(12) 特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2018年8月23日(23.08.2018)

WIPO | PCT

(10) 国際公開番号
WO 2018/150619 A1

(51) 国際特許分類:
G06F 21/51 (2013.01)　*G06F 21/12* (2013.01)

(21) 国際出願番号：　PCT/JP2017/034354

(22) 国際出願日：　2017年9月22日(22.09.2017)

(25) 国際出願の言語：　日本語

(26) 国際公開の言語：　日本語

(30) 優先権データ：
特願 2017-024901　2017年2月14日(14.02.2017)　JP

(71) 出願人：日本電信電話株式会社 (NIPPON TELEGRAPH AND TELEPHONE CORPORATION) [JP/JP]；〒1008116 東京都千代田区大手町一丁目５番１号 Tokyo (JP).

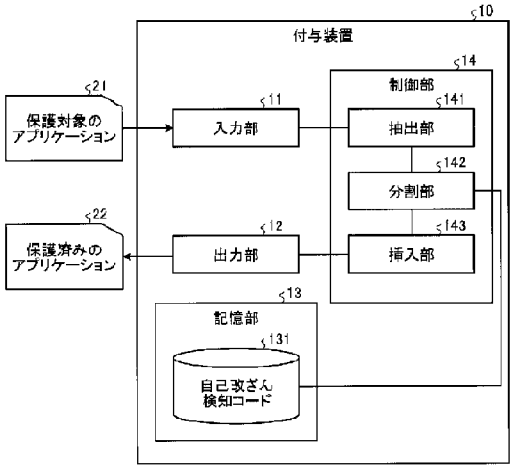
(72) 発明者：金井　文宏 (KANEI, Fumihiro)；〒1808585 東京都武蔵野市緑町３丁目９－１１ＮＴＴ知的財産センタ内 Tokyo (JP). 秋山 満昭(AKIYAMA, Mitsuaki)；〒1808585 東京都武蔵野市緑町３丁目９－１１ＮＴＴ知的財産センタ内 Tokyo (JP). 高田 雄太(TAKATA, Yuta)；〒1808585 東京都武蔵野市緑町３丁目９－１１ＮＴＴ知的財産センタ内 Tokyo (JP). 八木 毅 (YAGI, Takeshi)；〒1808585 東京都武蔵野市緑町３丁目９－１１ＮＴＴ知的財産センタ内 Tokyo (JP).

(74) 代理人：特許業務法人酒井国際特許事務所 (SAKAI INTERNATIONAL PATENT OFFICE)；〒1000013 東京都千代田区霞が関３丁目８番１号 虎の門三井ビルディング Tokyo (JP).

(81) 指定国(表示のない限り、全ての種類の国内保護が可能)：AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH,

(54) **Title:** IMPARTING DEVICE, IMPARTING METHOD, AND IMPARTING PROGRAM

(54) 発明の名称： 付与装置、付与方法及び付与プログラム



10 Imparting device
11 Input unit
12 Output unit
13 Storage unit
14 Control unit
21 Application to be protected
22 Protected application
131 Self-falsification detection code
141 Extraction unit
142 Division unit
143 Insertion unit

(57) Abstract: An extraction unit (141) extracts a block at random from among blocks of a command sequence that constitutes byte code of a first program, and extracts a block to be certainly executed before the randomly extracted block when the first program is executed. A division unit (142) divides a command sequence that constitutes byte code of a second program for detecting falsification of the first program into a plurality of blocks. An insertion unit (143) inserts, into different positions in the block extracted by the extraction unit (141), the blocks divided by the division unit (142) while maintaining the execution order in the second program.

(57) 要約：抽出部（１４１）は、第１のプログラムのバイトコードを構成する命令列のブロックから、ランダムにブロックを抽出し、第１のプログラムが実行される際に、当該ランダムに抽出したブロックより前に必ず実行されるブロックを抽出する。また、分割部（１４２）は、第１のプログラムの改ざんを検知する第２のプログラムのバイトコードを構成する命令列を、ランダムに複数のブロックに分割する。また、挿入部（１４３）は、分割部（１４２）によって分割された複数のブロックを、抽出部（１４１）によって抽出されたブロック内のそれぞれ異なる位置に、第２のプログラムにおける実行順序を維持して挿入する。

WO 2018/150619 A1

[続葉有]

CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) 指定国(表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, RU, TJ, TM), ヨーロッパ (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

添付公開書類:

一 国際調査報告(条約第21条(3))

明　細　書

発明の名称 ： 付与装置、付与方法及び付与プログラム

技術分野

[0001] 本発明は、付与装置、付与方法及び付与プログラムに関する。

背景技術

[0002] 従来、アプリケーションに、攻撃者による当該アプリケーションの改ざんを検知する自己改ざん検知機能を付与する技術が知られている。例えば、アプリケーションのバイトコードをクラス単位で暗号化しておき、実行時には復号ルーチンが必要に応じてバイトコードの復号、及び完全性検証を行う手法が知られている（例えば、非特許文献１を参照）。また、例えば、アプリケーションのソースコードに対して完全性検証を行う実行コードを多数自動挿入し、アプリケーション実行時にはそれらのコードが確率的に反応することで、攻撃者が全ての自己改ざん検知コードを無効化することを困難にする手法が知られている（例えば、非特許文献２を参照）。

先行技術文献

非特許文献

[0003] 非特許文献1：Mykola Protsenko, Sebastien Kreuter and Tilo Muller, “Dynamic Self-Protection and Tamperproofing for Android Apps using Native Code,” in Proceedings of the International Conference on Availability, Reliability and Security (ARES), pp. 129-138, 2015.

非特許文献2：Lannan Luo, Yu Fu, Dinghao Wu, Sencun Zhu, and Peng Liu “Repackage-proofing Android Apps,” in Proceedings of the International Conference on Dependable Systems and Networks (DSN), pp. 403-414, 2016.

発明の概要

発明が解決しようとする課題

[0004] しかしながら、従来の自己改ざん検知機能を付与する技術には、プログラムにおける自己改ざん検知機能の実装箇所が、攻撃者によって容易に発見されてしまう場合があるという問題があった。

[0005] 例えば、非特許文献１に記載の手法では、完全性検証処理を実装している箇所が固定位置であるため、当該箇所が容易に発見される場合があった。また、例えば、非特許文献２に記載の手法では、自己改ざん検知コードはソースコードの行単位で挿入される。このため、ソースコードがバイトコードにコンパイルされた際に、自己改ざん検知コードが挿入された複数の箇所に同じ命令列のブロックが現れることになり、シグネチャを用いたマッチングにより当該ブロックが容易に発見されてしまう場合があった。

課題を解決するための手段

[0006] 本発明の付与装置は、保護対象の第１のプログラムが改ざんされたことを検知する機能を、前記第１のプログラムに付与する付与装置であって、前記第１のプログラムのバイトコードを構成する命令列のブロックから、ランダムにブロックを抽出し、前記第１のプログラムが実行される際に、当該ランダムに抽出したブロックより前に必ず実行されるブロックを抽出する抽出部と、前記第１のプログラムの改ざんを検知する第２のプログラムのバイトコードを構成する命令列を、ランダムに複数のブロックに分割する分割部と、前記分割部によって分割された複数のブロックを、前記抽出部によって抽出されたブロック内のそれぞれ異なる位置に、前記第２のプログラムにおける実行順序を維持して挿入する挿入部と、を有することを特徴とする。

[0007] 本発明の付与方法は、保護対象の第１のプログラムが改ざんされたことを検知する機能を、前記第１のプログラムに付与する付与装置で実行される付与方法であって、前記第１のプログラムのバイトコードを構成する命令列のブロックから、ランダムにブロックを抽出し、前記第１のプログラムが実行される際に、当該ランダムに抽出したブロックより前に必ず実行されるブロックを抽出する抽出工程と、前記第１のプログラムの改ざんを検知する第２のプログラムのバイトコードを構成する命令列を、ランダムに複数のブロッ

クに分割する分割工程と、前記分割工程によって分割された複数のブロックを、前記抽出工程によって抽出されたブロック内のそれぞれ異なる位置に、前記第２のプログラムにおける実行順序を維持して挿入する挿入工程と、を含んだことを特徴とする。

