

(19) 日本国特許庁(JP)

(12) 公開特許公報(A)

(11) 特許出願公開番号

特開2012-118956

(P2012-118956A)

(43) 公開日 平成24年6月21日(2012.6.21)

|                   |                  |      |      |      |  |             |
|-------------------|------------------|------|------|------|--|-------------|
| (51) Int.Cl.      |                  | F I  |      |      |  | テーマコード (参考) |
| <b>G06F 21/22</b> | <b>(2006.01)</b> | G06F | 9/06 | 660L |  | 5B276       |
| <b>G09C 1/00</b>  | <b>(2006.01)</b> | G09C | 1/00 | 660D |  | 5J104       |
| <b>H04L 9/14</b>  | <b>(2006.01)</b> | H04L | 9/00 | 641  |  |             |

審査請求 有 請求項の数 16 O L (全 19 頁)

(21) 出願番号 特願2011-11927 (P2011-11927)  
 (22) 出願日 平成23年1月24日 (2011.1.24)  
 (31) 優先権主張番号 10-2010-0122719  
 (32) 優先日 平成22年12月3日 (2010.12.3)  
 (33) 優先権主張国 韓国 (KR)

(71) 出願人 510105938  
 ザ インダストリー アンド アカデミッ  
 ク コオペレーション イン チュンナム  
 ナショナル ユニバーシティ (アイエー  
 シー)  
 THE INDUSTRY & ACAD  
 EMIC COOPERATION IN  
 CHUNGNAM NATIONAL  
 UNIVERSITY (IAC)  
 大韓民国, 305-764, テジョン, ユ  
 ソン-グ, クン-ドン, 220  
 (74) 代理人 100071054  
 弁理士 木村 高久

最終頁に続く

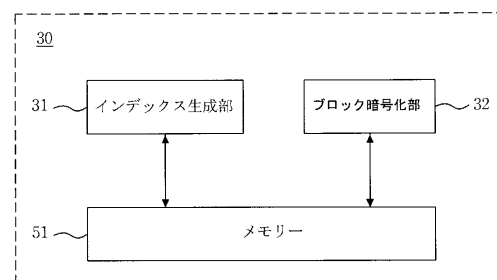
(54) 【発明の名称】 インデックステーブル基盤のコード暗号化及び復号化装置及びその方法

(57) 【要約】 (修正有)

【課題】 安全なコード暗号化を可能とするだけでなく、暗号化または復号化の実行時間を短縮して保存するデータの量を減らす。

【解決手段】 暗号化装置30は、実行ファイルの実行コードを呼出コードによりコードブロックで区分し、各コードブロックが呼び出される回数及び開始アドレスをインデックステーブルに保存するインデックス生成部31と、コードブロックを暗号キーで暗号化するが、1回呼び出されるコードブロック(以下、第1タイプのコードブロック)の暗号キーは第1タイプのコードブロックを呼び出すコードブロック(呼出ブロック)を利用して生成し、2回以上呼び出されるコードブロック(第2タイプのコードブロック)の暗号キーは乱数で生成して実行ファイルに保存するブロック暗号化部32と、を含む。

【選択図】 図12



**【特許請求の範囲】****【請求項 1】**

コンピュータプログラムの実行ファイルを暗号化するインデックステーブル基盤のコード暗号化装置であって、

前記実行ファイルの実行コードを呼出コードによりコードブロックで区分し、各コードブロックが呼び出される回数及び開始アドレスをインデックステーブルに保存するインデックス生成部と、

前記コードブロックを暗号キーで暗号化するが、1 回呼び出されるコードブロック(以下、第 1 種類のコードブロック)の暗号キーは前記第 1 種類のコードブロックを呼び出すコードブロック(以下、呼出ブロック)を利用して生成し、2 回以上呼び出されるコードブロック(以下、第 2 種類のコードブロック)の暗号キーは乱数で生成して前記実行ファイルに保存するブロック暗号化部と、を含むこと

10

を特徴とするインデックステーブル基盤のコード暗号化装置。

**【請求項 2】**

前記実行コードはバイナリコードまたはアセンブリコードであり、前記呼出コードはバイナリコードまたはアセンブリコードの分岐コードまたはジャンプコードであることを特徴とする請求項 1 に記載のインデックステーブル基盤のコード暗号化装置。

**【請求項 3】**

前記インデックス生成部は、最後の実行コードを呼出コードで含む一連の実行コードを一つのコードブロックで区分し、前記呼出コードは前記呼出コードが含まれたコードブロック以外の他のコードブロックを呼び出すことを特徴とする請求項 1 に記載のインデックステーブル基盤のコード暗号化装置。

20

**【請求項 4】**

前記インデックス生成部は、コードブロックの呼出回数をインデックステーブルに保存するが、各コードブロックの呼出コード各々に対して、前記呼出コードが呼び出すコードブロックの呼出回数を一つずつ増加させて保存することを特徴とする請求項 1 に記載のインデックステーブル基盤のコード暗号化装置。

**【請求項 5】**

前記ブロック暗号化部は、前記第 1 種類のコードブロックの呼出ブロックをハッシュして前記第 1 種類のコードブロックの暗号キーを生成することを特徴とする請求項 1 に記載のインデックステーブル基盤のコード暗号化装置。

30

**【請求項 6】**

前記ブロック暗号化部は、最初のコードブロックの暗号キーを初期キーで決めることを特徴とする請求項 1 に記載のインデックステーブル基盤のコード暗号化装置。

**【請求項 7】**

前記ブロック暗号化部は、前記第 2 種類のコードブロックの暗号キーを初期キーで暗号化して前記実行ファイルのデータ領域に保存することを特徴とする請求項 1 に記載のインデックステーブル基盤のコード暗号化装置。

**【請求項 8】**

前記ブロック暗号化部は、前記インデックステーブルを前記実行ファイルのデータ領域に保存することを特徴とする請求項 1 に記載のインデックステーブル基盤のコード暗号化装置。

40

**【請求項 9】**

前記インデックス生成部は、前記コードブロックのサイズを前記インデックステーブルに保存することを特徴とする請求項 1 に記載のインデックステーブル基盤のコード暗号化装置。

**【請求項 10】**

請求項 1 乃至請求項 9 の中でいずれの 1 項の装置により暗号化された実行ファイルを復号化するインデックステーブル基盤のコード復号化装置であって、

前記実行ファイルの暗号化された実行コードをコードブロック単位でコードブロックの

50

暗号キーを利用して復号化するが、前記インデックステーブルを参照して、前記コードブロックが第1種類のコードブロックであれば、前記コードブロックの呼出ブロックで暗号キーを生成して利用し、第2種類のコードブロックであれば、前記コードブロックの保存された暗号キーを利用するブロック復号化部と、

復号化されたコードブロックの最後の呼出コードが実行されると、前記コードブロックを再暗号化するブロック再暗号部と、を含むこと

を特徴とするインデックステーブル基盤のコード復号化装置。

【請求項11】

前記ブロック再暗号部は、前記コードブロックが反復実行される場合、反復回数を前記インデックステーブルに保存して前記コードブロックが呼び出される時ごとに前記反復回数を減少させて、反復回数が“0”になれば、前記コードブロックを再暗号化すること

10

を特徴とする請求項10に記載のインデックステーブル基盤のコード復号化装置。

【請求項12】

コンピュータプログラムの実行ファイルを暗号化するインデックステーブル基盤のコード暗号化方法であって、

(b)前記実行ファイルの実行コードを呼出コードによりコードブロックで区分し、各コードブロックが呼び出される回数及び開始アドレスをインデックステーブルに保存し、2回以上呼び出されるコードブロック(以下、第2種類のコードブロック)に対して暗号キーを乱数で生成して保存するステップと、

