKNNS33KMNZOF7H0IJAG
8KLPQA6HDMVU4JFDIOU9
```

[復号ルーチン]

暗号化後のプログラム

- **プログラムの全体または一部を暗号化する.**
- **復号ルーチンを追加する.**
- **プログラム実行時に復号処理を行う.**

プログラムの暗号化の技術動向

- 数多くの商品がある[1][2] .
- 攻撃に弱い.
  - 復号ルーチンが解析される.
  - デバッガによるメモリの覗き見, メモリダンプ
- 攻撃方法が知られている.
  - 攻撃方法を解説する本 [3]
  - 攻撃方法を解説するWebページ [4]

商品化もされており, よく用いられているが, 攻撃方法もまたよく知られている. 難読化と併用するのがよい.

[1] ASPack Software, “ASProtect,” http://www.aspack.com/
[2] Obsidium Software, “Obsidium” http://www.obsidium.de/
[3] Kracker’s & BEAMZ, “クラッカー・プログラム大全,” データハウス, 2003.
[4] “セキュリティアカデメイア – セキュリティ講座 – Krack”
http://akademeia.info/main/security_lecture.htm#krack

プログラムの自己書き換え

```
int n = 52;
int i, k, p=1;

for(i=1;i<=31;i++)
{
   k = n – i + 1;
   p = p * k / i;
}
return p;
```

元のプログラム

変換

```
:
cmpl    $123,-4(%ebp)
addl    %edx, %eax
movb    0x03, L15
ret
pushl   %ebp
addl    (%esp), %eax
xorl    $12, %esp
movb    0x06, L6
shrl    $8, %eax
subl    $2
andl    $1
addl    (%
inc     %eax
pushl   %ebp
cmpl    $123,-4(%ebp)
addl    %edx, %eax
movl    %eax, %ebx
movb    0x03, L15
pushl   %ebp
addl    (%esp), %eax
movb    0x03, L13
```

書き換え

書き換え

書き換え

書き換え

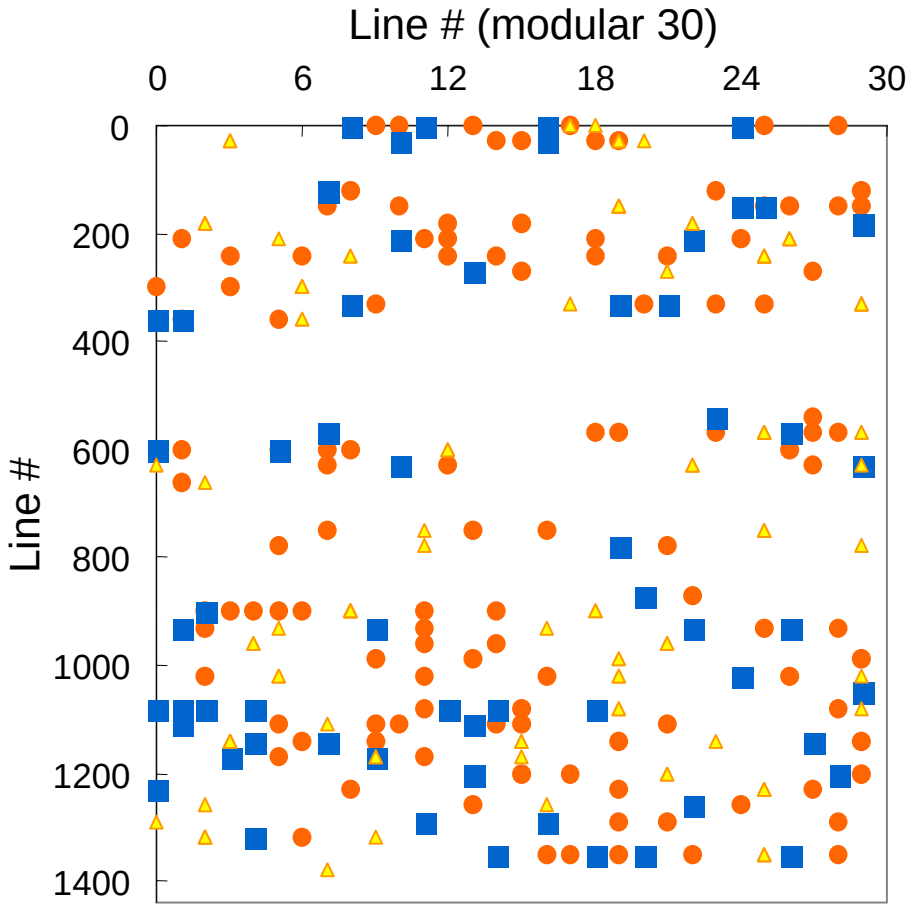
自己書き換えプログラム

- 自分自身を常に書き換えながら動作するプログラム[1]
  - 命令文，データを隠蔽できる

[1]門田, 神崎, "自己書き換え処理追加プログラム，自己書き換え処理追加装置及び自己書き換え処理追加方法," 特願2002-355881, Dec. 2002.

自己書き換えプログラムの例

偽の命令で上書きしておき，実行時に元来の命令に戻す．



対象アセンブリプログラム

- **全体の命令数** : 1000
- **偽の命令数**: 130

- 偽の命令
- 自己書き換えルーチン (復帰)
- 自己書き換えルーチン (隠蔽)

攻撃は極めて困難である.

試作版のシステムを公開中：http://se.naist.jp/rinrun/

まとめ

- プログラムを保護する必要性が高まっている.
- プログラムの暗号化が主流だが, 攻撃方法が確立されている.
- プログラムの難読化は, 近年, 特許が急激に増えている.
  - データの難読化が有望
- プログラムの自己書き換えは「古くて新しい」方式.
  - プログラム全体を暗号化するよりも安全 (攻撃に強い) .

- その他の技術:
  - アンチデバッガ, アンチ逆アセンブル, 改ざんの検出, NGSCB
  - ソフトウェア電子透かし[1] – サインを埋め込む
  - バースマーク[2] – プログラム固有の特徴量

[1] jmark http://se.naist.jp/jmark/
[2] jbirth http://se.naist.jp/jbirth/