発明の効果

[0008] 本発明によれば、プログラムにおける自己改ざん検知機能の実装箇所が、攻撃者によって容易に発見されないようにすることができる。

図面の簡単な説明

[0009] [図1]図１は、第１の実施形態に係る付与装置の構成の一例を示す図である。

[図2]図２は、第１の実施形態に係る自己改ざん検知コードの一例を示す図である。

[図3]図３は、第１の実施形態に係る制御フローグラフの一例を示す図である。

[図4]図４は、第１の実施形態に係る自己改ざん検知コードの分割について説明するための図である。

[図5]図５は、第１の実施形態に係る自己改ざん検知コードの挿入について説明するための図である。

[図6]図６は、第１の実施形態に係る付与装置の全体の処理の流れを示すフローチャートである。

[図7]図７は、第１の実施形態に係る付与装置の抽出処理の流れを示すフローチャートである。

[図8]図８は、第１の実施形態に係る付与装置の分割処理の流れを示すフローチャートである。

[図9]図９は、第１の実施形態に係る付与装置の挿入処理の流れを示すフローチャートである。

[図10]図１０は、プログラムが実行されることにより付与装置が実現されるコンピュータの一例を示す図である。

発明を実施するための形態

[0010] 以下に、本願に係る付与装置、付与方法及び付与プログラムの実施形態を図面に基づいて詳細に説明する。なお、本発明は、以下に説明する実施形態により限定されるものではない。

[0011] ［本発明の背景］

まず、実施形態の詳細な説明を行う前に、本発明の背景について説明する。本発明の保護対象のプログラムであって、改ざんの標的となるプログラムとして、例えば、Ａｎｄｒｏｉｄ（登録商標）が搭載された携帯電話端末上で実行されるアプリケーションがある。Ａｎｄｒｏｉｄでは、ＯＳ（Operating System）がオープンソースであり仕様が明確である点や、多くの解析ツールが公開されている点等から、リバースエンジニアリングを用いたアプリケーションの改ざんが容易である。

[0012] 攻撃者は、アプリケーション配布サイト（以下、マーケットと呼ぶ。）から取得した正規のアプリケーションに改ざんを行うことで、マルウェアや海賊版アプリケーションを作成する。例えば、攻撃者はマーケットから取得した人気のアプリケーションに対して、遠隔操作を行う機能を持った悪性コードを挿入することで、見かけ上は、元の正規アプリケーションのように動作するが、その裏で攻撃者からの指示を受け悪性な活動を行うようなマルウェアを作成することが可能である。

[0013] このようなアプリケーションの改ざんへの対策として、アプリケーションにあらかじめ自己改ざん検知機能を付与しておく対策が有効である。自己改ざん検知とはプログラム実行中に当該プログラムが自身の完全性を検証することで、第三者による改ざんを検知する技術である。改ざんが検知された際には、プログラムの動作を停止したり、プログラムを利用するユーザに対して警告を表示したりすることで、改ざんされたプログラムがユーザの環境上で動作することを防止する。

[0014] 一方で、このような対策は、アプリケーションの開発者が自発的に行う必要があり、自己改ざん検知機能をアプリケーションに実装する際にはアプリケーション改ざんに関する知識や、実装能力が要求される。そのため、個々

の開発者のセキュリティに対する意識や実装スキルに依存して、そもそも対策が付与されていない場合や、対策の強度が弱く攻撃者によって自己改ざん検知機能自体が容易に無効化可能である場合がある。

[0015] このような背景を踏まえ、アプリケーションに対して堅牢な自己改ざん検知機能を全自動で付与する技術が研究されている。しかしながら、前述の通り、従来技術には、プログラムにおける自己改ざん検知機能の実装箇所が攻撃者によって容易に発見されやすい場合があるという問題があった。そこで、本発明は、このような問題を解決することを目的としている。

[0016] 具体的に、本発明では、自己改ざん検知機能の実装箇所を発見されにくくするため、ソースコードの行単位よりもさらに細かい粒度であるバイトコードの命令単位で自己改ざん検知コードを挿入する。

[0017] また、本発明では、自己改ざん検知コードの挿入を自動化することによって、個々の開発者のセキュリティに対する意識や実装スキルによらない自己改ざん検知機能の付与を可能としている。

[0018] ［第１の実施形態の構成］

まず、図１を用いて、本発明の第１の実施形態に係る付与装置の構成について説明する。図１は、第１の実施形態に係る付与装置の構成の一例を示す図である。図１に示すように、付与装置１０は、入力部１１、出力部１２、記憶部１３及び制御部１４を有する。

[0019] なお、第１の実施形態においては、自己改ざん検知機能が付与されるプログラムが、Ａｎｄｒｏｉｄのアプリケーションである場合の例について説明するが、本発明の保護対象のプログラムは、Ａｎｄｒｏｉｄのアプリケーションに限られず、バイトコードであるか、又はバイトコードに変換可能な任意のプログラムとすることができる。

[0020] 入力部１１は、ユーザからのデータの入力を受け付ける。入力部１１は、例えば、マウスやキーボード等の入力装置である。出力部１２は、画面の表示等により、データを出力する。出力部１２は、例えば、ディスプレイ等の表示装置である。また、入力部１１及び出力部１２は、データの入出力のた

めのインタフェースであってもよい。

[0021] また、入力部１１は、保護対象のアプリケーション２１の入力を受け付ける。このとき、保護対象のアプリケーション２１は、バイトコードである。なお、入力部１１は、アプリケーションのソースコードの入力を受け付けてもよい。この場合、付与装置１０は、ソースコードをバイトコードに変換する機能部をさらに有する。なお、保護対象のアプリケーション２１は、第１のプログラムの一例である。

[0022] また、出力部１２は、保護済みのアプリケーション２２を出力する。保護済みのアプリケーション２２は、保護対象のアプリケーション２１に、自己改ざん検知機能が付与されたアプリケーションである。

[0023] 記憶部１３は、ＨＤＤ（Hard Disk Drive）、ＳＳＤ（Solid State Drive）、光ディスク等の記憶装置である。なお、記憶部１３は、ＲＡＭ（Random Access Memory）、フラッシュメモリ、ＮＶＳＲＡＭ（Non Volatile Static Random Access Memory）等のデータを書き換え可能な半導体メモリであってもよい。記憶部１３は、付与装置１０で実行されるＯＳや各種プログラムを記憶する。さらに、記憶部１３は、プログラムの実行で用いられる各種情報を記憶する。また、記憶部１３は、自己改ざん検知コード１３１を記憶する。

[0024] 自己改ざん検知コード１３１は、あらかじめ自己改ざん検知を行う機能が実装されたプログラムのテンプレートである。記憶部１３は、複数種類の自己改ざん検知コード１３１を記憶していてもよい。

[0025] また、自己改ざん検知コード１３１に実装された自己改ざん検知を行う機能は、アプリケーションの完全性を検証できるものであればよい。例えば、自己改ざん検知を行う機能は、アプリケーションに付与された自己署名の値をハードコードされた値と比較することによって改ざんを検知するものであってもよいし、実行コードのハッシュ値を取得し、あらかじめ計算しておいたハッシュ値との比較によって改ざんを検知するものであってもよい。

[0026] ここで、図２を用いて、自己改ざん検知コード１３１について説明する。

図２は、第１の実施形態に係る自己改ざん検知コードの一例を示す図である。図２に示すように、自己改ざん検知コード１３１は、複数の行によって構成されたバイトコードである。なお、自己改ざん検知コード１３１は、第２のプログラムの一例である。

[0027] 制御部１４は、付与装置１０全体を制御する。制御部１４は、例えば、ＣＰＵ（Central Processing Unit）、ＭＰＵ（Micro Processing Unit）等の電子回路や、ＡＳＩＣ（Application Specific Integrated Circuit）、ＦＰＧＡ（Field Programmable Gate Array）等の集積回路である。また、制御部１４は、各種の処理手順を規定したプログラムや制御データを格納するための内部メモリを有し、内部メモリを用いて各処理を実行する。また、制御部１４は、各種のプログラムが動作することにより各種の処理部として機能する。例えば、制御部１４は、抽出部１４１、分割部１４２及び挿入部１４３を有する。

[0028] 抽出部１４１は、保護対象のアプリケーション２１のバイトコードを構成する命令列のブロックから、ランダムにブロックを抽出し、保護対象のアプリケーション２１が実行される際に、当該ランダムに抽出したブロックより前に必ず実行されるブロックを抽出する。

[0029] 具体的に、まず、抽出部１４１は、保護対象のアプリケーション２１のバイトコードに含まれる関数ごとに制御フローグラフを構築する。制御フローグラフは、例えば、保護対象のアプリケーション２１のバイトコードを入力として既存のバイトコード解析フレームワークを利用することで構築することができる。

[0030] 抽出部１４１は、保護対象のアプリケーション２１に含まれる関数ごとの制御フローグラフのそれぞれから、ランダムに起点となるノードを抽出し、関数の実行開始から起点ノードまで状態が遷移する際に必ず通過するノード、すなわち、関数の実行開始から当該起点となるノードに至る経路上のノードである支配ノードを抽出する。このとき抽出された起点となるノード及び支配ノードに対応したバイトコードの命令列のブロックが、自己改ざん検知