(c)前記コードブロックを前記コードブロックの暗号キーで暗号化するが、1回呼び出されるコードブロック(以下、第1種類のコードブロック)の暗号キーは前記第1種類のコードブロックを呼び出すコードブロック(以下、呼出ブロック)を利用して生成するステップと、を含むこと

20

を特徴とするインデックステーブル基盤のコード暗号化方法。

【請求項13】

前記実行コードはアセンブリコードであり、前記呼出コードはアセンブリコードの分岐コードまたはジャンプコードであることを特徴とする請求項12に記載のインデックステーブル基盤のコード暗号化方法。

【請求項14】

前記ステップ(b)は、

30

(b1)前記実行コードを順次に検査するステップと、

(b2)検査した実行コードが呼出コードであれば、呼出コードの被演算子からアドレスを抽出するステップと、

(b3)前記アドレスが前記インデックステーブルに存在しなければ、前記アドレスを前記インデックステーブルにコードブロックの開始アドレスで挿入するステップと、

(b4)前記アドレスが前記インデックステーブルに存在すれば、前記アドレスに該当するコードブロックの呼出回数を一つずつ増加させるステップと、

(b5)前記コードブロックの呼出回数が1回以上であれば、前記コードブロックの暗号キーを乱数で生成して保存するステップと、を含むこと

を特徴とする請求項12に記載のインデックステーブル基盤のコード暗号化方法。

40

【請求項15】

前記ステップ(b)は、

(b6)前記実行ファイルの最後の実行コードを検査すれば、各コードブロックのサイズを計算して前記インデックステーブルに保存するステップをさらに含むことを特徴とする請求項12に記載のインデックステーブル基盤のコード暗号化方法。

【請求項16】

請求項12乃至請求項15の中でいずれの1項の方法により暗号化された実行ファイルを復号化するインデックステーブル基盤のコード復号化方法であって、

(d)前記実行ファイルの暗号化された実行コードをコードブロック単位でコードブロックの暗号キーを利用して復号化するが、前記インデックステーブルを参照して、前記コー

50

ドブロックが第1種類のコードブロックであれば、前記コードブロックの呼出ブロックで暗号キーを生成して利用し、第2種類のコードブロックであれば、前記コードブロックの保存された暗号キーを利用するステップと、

(f)復号化されたコードブロックの最後の実行コードが実行されると、前記コードブロックを再暗号化するステップと、を含むこと

を特徴とするインデックステーブル基盤のコード復号化方法。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、コンピュータプログラムの実行ファイルを暗号化するか、暗号化された実行ファイルを復号化するインデックステーブル基盤のコード暗号化及び復号化装置及びその方法に関する。

10

【0002】

特に、本発明は、1回呼び出されるコードブロックは自分を呼び出すコードブロックを利用して暗号/復号化し、2以上呼び出されるコードブロックは乱数で生成した暗号キーで暗号/復号化するインデックステーブル基盤のコード暗号化及び復号化装置及びその方法に関する。

【0003】

また、本発明は、コードブロックの開始アドレス、呼出回数、ブロックのサイズなどを保存するインデックステーブルを利用して実行コードをブロックで区分するか呼出回数を計算するインデックステーブル基盤のコード暗号化及び復号化装置及びその方法に関するものである。

20

【背景技術】

【0004】

一般的に、ソフトウェアの著作権侵害による被害は相当なことで推算されている。特に、逆工学が広く知られた以後、このような著作権侵害は非常に深刻である。これはソフトウェアの特性上、一度配布された以後には逆工学により攻撃者に完全に露出されるからである。

【0005】

したがって、ソフトウェアを保護するためには逆工学に備えた技術が必要である。コード暗号化スキーム(`code encryption scheme`)はバイナリ実行ファイルを暗号化する技術として、コンパイルされた時点以後のプログラムを暗号化することで行われる。しかし、上手なりバーサー(`reverser`)はこのようなスキームの秘密キーを容易に捜し出すことができる。このような問題を解決するためにコード暗号化スキームは安全なキー管理方法を要求する。

30

【0006】

このために、CappaertとJungはランタイム(`runtime`)でバイナリファイルと関連がある秘密キーを生成するスキームを提案した。しかし、Cappaertのスキームは正しい秘密キーの生成が不可能な問題点があった。もし、コード暗号化スキームで秘密キーが正しく生成されないと、これはプログラム衝突や他の意図しなかった問題を誘発するようになる。また、Jungのスキームはコードの長さをひどく長くする。これは効率性と関連された問題を発生させる。

40

【発明の概要】

【発明が解決しようとする課題】

【0007】

したがって、本発明は上述したような従来技術の問題点を解決するためになされたもので、その目的は、1回呼び出されるコードブロックは自分を呼び出すコードブロックを利用して暗号/復号化し、2回以上呼び出されるコードブロックは乱数で生成した暗号キーで暗号/復号化するインデックステーブル基盤のコード暗号化及び復号化装置及びその方法を提供することにある。

50

## 【0008】

本発明の他の目的は、コードブロックの開始アドレス、呼出回数、ブロックのサイズなどを保存するインデックステーブルを利用して実行コードをブロックで区分するか呼出回数を計算するインデックステーブル基盤のコード暗号化及び復号化装置及びその方法を提供することにある。

## 【課題を解決するための手段】

## 【0009】

前記目的を達成するために本発明は、コンピュータプログラムの実行ファイルを暗号化するインデックステーブル基盤のコード暗号化装置にあって、前記実行ファイルの実行コードを呼出コードによりコードブロックで区分し、各コードブロックが呼び出される回数及び開始アドレスをインデックステーブルに保存するインデックス生成部と、前記コードブロックを暗号キーで暗号化するが、1回呼び出されるコードブロック(以下、第1タイプのコードブロック)の暗号キーは前記第1タイプのコードブロックを呼び出すコードブロック(以下、呼出ブロック)を利用して生成し、2回以上呼び出されるコードブロック(以下、第2タイプのコードブロック)の暗号キーは乱数で生成して前記実行ファイルに保存するブロック暗号化部と、を含むことを特徴とする。

10

## 【0010】

また、本発明は、インデックステーブル基盤のコード暗号化装置において、前記実行コードはバイナリコードまたはアセンブリコードであり、前記呼出コードはバイナリコードまたはアセンブリコードの分岐コードまたはジャンプコードであることを特徴とする。

20

## 【0011】

また、本発明は、インデックステーブル基盤のコード暗号化装置において、前記インデックス生成部は、最後の実行コードを呼出コードで含む一連の実行コードを一つのコードブロックで区分し、前記呼出コードは前記呼出コードが含まれたコードブロック以外の他のコードブロックを呼び出すことを特徴とする。

## 【0012】

また、本発明は、インデックステーブル基盤のコード暗号化装置において、前記インデックス生成部は、コードブロックの呼出回数をインデックステーブルに保存し、各コードブロックの呼出コード各々に対して、前記呼出コードが呼び出すコードブロックの呼出回数を1ずつ増加させて保存することを特徴とする。

30

## 【0013】

また、本発明は、インデックステーブル基盤のコード暗号化装置において、前記ブロック暗号化部は、前記第1タイプのコードブロックの呼出ブロックをハッシュして前記第1タイプのコードブロックの暗号キーを生成することを特徴とする。

## 【0014】

また、本発明は、インデックステーブル基盤のコード暗号化装置において、前記ブロック暗号化部は、最初コードブロックの暗号キーを初期キーで決めることを特徴とする。

## 【0015】

また、本発明は、インデックステーブル基盤のコード暗号化装置において、前記ブロック暗号化部は、前記第2タイプのコードブロックの暗号キーを初期キーで暗号化して前記実行ファイルのデータ領域に保存することを特徴とする。

40

## 【0016】

また、本発明は、インデックステーブル基盤のコード暗号化装置において、前記ブロック暗号化部は、前記インデックステーブルを前記実行ファイルのデータ領域に保存することを特徴とする。

## 【0017】

また、本発明は、インデックステーブル基盤のコード暗号化装置において、前記インデックス生成部は、前記コードブロックのサイズを前記インデックステーブルに保存することを特徴とする。

## 【0018】

50

また、本発明は、インデックステーブル基盤のコード暗号化装置により暗号化された実行ファイルを復号化するインデックステーブル基盤のコード復号化装置であって、前記実行ファイルの暗号化された実行コードをコードブロック単位でコードブロックの暗号キーを利用して復号化するが、前記インデックステーブルを参照して、前記コードブロックが第1タイプのコードブロックであれば、前記コードブロックの呼出ブロックで暗号キーを生成して利用し、第2タイプのコードブロックであれば、前記コードブロックの保存された暗号キーを利用するブロック復号化部と、復号化されたコードブロックの最後の呼出コードが実行されると、前記コードブロックをさらに暗号化するブロック再暗号部と、を含むことを特徴とする。

#### 【0019】

また、本発明は、インデックステーブル基盤のコード復号化装置において、前記ブロック再暗号部は、前記コードブロックが反復実行される場合、反復回数を前記インデックステーブルに保存して前記コードブロックが呼び出される時ごとに前記反復回数を減少させて、反復回数が0回になれば、前記コードブロックを再暗号化することを特徴とする。

#### 【0020】

また、本発明は、コンピュータプログラムの実行ファイルを暗号化するインデックステーブル基盤のコード暗号化方法であって、(b)前記実行ファイルの実行コードを呼出コードによりコードブロックで区分し、各コードブロックが呼び出される回数及び開始アドレスをインデックステーブルに保存し、2回以上呼び出されるコードブロック(以下、第2タイプのコードブロック)に対して暗号キーを乱数で生成して保存するステップと、(c)前記コードブロックを前記コードブロックの暗号キーで暗号化するが、1回呼び出されるコードブロック(以下、第1タイプのコードブロック)の暗号キーは前記第1タイプのコードブロックを呼び出すコードブロック(以下、呼出ブロック)を利用して生成するステップと、を含むことを特徴とする。

#### 【0021】

また、本発明は、インデックステーブル基盤のコード暗号化方法において、前記実行コードはアセンブリコードであり、前記呼出コードはアセンブリコードの分岐コードまたはジャンプコードであることを特徴とする。

#### 【0022】

また、本発明は、インデックステーブル基盤のコード暗号化方法において、前記ステップ(b)は、(b1)前記実行コードを順次に検査するステップと、(b2)検査した実行コードが呼出コードであれば、呼出コードの被演算子からアドレスを抽出するステップと、(b3)前記アドレスが前記インデックステーブルに存在しなければ、前記アドレスを前記インデックステーブルにコードブロックの開始アドレスで挿入するステップと、(b4)前記アドレスが前記インデックステーブルに存在すれば、前記アドレスに該当するコードブロックの呼出回数を1回増加させるステップと、(b5)前記コードブロックの呼出回数が1回以上であれば、前記コードブロックの暗号キーを乱数で生成して保存するステップと、を含むことを特徴とする。

#### 【0023】

また、本発明は、インデックステーブル基盤のコード暗号化方法において、前記ステップ(b)は、(b6)前記実行ファイルの最後の実行コードを検査すれば、各コードブロックのサイズを計算して前記インデックステーブルに保存するステップをさらに含むことを特徴とする。

#### 【0024】

また、本発明は、前記インデックステーブル基盤のコード暗号化方法により暗号化された実行ファイルを復号化するインデックステーブル基盤のコード復号化方法であって、(d)前記実行ファイルの暗号化された実行コードをコードブロック単位でコードブロックの暗号キーを利用して復号化するが、前記インデックステーブルを参照して、前記コードブロックが第1タイプのコードブロックであれば、前記コードブロックの呼出ブロックで暗号キーを生成して利用し、第2タイプのコードブロックであれば、前記コードブロックの保

10

20

30

40

50

存された暗号キーを利用するステップと、(f)復号コードブロックの最後の実行コードが実行されると、前記コードブロックをさらに暗号化するステップと、を含むことを特徴とする。

【発明の効果】

【0025】

上述のように、本発明によるインデックステーブル基盤のコード暗号化及び復号化装置及びその方法によれば、インデックステーブルを利用してコードブロックの呼出回数によって暗号キーを相違に生成することにより、安全なコード暗号化が可能であるだけでなく、暗号化または復号化の実行時間を短縮させて保存するデータの量を減らす効果が得られる。

10

【図面の簡単な説明】

【0026】

【図1】本発明を実施するための全体システム構成の一例を示した図である。

【図2】(a)は、本発明の一実施形態による実行ファイルの一例であり、(b)は、本発明の一例によって実行ファイルがメモリーに常住された状態を表示した図である。

【図3】本発明に使われた用語及び略語表である。

【図4】(a)～(c)は、本発明の一実施形態によるインデックステーブル基盤のコード暗号化方法を説明するフローチャートである。

【図5】(a)及び(b)は、本発明の一実施形態によるインデックステーブル基盤のコード暗号化方法の疑似コード(Pseudo-code)である。

20

【図6】本発明の一実施形態によるキーチェーンの一例を示した図である。

【図7】(a)及び(b)は、本発明による一実施形態によるプログラミング言語とアセンブリコードを例示した図である。

【図8】本発明の一実施形態による実行コードとそれによるインデックステーブル生成の一例を表示した図である。

【図9】本発明の一実施形態による実行コードとそれによるインデックステーブル生成の一例を表示した図である。

【図10】本発明の一実施形態によるインデックステーブル基盤のコード復号化方法を説明するフローチャートである。

【図11】本発明の一実施形態によるインデックステーブル基盤のコード復号化方法の疑似コード(Pseudo-code)である。

30

【図12】本発明の一実施形態によるインデックステーブル基盤のコード暗号化装置の構成に対するブロック図である。

【図13】本発明の一実施形態によるインデックステーブル基盤のコード復号化装置の構成に対するブロック図である。

【発明を実施するための形態】

【0027】

以下、添付図面を参照して本発明の好ましい実施の形態について詳細に説明する。

【0028】

また、本発明の説明において同一部分には同一符号を付けて、その繰り返し説明は省略する。

40

【0029】

先に、本発明を実施するための全体システムの構成の一例に対して図1を参照して説明する。

【0030】

図1に示したように、本発明を実施するための全体システムは、暗号化装置30と復号化装置40で構成される。

【0031】

暗号化装置30または復号化装置40は、プログラムで構成されてコンピュータ端末11、12に設置され実行される。コンピュータ端末11、12に設置されたプログラムは

50

一つのシステム 30、40 のように動作できる。一方、他の実施形態として、暗号化装置 30 または復号化装置 40 は A S I C (注文型半導体) など一つの電子回路で構成して実施できる。すなわち、ソフトウェア形態、F P G A チップや多数個の回路素子で構成された電子回路の形態で構成することもできる。その外の可能な他の形態でも実施できる。しかし、以下では説明の便宜のためにコンピュータ端末 11、12 に具現された暗号化装置 30 または復号化装置 40 で説明する。