コードの挿入候補地点である。

[0031] ここで、図３を用いて、制御フローグラフについて説明する。図３は、第１の実施形態に係る制御フローグラフの一例を示す図である。制御フローグラフ２００は、保護対象のアプリケーション２１に含まれる所定の関数に対応した制御フローグラフである。図３に示すように、制御フローグラフ２００は、ノード２０１～２０７を有する。このとき、抽出部１４１は、起点ノードとして、ノード２０６を抽出する。なお、抽出部１４１は、起点ノードとして抽出するノードをランダムに決定する。

[0032] 次に、抽出部１４１は、関数の実行開始から起点ノードまで状態が遷移する際に必ず通過するノードであるノード２０１及び２０３を支配ノードとして抽出する。ここで、保護対象のアプリケーション２１が実行される際、ノード２０６に状態が遷移する場合であれば、ノード２０１、２０３及び２０６については、ノード２０１、２０３、２０６の順で実行されることが保証されている。この場合、自己改ざん検知コードの挿入候補地点は、ノード２０１、２０３及び２０６に対応したバイトコードの命令列のブロックに含まれる。また、抽出部１４１は、保護対象のアプリケーション２１に含まれる全ての関数について、起点ノード及び支配ノードの抽出を行う。

[0033] 分割部１４２は、保護対象のアプリケーション２１の改ざんを検知する自己改ざん検知コード１３１のバイトコードを構成する命令列を、ランダムに複数のブロックに分割する。このとき、分割部１４２は、バイトコードである自己改ざん検知コード１３１上の１命令を最小単位として分割を行うことができる。

[0034] また、分割部１４２は、抽出部１４１によって起点ノード及び支配ノードの抽出が行われた関数ごとに、自己改ざん検知コード１３１の分割を行う。また、分割部１４２は、分割後のブロック数や、各ブロックに含めるバイトコードの命令数を、分割対象の関数ごとにランダムに決定する。

[0035] ここで、図４を用いて、自己改ざん検知コード１３１の分割について説明する。図４は、第１の実施形態に係る自己改ざん検知コードの分割について

説明するための図である。

なお、図４は、図２に示す自己改ざん検知コード１３１の分割の例を表している。自己改ざん検知コード１３１は、１行目～１７行目のそれぞれが１つの命令列である。なお、自己改ざん検知コード１３１は、説明のため、実際の自己改ざん検知コードの一部を省略したものである。図４の例では、分割部１４２は、自己改ざん検知コード１３１をブロック１３２～１３８の７個のブロックに分割する。

[0036] このとき、分割部１４２は、自己改ざん検知コード１３１のバイトコードを構成する命令列のうち、あらかじめ設定された所定の条件を満たす連続する複数の命令列が同じブロックに含まれるように、分割を行ってもよい。例えば、分割部１４２は、連続して実行される必要がある命令列や条件分岐に依存して実行される命令列が同じブロックに含まれるように分割する。これにより、分割部１４２による分割に起因して、自己改ざん検知コード１３１によって実行される処理の挙動が想定されていたものと異なってしまうことを防止する。

[0037] 例えば、「invoke-virtual」で始まる命令列の次に「move-result」で始まる命令列があることを、前述の所定の条件として設定することができる。この場合、図４の例において、分割部１４２は、例えば１行目と２行目の命令列がいずれもブロック１３２に含まれるように分割を行う。なお、図４の１行目と２行目の命令列は、連続的に実行される必要がある命令列の一例である。

[0038] また、例えば「if-」と「return-void」の間に含まれることを前述の所定の条件として設定することができる。この場合、図４の例において、分割部１４２は、例えば１５行目から１７行目までの命令列がいずれもブロック１３８に含まれるように分割を行う。なお、図４の１行目と２行目の命令列は、条件分岐に依存して実行される命令列の一例である。

[0039] 挿入部１４３は、分割部１４２によって分割された複数のブロックを、抽出部１４１によって抽出されたブロック内のそれぞれ異なる位置に、自己改

ざん検知コード１３１における実行順序を維持して挿入する。具体的には、挿入部１４３は、制御フローグラフごとに、分割部１４２によって分割された複数のブロックを、起点となるノード及び支配ノードに対応するバイトコードのブロック内のそれぞれ異なる位置、すなわち自己改ざん検知コード１３１の挿入候補地点に挿入する。

[0040] 挿入部１４３は、分割部１４２によって分割された複数のブロックの挿入位置を、挿入候補地点の中からランダムに決定することができる。ただし、挿入部１４３は、挿入するブロックの実行順序が、自己改ざん検知コード１３１における実行順序と同一となるように挿入する。

[0041] ここで、図５を用いて、自己改ざん検知コード１３１の挿入について説明する。図５は、第１の実施形態に係る自己改ざん検知コードの挿入について説明するための図である。図５の例では、まず、挿入部１４３は、例えば、ノード２０１の２箇所を挿入位置として決定する。そして、挿入部１４３は、自己改ざん検知コード１３１における実行順序に従って、２箇所のうち先に実行される箇所にブロック１３２を挿入し、後に実行される箇所にブロック１３３を挿入する。同様に、挿入部１４３は、ブロック１３４～１３６をノード２０３の３箇所に挿入し、ブロック１３７及び１３８をノード２０６の２箇所に挿入する。

[0042] 挿入部１４３は、分割部１４２によって分割された複数のブロックを挿入する際に、例外処理の追加、又は変数保存領域の確保の少なくともいずれかを行う。例えば、挿入部１４３は、分割後のブロックに例外を発生させるコードが含まれていた場合、ブロックを挿入するとともに、例外処理を行うコードをさらに挿入する。また、分割後のブロックで一時的な変数が利用されている場合、挿入部１４３は、元々のバイトコード内で未使用の変数保存領域を利用するか、新たに追加した変数保存領域を利用するように当該ブロックを変更する。

[0043] なお、記憶部１３には、複数種類の自己改ざん検知コード１３１を用意しておいてもよい。この場合、挿入部１４３は、関数ごとに異なる自己改ざん

検知コード１３１を挿入することができる。

[0044] ［第１の実施形態の処理］

図６～９を用いて、付与装置１０の処理について説明する。まず、図６を用いて、付与装置１０の全体の処理の流れを説明する。図６は、第１の実施形態に係る付与装置の全体の処理の流れを示すフローチャートである。

[0045] 図６に示すように、まず、抽出部１４１は、自己改ざん検知コード１３１の挿入候補地点を抽出する（ステップＳ１０）。ここで、挿入部１４３は、保護対象のアプリケーション２１のバイトコードから、自己改ざん検知コード１３１の挿入候補地点が存在し、かつ、自己改ざん検知コード１３１が未挿入である関数を選択する（ステップＳ２０）。次に、分割部１４２は、自己改ざん検知コード１３１の分割を行う（ステップＳ３０）。そして、挿入部１４３は、選択した関数に、分割部１４２によって分割された自己改ざん検知コード１３１を挿入する（ステップＳ４０）。

[0046] 対象となる関数に、自己改ざん検知コード１３１の挿入が行われていないものが残っている場合（ステップＳ５０、Ｎｏ）、付与装置１０はステップＳ２０～Ｓ４０の処理をさらに実行する。一方、全ての対象となる関数に、自己改ざん検知コード１３１の挿入が行われた場合（ステップＳ５０、Ｙｅｓ）、付与装置１０は処理を終了する。その後、付与装置１０は、保護済みのアプリケーション２２のバイトコードを出力することができる。

[0047] 次に、図７を用いて、抽出部１４１による自己改ざん検知コード１３１の挿入候補地点を抽出する処理（ステップＳ１０）について詳細に説明する。図７は、第１の実施形態に係る付与装置の抽出処理の流れを示すフローチャートである。

[0048] 図７に示すように、抽出部１４１は、まず、保護対象のアプリケーション２１のバイトコード内の全ての関数を取得する（ステップＳ１０１）。次に、抽出部１４１は、自己改ざん検知コード１３１の挿入候補地点が未抽出である関数を選択する（ステップＳ１０２）。次に、抽出部１４１は、選択した関数の制御フローグラフを構築する（ステップＳ１０３）。

[0049] ここで、抽出部１４１は、構築した制御フローグラフ中の全ノードから、起点ノードをランダムに選択する（ステップＳ１０４）。そして、抽出部１４１は、起点ノード及び起点ノードの支配ノードを自己改ざん検知コード１３１の挿入候補地点として抽出する（ステップＳ１０５）。

[0050] ここで、自己改ざん検知コード１３１の挿入候補地点の抽出が行われていない関数が残っている場合（ステップＳ１０６、Ｎｏ）、抽出部１４１はステップＳ１０２～Ｓ１０５の処理をさらに実行する。一方、全関数から自己改ざん検知コード１３１の挿入候補地点の抽出が行われている場合、抽出部１４１は処理を終了する（ステップＳ１０６、Ｙｅｓ）。

[0051] 次に、図８を用いて、分割部１４２による自己改ざん検知コード１３１の分割処理（ステップＳ３０）について詳細に説明する。図８は、第１の実施形態に係る付与装置の分割処理の流れを示すフローチャートである。

[0052] 図８に示すように、分割部１４２は、まず、テンプレートとして記憶されている自己改ざん検知コード１３１を取得する（ステップＳ３０１）。次に、分割部１４２は、取得した自己改ざん検知コード１３１を複数のブロックにランダムに分割する（ステップＳ３０２）。

[0053] 次に、図９を用いて、分割部１４２による自己改ざん検知コード１３１の挿入処理（ステップＳ４０）について詳細に説明する。図９は、第１の実施形態に係る付与装置の挿入処理の流れを示すフローチャートである。

[0054] 図９に示すように、挿入部１４３は、まず抽出部１４１によって抽出された、自己改ざん検知コード１３１の挿入候補地点を取得する（ステップＳ４０１）。次に、挿入部１４３は、分割部１４２による分割が行われた自己改ざん検知コード１３１を取得する（ステップＳ４０２）。そして、挿入部１４３は、自己改ざん検知コード１３１の挿入候補地点に、分割後の自己改ざん検知コード１３１を挿入する（ステップＳ４０３）。

[0055] ［第１の実施形態の効果］

抽出部１４１は、第１のプログラムのバイトコードを構成する命令列のブロックから、ランダムにブロックを抽出し、第１のプログラムが実行される

際に、当該ランダムに抽出したブロックより前に必ず実行されるブロックを抽出する。また、分割部１４２は、第１のプログラムの改ざんを検知する第２のプログラムのバイトコードを構成する命令列を、ランダムに複数のブロックに分割する。また、挿入部１４３は、分割部１４２によって分割された複数のブロックを、抽出部１４１によって抽出されたブロック内のそれぞれ異なる位置に、第２のプログラムにおける実行順序を維持して挿入する。

[0056] 抽出部１４１によって抽出されたブロックは、第１のプログラム、すなわち保護対象のプログラムが実行される際に必ず順序通りに実行されるようになっているため、第２のプログラム、すなわち自己改ざん検知コードを挿入した場合に、各ブロックを漏れなく順序通りに実行させることができる。

[0057] また、保護対象のプログラムに含まれる関数ごとに、自己改ざん検知コードをランダムに分割したうえで挿入しているため、保護済みプログラムから、自己改ざん検知コード全体を機械的に発見することは困難である。このため、第１の実施形態によれば、プログラムにおける自己改ざん検知機能の実装箇所が、攻撃者によって容易に発見されないようにすることが可能になる。

[0058] また、抽出部１４１は、第１のプログラムに含まれる関数ごとの制御フローグラフのそれぞれから、ランダムに起点となるノードを抽出し、関数の実行開始から当該起点となるノードに至る経路上のノードである支配ノードを抽出することができる。このとき、挿入部１４３は、制御フローグラフごとに、分割部１４２によって分割された複数のブロックを、起点となるノード及び支配ノードに対応するバイトコードのブロック内のそれぞれ異なる位置に挿入する。制御フローグラフは、既存のバイトコード解析フレームワークを利用することで構築することができるため、第１の実施形態によれば、自己改ざん検知コードの挿入を自動化し、開発者の知識や実装能力によらない自己改ざん検知機能の付与を可能にすることができる。

[0059] また、分割部１４２は、第２のプログラムのバイトコードを構成する命令列のうち、あらかじめ設定された所定の条件を満たす連続する複数の命令列

が同じブロックに含まれるように、分割を行ってもよい。これにより、自己改ざん検知コードを分割することによって、保護済みのプログラムで実行される自己改ざん検知処理の挙動が、想定されていたものと異なってしまうことを防止することができる。

[0060] 挿入部１４３は、分割部１４２によって分割された複数のブロックを挿入する際に、例外処理の追加、又は変数保存領域の確保の少なくともいずれかを行う。これにより、挿入部１４３によるブロックの挿入により、保護済みのアプリケーション２２の実行に不整合が生じることを防止することができる。

[0061] ［システム構成等］

また、図示した各装置の各構成要素は機能概念的なものであり、必ずしも物理的に図示のように構成されていることを要しない。すなわち、各装置の分散・統合の具体的形態は図示のものに限られず、その全部又は一部を、各種の負荷や使用状況等に応じて、任意の単位で機能的又は物理的に分散・統合して構成することができる。さらに、各装置にて行われる各処理機能は、その全部又は任意の一部が、ＣＰＵ及び当該ＣＰＵにて解析実行されるプログラムにて実現され、あるいは、ワイヤードロジックによるハードウェアとして実現され得る。

[0062] また、本実施形態において説明した各処理のうち、自動的に行われるものとして説明した処理の全部又は一部を手動的に行うこともでき、あるいは、手動的に行われるものとして説明した処理の全部又は一部を公知の方法で自動的に行うこともできる。この他、上記文書中や図面中で示した処理手順、制御手順、具体的名称、各種のデータやパラメータを含む情報については、特記する場合を除いて任意に変更することができる。

[0063] ［プログラム］

一実施形態として、付与装置１０は、パッケージソフトウェアやオンラインソフトウェアとして上記の自己改ざん検知機能の付与を実行する付与プログラムを所望のコンピュータにインストールさせることによって実装できる

。例えば、上記の付与プログラムを情報処理装置に実行させることにより、情報処理装置を付与装置１０として機能させることができる。ここで言う情報処理装置には、デスクトップ型又はノート型のパーソナルコンピュータが含まれる。また、その他にも、情報処理装置にはスマートフォン、携帯電話機やＰＨＳ（Personal Handyphone System）等の移動体通信端末、さらには、ＰＤＡ（Personal Digital Assistant）等のスレート端末等がその範疇に含まれる。

[0064] また、付与装置１０は、ユーザが使用する端末装置をクライアントとし、当該クライアントに上記の自己改ざん検知機能の付与に関するサービスを提供する付与サーバ装置として実装することもできる。例えば、付与サーバ装置は、保護対象のプログラムのバイトコードを入力とし、保護済みのプログラムを出力とする付与サービスを提供するサーバ装置として実装される。この場合、付与サーバ装置は、Ｗｅｂサーバとして実装することとしてもよいし、アウトソーシングによって上記の自己改ざん検知機能の付与に関するサービスを提供するクラウドとして実装することとしてもかまわない。

[0065] 図１０は、プログラムが実行されることにより付与装置が実現されるコンピュータの一例を示す図である。コンピュータ１０００は、例えば、メモリ１０１０、ＣＰＵ１０２０を有する。また、コンピュータ１０００は、ハードディスクドライブインタフェース１０３０、ディスクドライブインタフェース１０４０、シリアルポートインタフェース１０５０、ビデオアダプタ１０６０、ネットワークインタフェース１０７０を有する。これらの各部は、バス１０８０によって接続される。

[0066] メモリ１０１０は、ＲＯＭ（Read Only Memory）１０１１及びＲＡＭ１０１２を含む。ＲＯＭ１０１１は、例えば、ＢＩＯＳ（Basic Input Output System）等のブートプログラムを記憶する。ハードディスクドライブインタフェース１０３０は、ハードディスクドライブ１０９０に接続される。ディスクドライブインタフェース１０４０は、ディスクドライブ１１００に接続される。例えば磁気ディスクや光ディスク等の着脱可能な記憶媒体が、デ

ィスクドライブ１１００に挿入される。シリアルポートインタフェース１０５０は、例えばマウス１１１０、キーボード１１２０に接続される。ビデオアダプタ１０６０は、例えばディスプレイ１１３０に接続される。

[0067] ハードディスクドライブ１０９０は、例えば、ＯＳ１０９１、アプリケーションプログラム１０９２、プログラムモジュール１０９３、プログラムデータ１０９４を記憶する。すなわち、付与装置１０の各処理を規定するプログラムは、コンピュータにより実行可能なコードが記述されたプログラムモジュール１０９３として実装される。プログラムモジュール１０９３は、例えばハードディスクドライブ１０９０に記憶される。例えば、付与装置１０における機能構成と同様の処理を実行するためのプログラムモジュール１０９３が、ハードディスクドライブ１０９０に記憶される。なお、ハードディスクドライブ１０９０は、ＳＳＤにより代替されてもよい。

[0068] また、上述した実施形態の処理で用いられる設定データは、プログラムデータ１０９４として、例えばメモリ１０１０やハードディスクドライブ１０９０に記憶される。そして、ＣＰＵ１０２０が、メモリ１０１０やハードディスクドライブ１０９０に記憶されたプログラムモジュール１０９３やプログラムデータ１０９４を必要に応じてＲＡＭ１０１２に読み出して実行する。