#### 【0032】

実行ファイル 60 はコンピュータ端末 11 の保存媒体にあらかじめ保存され、保存された実行ファイル 60 を暗号化装置 30 により読み出して入力することができる。暗号化装置 30 は実行ファイル 60 の実行コードを暗号化して暗号化された実行ファイル 60 a を生成する。

10

#### 【0033】

図 2 の (a) に示したように、実行ファイル 60 は実行コード領域 61 とデータ領域 62 で区分される。実行コード 61 はコードブロック 62 で区分される。好ましくは、実行ファイル 60 はバイナリコード、アセンブリコードなどの実行コード 61 で構成される。実行コード 61 が中央処理装置 (C P U) により直接実行されるコードであれば、機械語またはバイナリコードである必要があるが、インタプリタなどにより解釈され実行される場合はそのインタプリタに合わせるアセンブリコードなどで生成されなければならない。

#### 【0034】

実行コード領域 61 はアセンブリコードなど実行コードを保存するための領域であり、データ領域 62 はデータを保存するための領域である。

20

#### 【0035】

この時、暗号化装置 30 は実行コード 61 をコードブロック 63 単位で暗号化する。したがって、コードブロック単位で復号化することも可能である。例えば、“コードブロック 3” だけを別に復号化することができる。

#### 【0036】

コードブロック 63 は一連の実行コード 61 a で構成される。“実行コード” という用語は実行コード全体を意味する用語や一つのラインである実行コードを意味する用語で混用する。

#### 【0037】

コードブロック 63 内の実行コード 61 a は順次に実行され、コードブロック 63 内の最後の実行コード 61 a は呼出コードとして、呼出コードが実行されると、他のコードブロック 63 に移動され実行される。呼出コードはアセンブリコードにおいてジャンプコード (j u m p c o d e) または分岐コード (b r a n c h c o d e) などを意味する。すなわち、コードブロック 63 の呼出コードにより順次に次の実行コードが実行されないで、他の位置にあるコードブロック 63 が新しく開始される。

30

#### 【0038】

また、前記実行コードのコードブロックに対する情報はインデックステーブル 70 に保存される。インデックステーブル 70 はデータ領域 62 に保存される。インデックステーブル 70 は各コードブロックのアドレス (または開始アドレス)、ブロックのサイズ、呼出回数などを保存するテーブルで構成される。好ましくは、インデックステーブル 70 は初期キーで暗号化されて保存できる。また、データ領域全体も初期キーにより暗号化されて保存できる。

40

#### 【0039】

前記のように、暗号化された実行ファイル 60 a はネットワーク 20 を介して配布されるかオフライン上に配布される。配布された実行ファイル 60 a はコンピュータ端末 12 に設置され実行される。

#### 【0040】

図 2 の (b) に示したように、実行ファイル 60 a の実行コード 61 及びデータ領域 62 はコンピュータ端末 12 のメモリー 12 b にローディングされて常住する。各実行コード

50

6 1 aは1ラインずつ(一つのコマンドずつ)コンピュータ端末1 2のCPU 1 2 aにより読み出され実行される。この時、実行コード6 1は全体がメモリー1 2 bにローディングされるか、一部だけがローディングされることもできる。この時、一つのラインの実行コードはアドレスにより識別され、一つのコードブロックは多数の実行コード(多数のラインコマンド)で構成される。

#### 【0041】

一方、実行ファイル6 0 aがローディングされ実行されるとき(ランタイムの場合)、実行ファイル6 0 aにも、固定されたデータ領域6 2と、スタック(stack)または流動的なデータのための流動的なデータ領域6 5が生成される。以下では区別せずデータ領域6 2だけで記載する。

10

#### 【0042】

上述のように、一つのコードブロック6 3は順次実行され、コードブロック6 3の最後の実行コードである呼出コードが実行されると、他のコードブロックにジャンプされ実行される。同一なコードブロック内で反復されて順次実行される場合もある。

#### 【0043】

復号化装置4 0は暗号化された実行ファイル(6 0または6 0 a)の実行コードを復号化して実行コードを実行させる。この時、復号化装置4 0は実行されるコードブロック6 3だけを復号化し、復号化されたコードブロック6 3を全て実行すれば、復号化したコードブロック6 3を再暗号化する。そして、次に実行されるコードブロック6 3を復号化して復号化されたコードブロック6 3を実行する。

20

#### 【0044】

図2の(b)の例において、コードブロック1→コードブロック3→コードブロック2を実行すると仮定する場合、先にコードブロック1を復号化する。コードブロック1の復号化を全て実行すれば、コードブロック1を再暗号化し、次に実行するコードブロック3を復号化する。そして、コードブロック3の復号化を全て実行すれば、コードブロック3を暗号化してコードブロック2を復号化する。

#### 【0045】

次に、本発明を説明するための用語及び略語について図3を参照して説明する。

#### 【0046】

IKは初期キーとして、乱数とインデックステーブル(indexed table)を同時に保護する。乱数は、多重ブロックにより呼び出される基本ブロック(またはコードブロック)を暗号化し、また初期キーIKは乱数を暗号化する。初期キーIKを保護してまた提供することはアプリケーションに依存し、初期キーIKは外装ハードディスクやTPM(Trusted Platform Module)などの外部記憶媒体に記憶することができる。したがって、初期キーIKがオフライン上で安全に分配され記憶されると仮定する。

30

#### 【0047】

乱数は2個以上のコードブロックから呼び出されるコードブロックを暗号化し、初期キーIKを暗号化する。保護キーPKは初期キーIKにより暗号化された乱数を意味する。保護キーPKはバイナリコード(または実行ファイル)のデータセクション(またはデータ領域)に記憶される。また、暗号化または復号化アルゴリズムは従来の暗復号化技法を利用するので、その具体的な説明は省略する。

40

#### 【0048】

$E_K()$ と $D_K()$ は各々暗号キーKを利用して暗号化または復号化演算を行うことを表示する。 $H()$ は一方方向のハッシュ関数を意味する。

#### 【0049】

次に、本発明によるインデックステーブル基盤のコード暗号化及び復号化装置及びその方法が充足すべき要求事項、すなわち、安全なコード暗号技法の要求事項について説明する。

#### 【0050】

50

先に、気密性を備える必要がある。原本バイナリコード(実行コードまたはアセンブリコード)は気密性維持を通じて静的分析から保護されなければならない。バイナリコードをフローや制御移動などの動的分析から保護するためにメモリー上には最小個数のコードブロックが存在する必要がある。メモリー上にコードが暗号化されている限りプログラムは静的及び動的分析から保護される。

【0051】

また、メモリーダンプが防止される必要がある。もし、単一ルーチンを通じて一つのプログラムを丸ごと暗号化する場合、復号化ルーチンがプログラム全体を復号化した後、開始地点を指定すると、これはメモリーダンプ攻撃に脆弱になる。したがって、最小限のプログラムが復号化される必要があり、残り部分は暗号化された状態を維持しなければならない。

10

【0052】

また、正しいキーチェーンを具備する必要がある。コード暗号化がプログラムに適用されると、正しいキー(または暗号キー)が必要になる。もし、多重経路が存在するとき正しいキーチェーンを持たないと、システムクラッシュや意図しなかった実行が可能である。