[0069] なお、プログラムモジュール１０９３やプログラムデータ１０９４は、ハードディスクドライブ１０９０に記憶される場合に限らず、例えば着脱可能な記憶媒体に記憶され、ディスクドライブ１１００等を介してＣＰＵ１０２０によって読み出されてもよい。あるいは、プログラムモジュール１０９３及びプログラムデータ１０９４は、ネットワーク（ＬＡＮ（Local Area Network）、ＷＡＮ（Wide Area Network）等）を介して接続された他のコンピュータに記憶されてもよい。そして、プログラムモジュール１０９３及びプログラムデータ１０９４は、他のコンピュータから、ネットワークインタフェース１０７０を介してＣＰＵ１０２０によって読み出されてもよい。

符号の説明

[0070] １０　付与装置

１１　入力部

１２　出力部

１３　記憶部

１４　制御部

２１　保護対象のアプリケーション

２２　保護済みのアプリケーション

１３１　自己改ざん検知コード

１３２、１３３、１３４、１３５、１３６、１３７、１３８　ブロック

１４１　抽出部

１４２　分割部

１４３　挿入部

２００　制御フローグラフ

２０１、２０２、２０３、２０４、２０５、２０６、２０７　ノード

請求の範囲

[請求項1] 保護対象の第１のプログラムが改ざんされたことを検知する機能を、前記第１のプログラムに付与する付与装置であって、

前記第１のプログラムのバイトコードを構成する命令列のブロックから、ランダムにブロックを抽出し、前記第１のプログラムが実行される際に、当該ランダムに抽出したブロックより前に必ず実行されるブロックを抽出する抽出部と、

前記第１のプログラムの改ざんを検知する第２のプログラムのバイトコードを構成する命令列を、ランダムに複数のブロックに分割する分割部と、

前記分割部によって分割された複数のブロックを、前記抽出部によって抽出されたブロック内のそれぞれ異なる位置に、前記第２のプログラムにおける実行順序を維持して挿入する挿入部と、

を有することを特徴とする付与装置。

[請求項2] 前記抽出部は、前記第１のプログラムに含まれる関数ごとの制御フローグラフのそれぞれから、ランダムに起点となるノードを抽出し、関数の実行開始から当該起点となるノードに至る経路上のノードである支配ノードを抽出し、

前記挿入部は、前記制御フローグラフごとに、前記分割部によって分割された複数のブロックを、前記起点となるノード及び前記支配ノードに対応するバイトコードのブロック内のそれぞれ異なる位置に挿入することを特徴とする請求項１に記載の付与装置。

[請求項3] 前記分割部は、前記第２のプログラムのバイトコードを構成する命令列のうち、あらかじめ設定された所定の条件を満たす連続する複数の命令列が同じブロックに含まれるように、分割を行うことを特徴とする請求項１又は２に記載の付与装置。

[請求項4] 前記挿入部は、前記分割部によって分割された複数のブロックを挿入する際に、例外処理の追加、又は変数保存領域の確保の少なくとも

いずれかを行うことを特徴とする請求項１から３のいずれか１項に記載の付与装置。

[請求項5] 保護対象の第１のプログラムが改ざんされたことを検知する機能を、前記第１のプログラムに付与する付与装置で実行される付与方法であって、

前記第１のプログラムのバイトコードを構成する命令列のブロックから、ランダムにブロックを抽出し、前記第１のプログラムが実行される際に、当該ランダムに抽出したブロックより前に必ず実行されるブロックを抽出する抽出工程と、

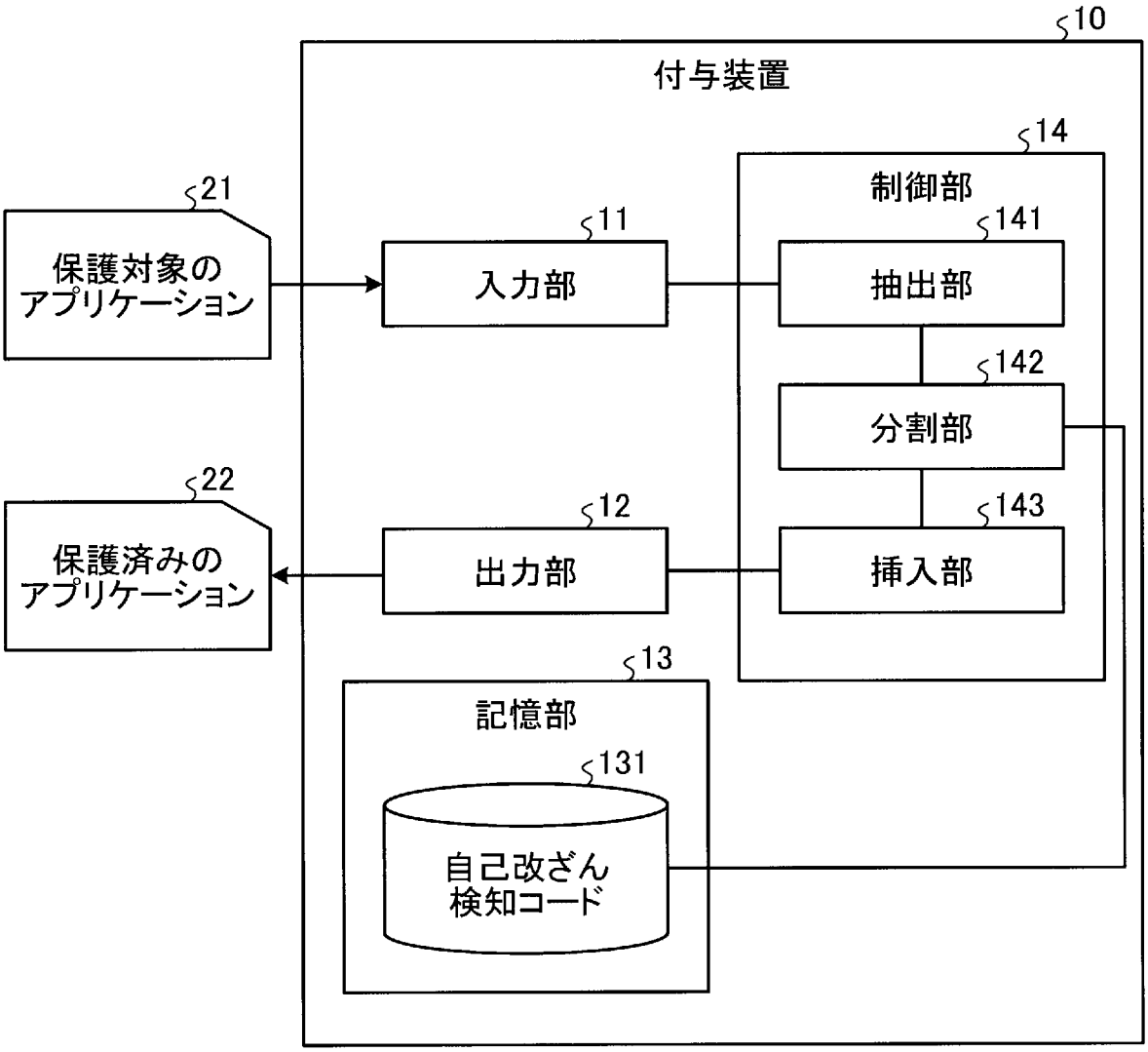
前記第１のプログラムの改ざんを検知する第２のプログラムのバイトコードを構成する命令列を、ランダムに複数のブロックに分割する分割工程と、

前記分割工程によって分割された複数のブロックを、前記抽出工程によって抽出されたブロック内のそれぞれ異なる位置に、前記第２のプログラムにおける実行順序を維持して挿入する挿入工程と、

を含んだことを特徴とする付与方法。

[請求項6] コンピュータを、請求項１から４のいずれか１項に記載の付与装置として機能させるための付与プログラム。

[図1]

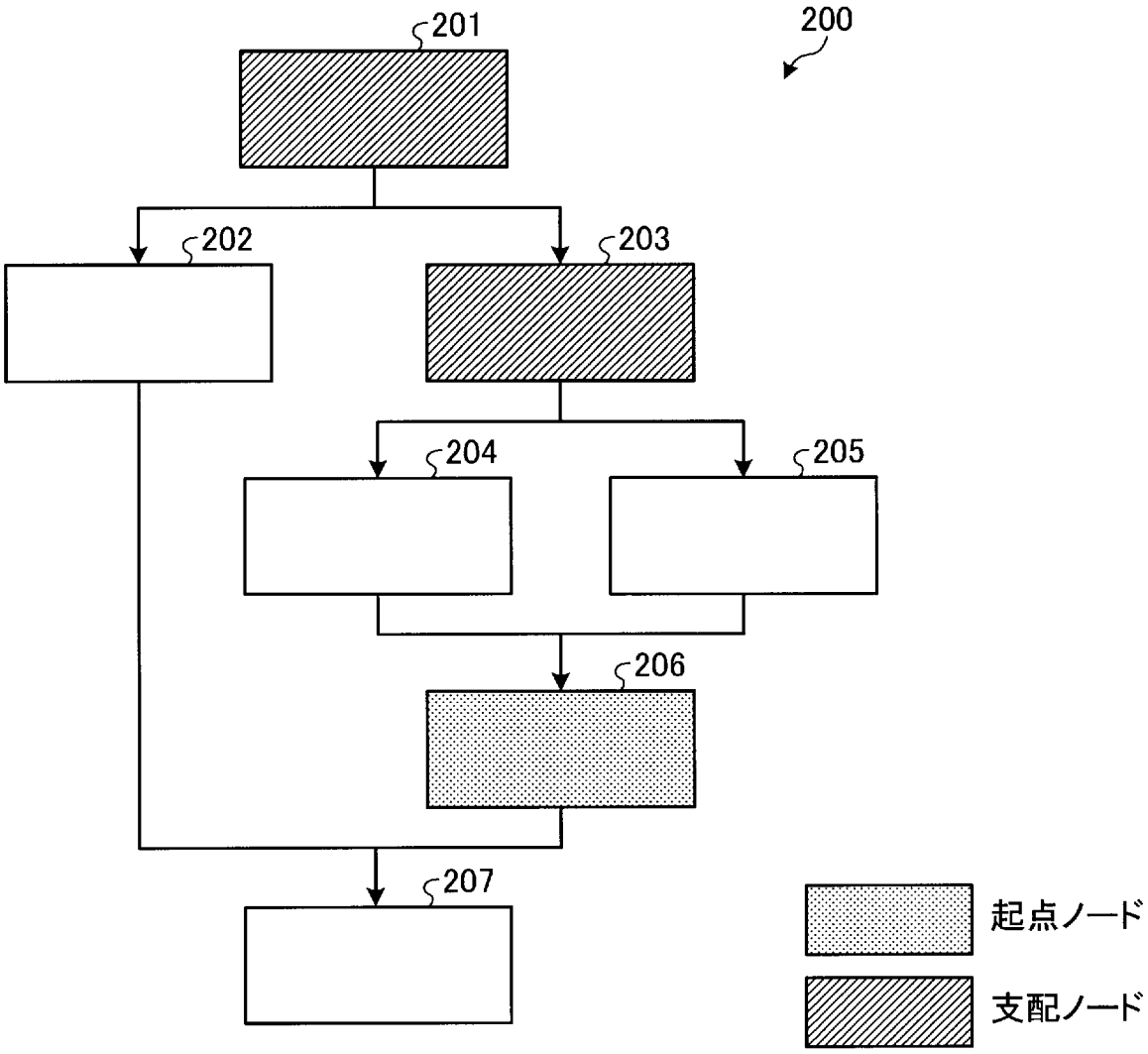

10
付与装置
14
制御部
21
保護対象の
アプリケーション
11
入力部
141
抽出部
142
分割部
22
保護済みの
アプリケーション
12
出力部
143
挿入部
13
記憶部
131
自己改ざん
検知コード

[図2]

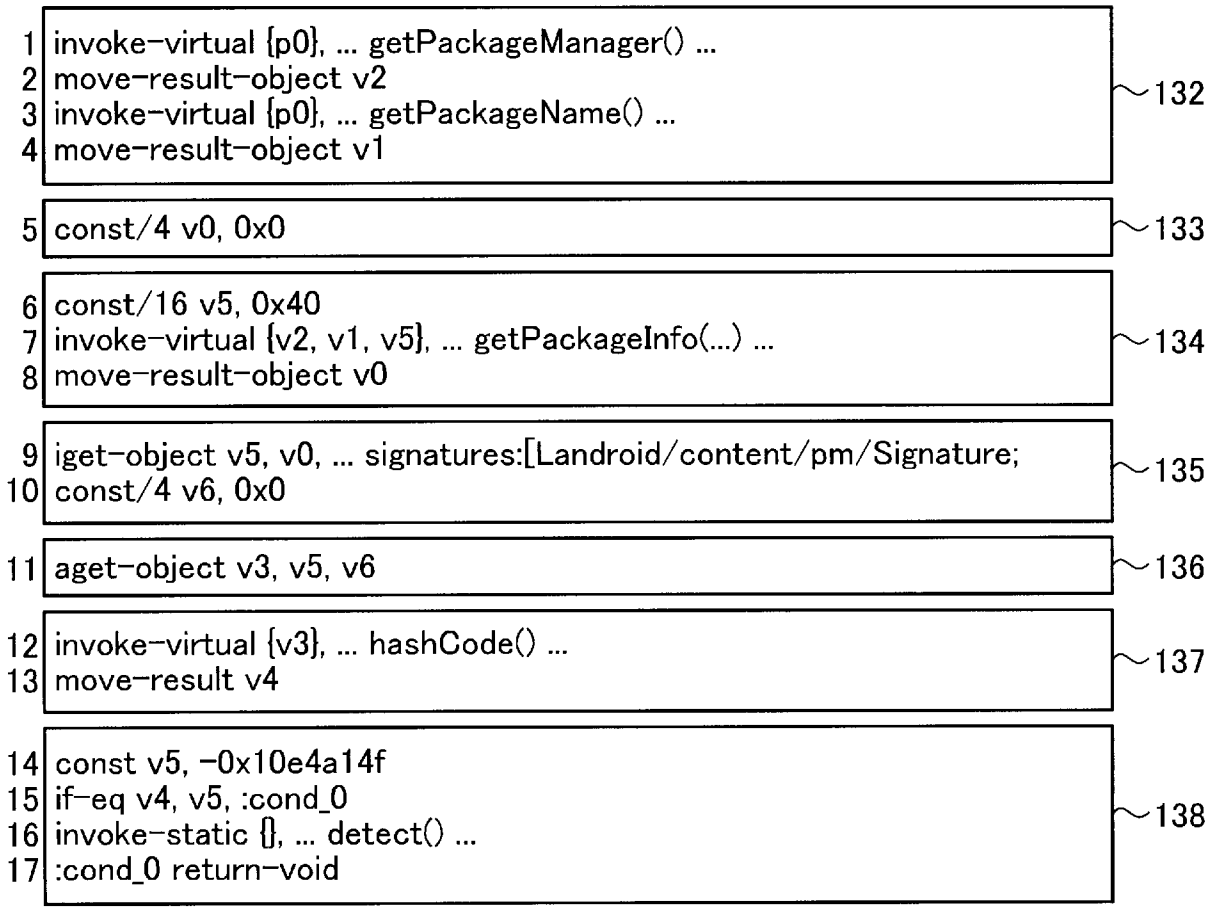
131

```
1  invoke-virtual {p0}, ... getPackageManager() ...
2  move-result-object v2
3  invoke-virtual {p0}, ... getPackageName() ...
4  move-result-object v1
5  const/4 v0, 0x0
6  const/16 v5, 0x40
7  invoke-virtual {v2, v1, v5}, ... getPackageInfo(...) ...
8  move-result-object v0
9  iget-object v5, v0, ... signatures:[Landroid/content/pm/Signature;
10 const/4 v6, 0x0
11 aget-object v3, v5, v6
12 invoke-virtual {v3}, ... hashCode() ...
13 move-result v4
14 const v5, -0x10e4a14f
15 if-eq v4, v5, :cond_0
16 invoke-static {}, ... detect() ...
17 :cond_0 return-void
```