【0053】

また、不正使用(tampering)防止を満足しなければならない。不正使用から保護するためには無欠性維持が必要である。これは下記のような特性が必要とする。

【0054】

暗号化過程で、1ビットが一つの基本ブロックを変更すれば、その効果はすべての暗号化されたブロックに影響を及ぼさなければならない。

20

【0055】

復号化過程で、もし、暗号化されたブロックで一つ以上のビットが変更されると、復号化結果は一つ以上のビットが変更されなければならない。

【0056】

次に、本発明の一実施形態によるインデックステーブル基盤のコード暗号化方法について図4を参照してより具体的に説明する。

【0057】

図4の(a)に示したように、本発明によるインデックステーブル基盤のコード暗号化方法は(a)コンパイルするステップ(ステップS10)、(b)インデックステーブルを生成するステップ(ステップS20)及び(c)コードブロックを暗号化するステップ(ステップS30)に区分される。

30

【0058】

前記ステップ(a)(またはコンパイルするステップ)は、プログラムソースコードをコンパイルして実行コード(アセンブリコードまたはバイナリコード)を生成するステップである。

【0059】

前記ステップ(b)では、実行コードを呼出コードによりコードブロックで区分し、各コードブロックの呼出回数及び開始アドレスなどの情報をインデックステーブルで生成する(ステップS20)。この時、呼出回数が2回以上のコードブロック(または第2タイプのコードブロック)に対して乱数で暗号キーを生成して記憶する。

40

【0060】

この時、呼出回数が2回以上のコードブロックを第2タイプのコードブロックと称し、その他のコードブロックを第1タイプのコードブロックと称する。第1タイプのコードブロックは呼出回数が1回である。一方、最初のコードブロックは初めてプログラムの開始により1回呼び出されることで仮定する。

【0061】

次に、ステップ(b)をより具体的に説明する前に、ステップ(c)を具体的に説明する。前記ステップ(c)は、図4の(c)に示したとおりである。

【0062】

50

図4の(c)に示したように、前記ステップ(c)は、コードブロックを暗号キーで暗号化する。この時、1回呼び出されるコードブロックの暗号キーはこのコードブロックを呼び出すコードブロック(以下、呼出ブロック)を利用して生成する。すなわち、呼出ブロックを利用して暗号化する。また、2回以上呼び出されるコードブロックは乱数で生成された暗号キーを通じて暗号化される。例えば、乱数自体を暗号キーを使ってコードブロックを暗号化する。

【0063】

コードブロック(または第1タイプのコードブロック)は、次の[数1]により暗号化される。

【0064】

【数1】

$$C_i = \text{Enc}_{H(P_{i-1})}(P_i), \text{ where } H(P_0) = IK.$$

【0065】

iはインデックステーブル内にあるコードブロックの手順である。もし、フラグ(または多重経路フラグ)が“0”ではないと、ブロック(またはコードブロック)は乱数により暗号化されたのである。初期キーIKは乱数を暗号化して実行ファイル内に保存される。もし、乱数が露出されると、該当ブロックは乱数により復号化され、したがって、ブロックは攻撃者により分析可能になるからこのような暗号化ステップが必要である。

【0066】

インデックステーブルは正しいキーチェーンを形成するために使われる。テーブル構成は次のようである。まず、基本ブロック(またはコードブロック)の現在アドレスを保存した後、呼出コード(jumpやbranch系列の命令語)をポインタを移動しながら調査する。このような呼出コードの命令語は次に実行されるコードブロックのアドレスを後に(被演算子として)有している。

【0067】

次のアドレス(呼出コードの被演算子)が現在の基本ブロックを示している場合、これはループ(または反復文)や再帰を意味する。ループや再帰が発生すれば、cmp命令語などを通じて呼出回数を決めることができるようになり、このような呼出回数をテーブルに保存する。これと同様に、現在のブロックがもうテーブルに保存されている場合、これは該当コードブロックの多重呼出(2回以上の呼び出し)を意味する。このとき保護キーPKが生成されてバイナリイメージ(または実行ファイルのバイナリイメージ)のデータ領域(data section)に保存される。保護キーPKは乱数を暗号化したもので、乱数がコードブロックを暗号化する暗号キーとして使用される。

【0068】

図6に示したように、基本ブロック(またはコードブロック)Dは、B、CそしてFなどの多重ブロックにより呼び出される。すなわち、3個の相違なるコードブロックにより呼び出されている。Bの暗号キーはAブロックのハッシュ値であり、Cの秘密キーと同一である。FはAにより直接呼び出されなくて、Dにより呼び出される。その時、乱数rがDの秘密キー(または暗号キー)で生成され、これはさらに初期キー(IK)で暗号化される。その結果、保護キー(PK)が新たに生成され、これはバイナリファイル(または実行ファイル)内に保存される。保護キー(PK)は乱数rが初期キー(IK)により暗号化された値を意味する。

【0069】

リソースやウィンドウなどの一般的なコンピュータ運営体制は、実行ファイルイメージ内に変数を保存することができるデータ領域を支援し、したがって、保護キー(PK)はこのような実行ファイル内に存在するデータ領域内に保存できる。

【0070】

インデックステーブルは反復呼び出される回数を含む。もし、これが考慮されなければ

10

20

30

40

50

、ループまたは再帰などを意味する基本ブロック(またはコードブロック)が数回復号化できる。したがって、ループや再帰回数をテーブル内に表示しておくことによりこのような問題を解決することができる。もし、基本ブロックが呼び出されると、テーブル内に保存された呼出回数が1減少し、このような方式で呼出回数が“0”になれば、該当ブロックはメモリーダンプ防止のために再暗号化される。

#### 【0071】

例えば、ループの場合、C言語で使われた例示は図7のようである。cmp命令語において二番目の被演算子の値は“OA”である。これはブロック“loc\_401006”は10回実行されることを意味し、これはブロックのループまたは再帰回数であることが分かる。

10

#### 【0072】

図4の(c)を参照して加えて説明すれば、最初のコードブロック $P_0$ は初期キーIKで暗号化する(ステップS31)。インデックステーブルから次のコードブロックのアドレス(開始アドレス)及び種類を読み出して(ステップS32)、何の種類であるかを判断する(ステップS33)。インデックステーブルでフラグ(または多重経路呼出フラグ)が“0”であれば、第1種類のコードブロックであり、“0”ではなければ、第2種類のコードブロックである。

#### 【0073】

現在のコードブロック $P_i$ が第1種類のコードブロックであれば、直前の呼出ブロック(またはインデックステーブルで直前コードブロック) $P_{i-1}$ をハッシュしてハッシュ値 $H(P_{i-1})$ で現在のコードブロック $P_i$ を暗号化する(ステップS34)。そして、現在のコードブロック $P_i$ が第2種類のコードブロックであれば、乱数を発生して(ステップS35)、乱数 $r$ で現在のコードブロックを暗号化する(ステップS36)。この時、ステップ(b)で発生させた乱数を利用することもでき、ステップ(b)で乱数を発生しなければこのステップで乱数を発生する。

20

#### 【0074】

インデックステーブルから読み出した現在のコードブロック $P_i$ が最後のコードブロックであるか否かを確認して(ステップS37)、最後のコードブロックであれば終了し、そうではなければ、次のコードブロックを読み出して前記過程を繰り返す。

#### 【0075】

次に、前記ステップ(b)を、図4bと図8及び図9を参照してより具体的に説明する。図4bは、インデックステーブルを生成するステップを説明するフローチャートであり、図9は、図8の実行コードの例からインデックステーブルを生成した一例を示した図である。

30

#### 【0076】

図8に示したように、例示コード(または実行コード)は大きく5個のコードブロックに分けられる。基本ブロック(またはコードブロック)はjumpやbranch系列の命令語(または呼出コードの命令語)により分けられる。先に初期化が実行される。0x0040103Eはプログラムの開始地点でセッティングされる。以後jumpやbranch系列の命令語が探索される。

40

#### 【0077】

図9に示したように、もし、命令語がjumpやbranchであれば、これは他のブロックの一番目のアドレスになるため、被演算子をテーブルに保存する。例えば、0x0040105Aはアドレス0x0040104Fに存在する“jne 0x0040105A”コードによりテーブルに保存される。命令語の次のアドレスは他のブロックの一番目のアドレスになる。すなわち、0x00401051はテーブルに保存される。このような方法で、0x0040106Cと0x00401060が順に保存される。アドレス0x0040106Aで命令語は“jmp 0x00401051”で識別される。0x00401051はもうテーブルに保存されているので、これは多重呼出されるブロックであると言える。したがって、該当ブロックの情報が更新されて乱数が生成される。この

50

ような方法ですべてのブロックが識別され区分される。

【0078】

図9のインデックステーブルにおいて、アドレスはコードブロックの開始アドレスであり、コードブロックを識別する識別子の機能をする。また、呼出回数は他のコードブロックが該当コードブロックを呼び出す回数を記録したのである。フラグは多重経路呼出フラグとして、1回呼び出される場合“0”を記録し、2回以上呼び出されて多重経路の呼出が形成される場合“1”を記録する。

【0079】

前記ステップ(b)及びステップ(c)に対する疑似コード(Pseudo-code)は各々図5の(a)と(b)のようである。

【0080】

次に、本発明の一実施形態によるインデックステーブル基盤のコード復号化方法について図10を参照してより具体的に説明する。

【0081】

図10に示したように、本発明によるインデックステーブル基盤のコード復号化方法は、大きく(d)コードブロックを復号化するステップ(ステップS40)、(e)コードブロックの実行コードを実行するステップ(ステップS50)及び(e)復号化されたコードブロックを再暗号化するステップ(ステップS60)に区分される。

【0082】

先に、最初のコードブロックは初期キーIKで復号化する(ステップS41)。最初のコードブロックの実行が完了されると、次のコードブロックを読み出す。この時、インデックステーブルで次のコードブロックのアドレスを検索する(ステップS42)。次のコードブロックがどの類型(第1類型または第2類型)であるかを判断して(ステップS43)、第1類型であれば、直前のコードブロック(または呼出ブロック)をハッシュしてハッシュ値を暗号キーとしてコードブロックを復号化する(ステップS44)。第2類型であれば、保存された保護キーを持ち込んで初期キーIKで復号化して乱数を抽出して(ステップS45)、抽出された乱数を暗号キーとしてコードブロックを復号化する(ステップS46)。そして、コードブロックを実行し(ステップS50)、実行されたコードブロックを再暗号化する(ステップS60)。すべてのコードブロックが完了されて最後のコードブロックになると終了する(ステップS47)。

【0083】

すなわち、前記ステップ(d)は、実行ファイルの暗号化された実行コードをコードブロック単位でコードブロックの暗号キーを利用して復号化するが、前記インデックステーブルを参照して、前記コードブロックが第1類型のコードブロックであれば、前記コードブロックの呼出ブロックで暗号キーを生成して利用し、第2類型のコードブロックであれば、前記コードブロックの保存された暗号キー(または保護キーを復号化した乱数)を利用する。

【0084】

先に、プログラムが始まる前に、インデックステーブルを参照してプログラムのエン트리ポイント(entry point)を探す。そして、暗号化されたコードブロック $C_i$ を実行コードまたはコードブロック $P_i$ で復号化する。暗号化されたコードブロックは直前ブロックのハッシュ値を利用して復号化され、復号化されたコード $P_i$ が実行される。もし、直前ブロック $P_{i-1}$ が $P'_{i-1}$ に変更(tampering)された場合、暗号キーは $H(P_{i-1}) \neq H(P'_{i-1})$ である。したがって、コードブロック $P_i$ は正しく復号化されない。

【0085】

一方、インデックステーブルはフラグを含む。フラグはコードブロックが乱数を利用するか利用しないかを示すフラグである。もし、フラグが“1”であれば、暗号化されたコードブロックは乱数で復号化しなければならない。復号化コード作業が完了すれば、復号化されたコードブロックはさらに暗号化されてメモリーに保存される。

## 【0086】

次に、本発明の一実施形態によるインデックステーブル基盤のコード暗号化装置30の構成について図12を参照してより具体的に説明する。

## 【0087】

図12に示したように、本発明による暗号化装置30はインデックス生成部31及びブロック暗号化部32で構成される。

## 【0088】

インデックス生成部31は、実行ファイルの実行コードを呼出コードによりコードブロックで区分し、各コードブロックが呼び出される回数及び開始アドレスをインデックステーブルに保存する。

## 【0089】

ブロック暗号化部32は、コードブロックを暗号キーで暗号化するが、1回呼び出されるコードブロック(以下、第1タイプのコードブロック)の暗号キーは前記第1タイプのコードブロックを呼び出すコードブロック(以下、呼出ブロック)を利用して生成し、2回以上呼び出されるコードブロック(以下、第2タイプのコードブロック)の暗号キーは乱数で生成して前記実行ファイルに保存する。

## 【0090】

この時、実行コードはバイナリコードまたはアセンブリコードであり、前記呼出コードはバイナリコードまたはアセンブリコードの分岐コードまたはジャンプコードである。

## 【0091】

一方、インデックス生成部31は最後の実行コードを呼出コードで含む一連の実行コードを一つのコードブロックで区分し、前記呼出コードは前記呼出コードが含まれたコードブロック以外の他のコードブロックを呼び出す。また、インデックス生成部31はコードブロックの呼出回数をインデックステーブルに保存し、各コードブロックの呼出コード各々に対して、前記呼出コードが呼び出すコードブロックの呼出回数を一つずつ増加させて保存する。また、インデックス生成部31は前記コードブロックのサイズを前記インデックステーブルに保存する。

## 【0092】

ブロック暗号化部32は、前記第1タイプのコードブロックの呼出ブロックをハッシュして前記第1タイプのコードブロックの暗号キーを生成する。また、ブロック暗号化部32は最初コードブロックの暗号キーを初期キーで決める。また、ブロック暗号化部32は、前記第2タイプのコードブロックの暗号キーを初期キーで暗号化して前記実行ファイルのデータ領域に保存する。好ましくは、ブロック暗号化部32は前記インデックステーブルを前記実行ファイルのデータ領域に保存する。

## 【0093】

次に、本発明の一実施形態によるインデックステーブル基盤のコード復号化装置40の構成について図13を参照してより具体的に説明する。

## 【0094】

図13に示したように、本発明による復号化装置40はブロック復号化部41及びブロック再暗号部42で構成される。

## 【0095】

ブロック復号化部41は、前記実行ファイルの暗号化された実行コードをコードブロック単位でコードブロックの暗号キーを利用して復号化するが、前記インデックステーブルを参照して、前記コードブロックが第1タイプのコードブロックであれば、前記コードブロックの呼出ブロックで暗号キーを生成して利用し、第2タイプのコードブロックであれば、前記コードブロックの保存された暗号キーを利用する。

## 【0096】

ブロック再暗号部42は、復号化されたコードブロックの最後の呼出コードが実行されると、前記コードブロックを再暗号化する。特に、ブロック再暗号部42は、前記コードブロックが反復実行される場合、反復回数を前記インデックステーブルに保存して前記コ