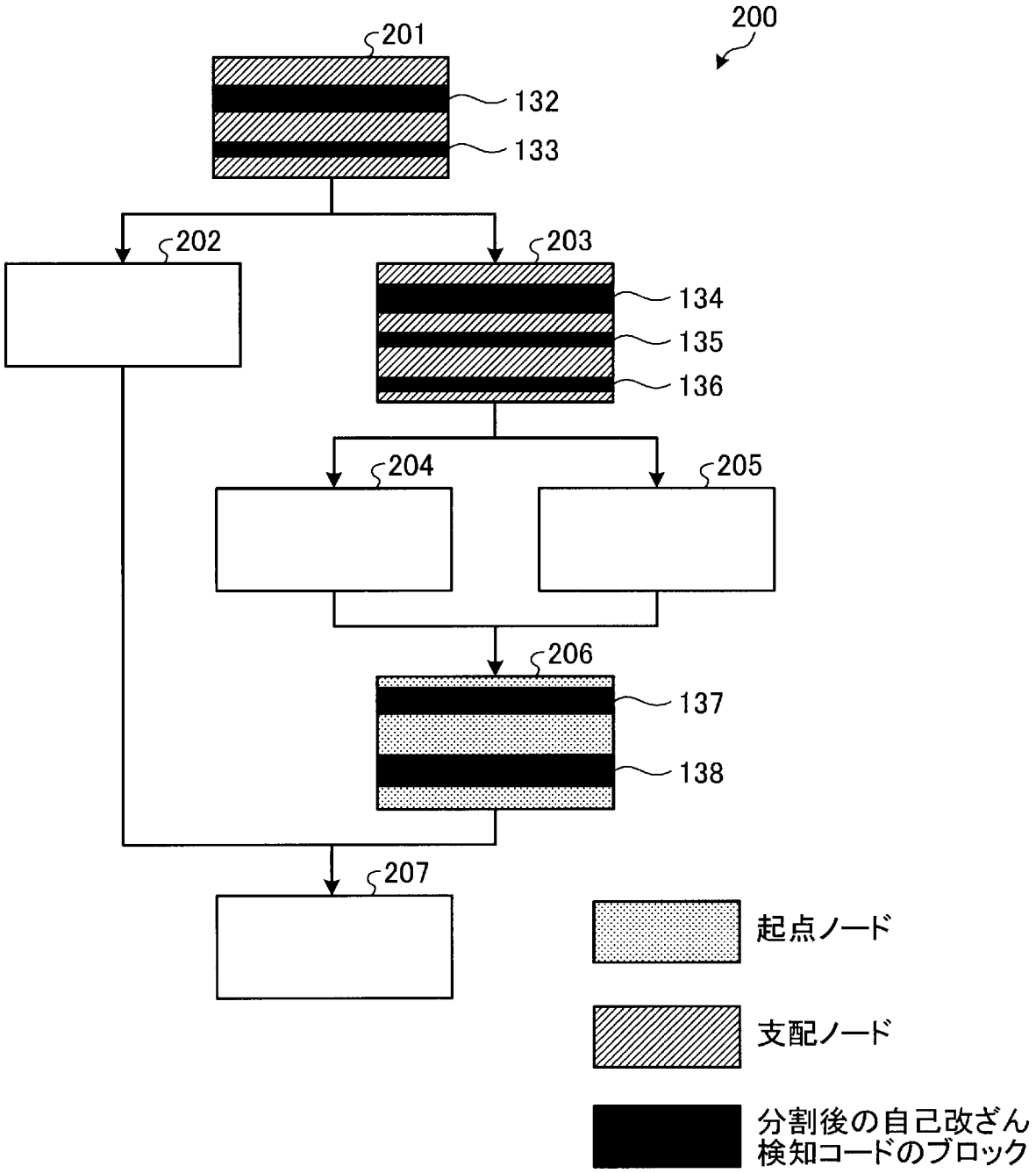
[図3]



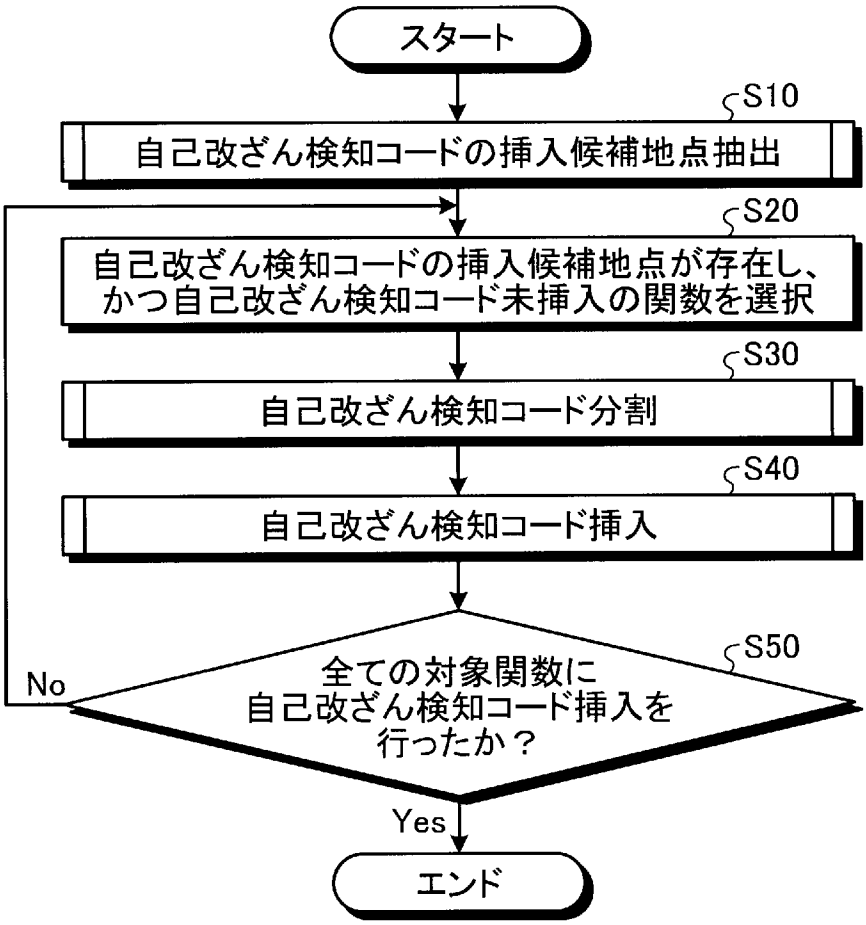
[図4]


1 invoke-virtual {p0}, ... getPackageManager() ...
2 move-result-object v2
3 invoke-virtual {p0}, ... getPackageName() ...
4 move-result-object v1
132
5 const/4 v0, 0x0
133
6 const/16 v5, 0x40
7 invoke-virtual {v2, v1, v5}, ... getPackageInfo(...) ...
8 move-result-object v0
134
9 iget-object v5, v0, ... signatures:[Landroid/content/pm/Signature;
10 const/4 v6, 0x0
135
11 aget-object v3, v5, v6
136
12 invoke-virtual {v3}, ... hashCode() ...
13 move-result v4
137
14 const v5, -0x10e4a14f
15 if-eq v4, v5, :cond_0
16 invoke-static {}, ... detect() ...
17 :cond_0 return-void
138

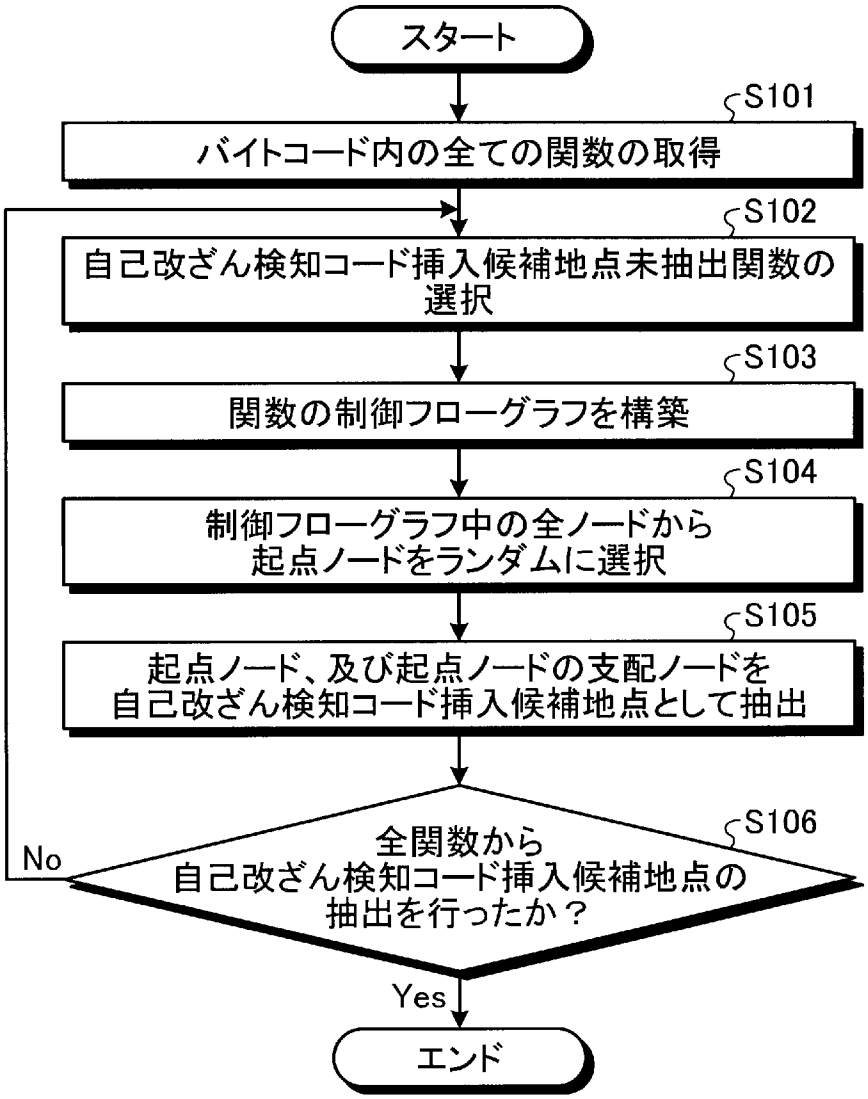
[図5]