10

20

30

40

50

ードブロックが呼び出される時ごとに前記反復回数を減少させて、反復回数が“0”になれば、前記コードブロックを再暗号化する。

【0097】

前記暗号化または復号化装置に対する説明の中で省略した部分は上述した暗号化または復号化方法の記載を参照する。

【0098】

以上、本発明を具体的な実施形態を参照して詳細に説明したが、本発明の範囲は前述の実施形態によって限定されるべきではなく、特許請求の範囲の記載及びこれと均等なもの  
の範囲内で様々な変形が可能なことは、当該技術分野における通常の知識を持つ者には明  
らかである。

10

【産業上の利用可能性】

【0099】

本発明は、1回呼び出されるコードブロックを直前の呼出コードブロックで暗号/復号  
化し、2回以上呼び出されるコードブロックを乱数で生成した暗号キーで暗号/復号化す  
るインデックステーブル基盤のコード暗号化及び復号化装置の開発において有用である。

【符号の説明】

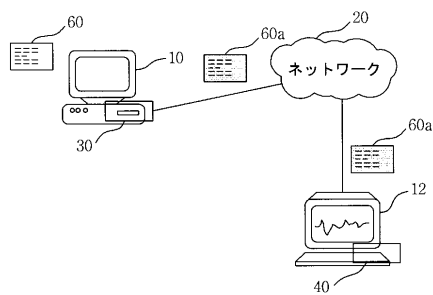
【0100】

- 11 コンピュータ端末
- 12 コンピュータ端末
- 20 ネットワーク
- 30 暗号化装置
- 31 インデックス生成部
- 32 ブロック暗号化部
- 40 復号化装置
- 41 ブロック復号化部
- 42 ブロック再暗号部
- 60 実行ファイル
- 60a 暗号化された実行ファイル
- 61 実行コード領域
- 61a 実行コード
- 62 データ領域
- 63 コードブロック
- 65 動的データ領域

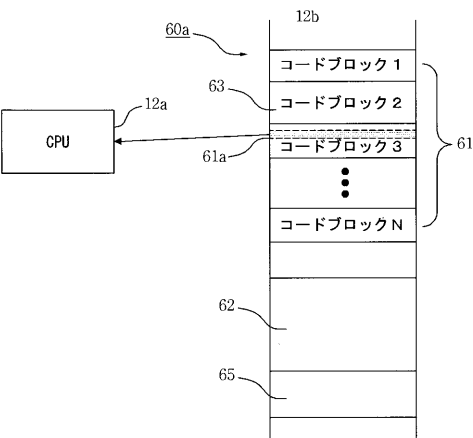
20

30

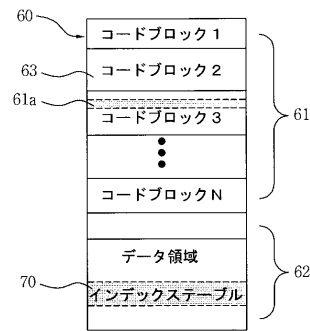
【図 1】



【図 2 b】



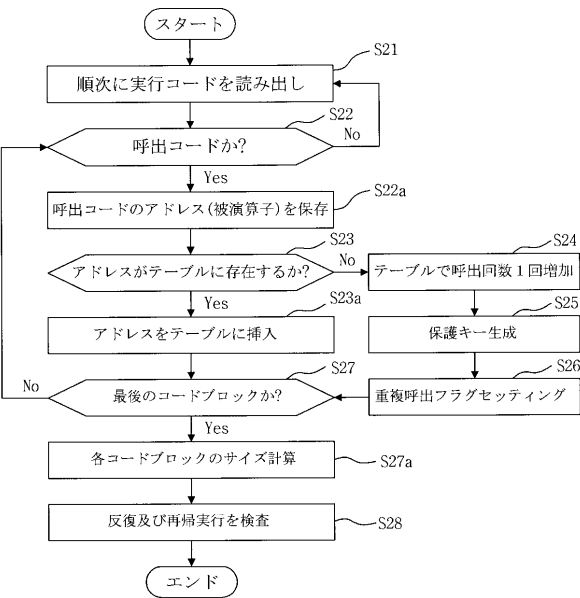
【図 2 a】



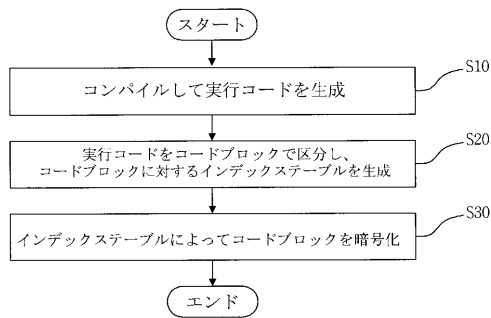
【図 3】

| 用語及び略語       | 説明                                                    |
|--------------|-------------------------------------------------------|
| I K          | 初期キー (Initial Key)                                    |
| P K          | 保護キー (Protected Key)                                  |
| $E_k(\cdot)$ | キー K による暗号化 (Encrypt with key K)                      |
| $D_k(\cdot)$ | キー K による復号化 (decrypt with key K)                      |
| $H(\cdot)$   | 一方向のハッシュ関数 (One-way hash function)                    |
| R a n d      | 乱数 (Random number)                                    |
| A ~ Z        | バイナリコードの基本ブロック<br>(Basic blocks in binary code)       |
| a ~ z        | 基本ブロック A ~ Z の m ビット<br>(m bits of basic block A ~ Z) |

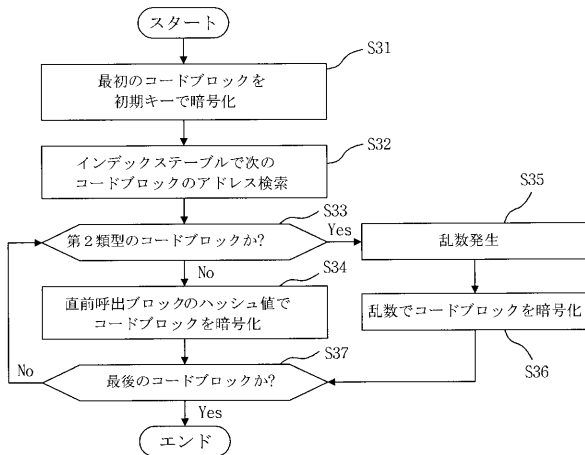
【図 4 b】



【図 4 a】



【図 4 c】



【図 5 a】

```

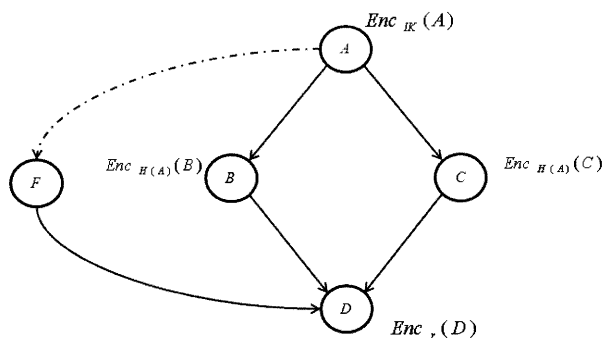
Procedure ConstructTable()
1  entrypoint ← Find_EntryPoint(); // store an address of entry point
2  currentPointer ← entrypoint;
3  nextPointer ← currentPointer++;
4  index ← 0;
5  while (File pointer is not end of file) {
6    if (Current_opcode == jump or Current_opcode == branch) {
7      branch or jump converted to an usual block
8      nextAddress ← operand;
9      store an address of current address;
10     if (currentAddress == nextAddress) {
11       loop or recursion
12       Tuple(index).Address ← currentAddress;
13       Tuple(index).Size ← nextBlock;
14       Tuple(index).Out ← prev_operand_2;
15       Tuple(index).Flag ← 0;
16       index ← next_pointerAddress; // i.e. number of calling, and no processed key
17     }
18     StoreAmbiguous(Tuple(index));
19   }
20   else {
21     if (FindAddress(Tuple(index).currentAddress)) {
22       repeated calling
23       GenerateProtectedKey();
24       StoreDataSection();
25       Tuple(index).Flag ← 1;
26     }
27   }
28   nextBlock ← Get_NextBlock(currentAddress);
29   Get next block's address;
30   currentAddress ← nextAddress;
31   Sort(Tuple);
32   Compare_blocksize(Tuple);
33   CheckIteration();
34   index++;
}
  
```

【図 5 b】

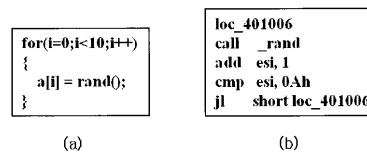
```

Procedure Encryption()
1  Compile(); // compile the source code to generate object or executable file
2
3  entrypoint ← Find_EntryPoint(); // store an address of entry point
4  currentAddress = entrypoint; // initialization
5  nextAddress = 0;
6
7  ConstructTable(); // this procedure is described in Fig. 10.
8
9  nextAddress = Find_next(entrypoint); // find an address of next block in current block
10 Encrypt(IK, entrypoint); // encrypt first block
11
12 while (File pointer is not end of file)
13 {
14   if (SearchTable(nextAddress)) // if next block's flag in the indexed table indicates 1
15   {
16     random = GenerateRandom(); // generate random number
17     Encrypt(random, nextAddress); // Encrypt with the random number
18     Encrypt(random, IK); // Encrypt the random number with the IK
19   }
20   else // next block's flag indicates 0
21     Encrypt(random, currentAddress); // Encrypt with the current block
22 }
  
```

【図 6】



【図 7】



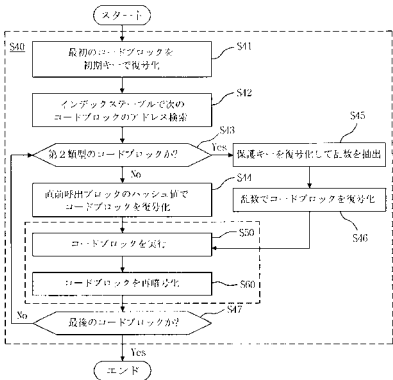
【図 8】

|   |          |      |                          |
|---|----------|------|--------------------------|
|   | 0040103E | mov  | ecx, 64h                 |
|   | 00401043 | idiv | eax, ecx                 |
|   | 00401045 | mov  | dword ptr [ebp-0Ch], edx |
|   | 00401048 | cmp  | dword ptr [ebp-10h], 8   |
|   | 0040104F | jne  | 0040105a                 |
| B | 00401051 | mov  | edx, dword ptr [ebp-10h] |
|   | 00401054 | add  | edx, 1                   |
|   | 00401057 | mov  | dword ptr [ebp-10h], edx |
| C | 0040105A | cmp  | dword ptr [ebp-10h], 5   |
|   | 0040105E | jge  | 0040106c                 |
|   | 00401060 | mov  | eax, dword ptr [ebp-4]   |
| D | 00401063 | imul | eax, dword ptr [ebp-10h] |
|   | 00401067 | mov  | dword ptr [ebp-4], eax   |
|   | 0040106A | jmp  | 00401051                 |
|   | 0040106C | mov  | ecx, dword ptr [ebp-4]   |
| E | 0040106F | cmp  | ecx, dword ptr [ebp-8]   |
|   | 00401072 | jle  | 0040108a                 |

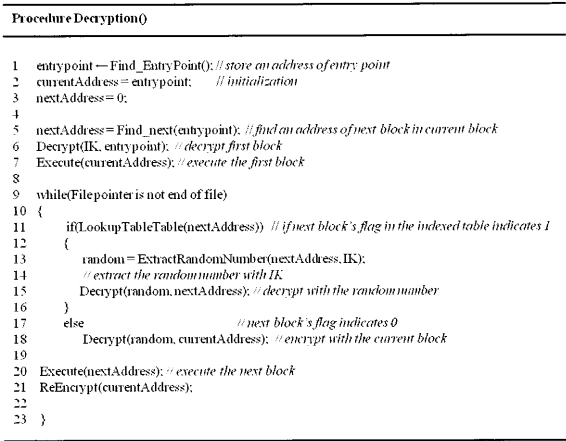
【図 9】

| アドレス(Offset) | ブロックサイズ | 呼出回数 | フラグ |
|--------------|---------|------|-----|
| 0x0040103E   | 19      | 1    | 0   |
| 0x00401051   | 9       | 2    | 1   |
| 0x0040105A   | 6       | 2    | 1   |
| 0x00401060   | 12      | 1    | 0   |
| 0x0040106C   | 8       | 2    | 1   |
| ....         |         |      |     |

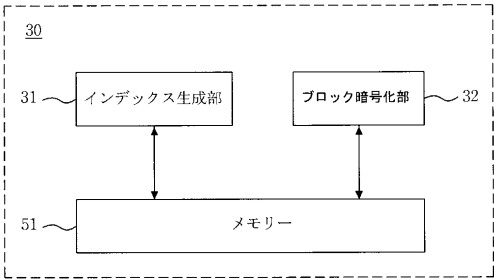
【図 10】



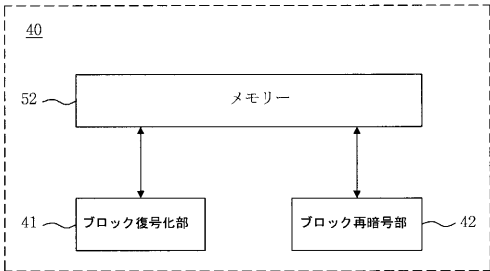
【図 11】



【図 12】



【図 13】



---

フロントページの続き

- (72)発明者 セウングジョー キム  
大韓民国、ギェオンッギード 440-746、スウォンシ、スングキュンクワン ユニバーシ  
ティ、シャープ27303
- (72)発明者 ドングホ ウォン  
大韓民国、ギェオンッギード 440-746、スウォンシ、スングキュンクワン ユニバーシ  
ティ、シャープ27303
- (72)発明者 スングキュ チョー  
大韓民国、ギェオンッギード 440-746、スウォンシ、スングキュンクワン ユニバーシ  
ティ、シャープ27303
- (72)発明者 ドングウィ シン  
大韓民国、ソウル 138-160、ソングパーク、ガラクードング、79-3
- (72)発明者 ヒェスク ジョー  
大韓民国、ギェオンッギード 440-746、スウォンシ、スングキュンクワン ユニバーシ  
ティ、シャープ27303
- (72)発明者 ドングヒュン チョイ  
大韓民国、ギェオンッギード 440-746、スウォンシ、スングキュンクワン ユニバーシ  
ティ、シャープ27303
- (72)発明者 ジャエーチェオル リョウ  
大韓民国、ダエジェオン 305-755、ユセオングーグ、エオエウソードング、ハンビト エ  
ーティーピー、132-801

Fターム(参考) 5B276 FA07

5J104 AA12 AA16 EA04 EA15 NA02 NA12 PA14