200
201
132
133
202
203
134
135
136
204
205
206
137
138
207
起点ノード
支配ノード
分割後の自己改ざん検知コードのブロック

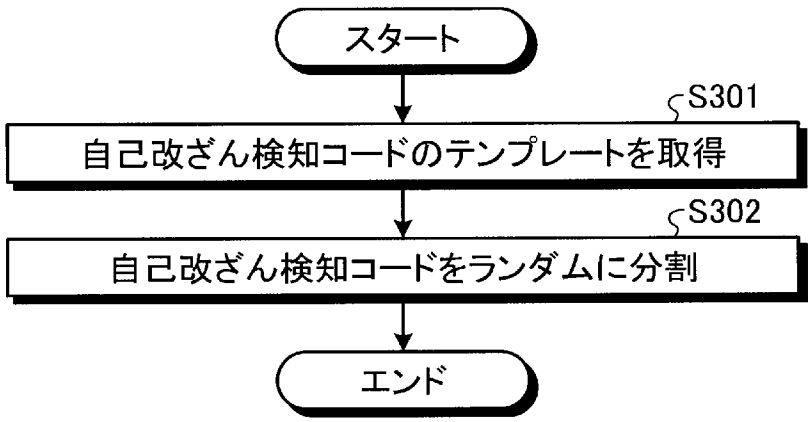
[図6]


スタート
S10
自己改ざん検知コードの挿入候補地点抽出
S20
自己改ざん検知コードの挿入候補地点が存在し、
かつ自己改ざん検知コード未挿入の関数を選択
S30
自己改ざん検知コード分割
S40
自己改ざん検知コード挿入
S50
全ての対象関数に
自己改ざん検知コード挿入を
行ったか？
No
Yes
エンド

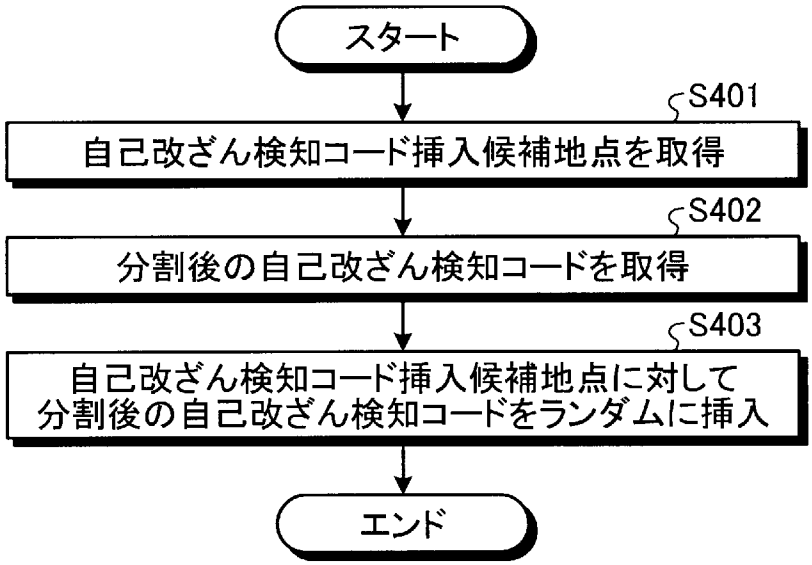
[図7]


スタート
S101
バイトコード内の全ての関数の取得
S102
自己改ざん検知コード挿入候補地点未抽出関数の選択
S103
関数の制御フローグラフを構築
S104
制御フローグラフ中の全ノードから起点ノードをランダムに選択
S105
起点ノード、及び起点ノードの支配ノードを自己改ざん検知コード挿入候補地点として抽出
S106
全関数から自己改ざん検知コード挿入候補地点の抽出を行ったか？
No
Yes
エンド

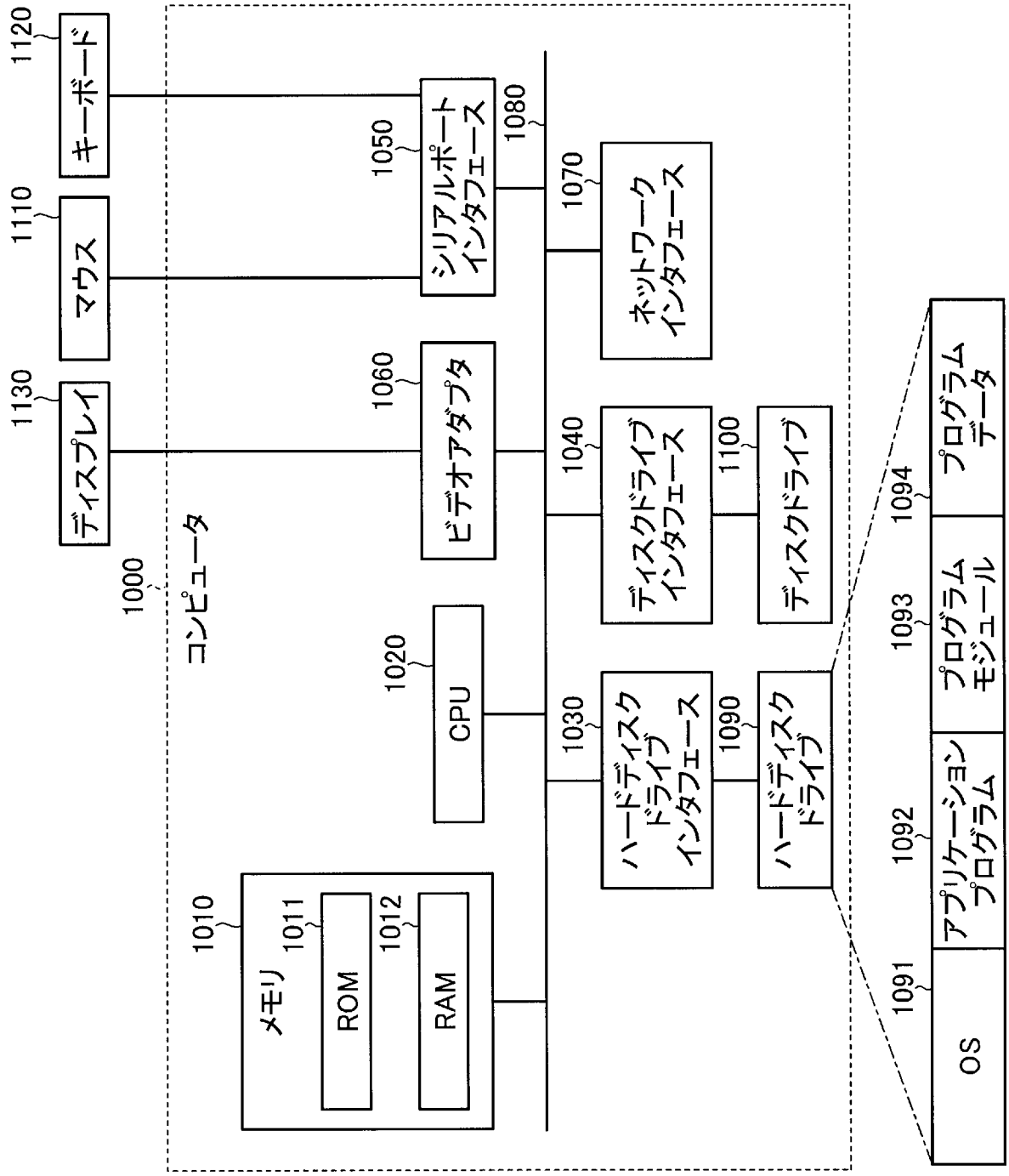
[図8]


スタート
S301
自己改ざん検知コードのテンプレートを取得
S302
自己改ざん検知コードをランダムに分割
エンド

[図9]


スタート
S401
自己改ざん検知コード挿入候補地点を取得
S402
分割後の自己改ざん検知コードを取得
S403
自己改ざん検知コード挿入候補地点に対して
分割後の自己改ざん検知コードをランダムに挿入
エンド

[図10]


1120
キーボード
1110
マウス
1130
ディスプレイ
1000
コンピュータ
1050
シリアルポート
インタフェース
1080
1070
ネットワーク
インタフェース
1060
ビデオアダプタ
1040
ディスクドライブ
インタフェース
1100
ディスクドライブ
1020
CPU
1030
ハードディスク
ドライブ
インタフェース
1090
ハードディスク
ドライブ
1010
メモリ
1011
ROM
1012
RAM
1091
OS
1092
アプリケーション
プログラム
1093
プログラム
モジュール
1094
プログラム
データ

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2017/034354

A. CLASSIFICATION OF SUBJECT MATTER
Int.Cl. G06F21/51(2013.01)i, G06F21/12(2013.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
Int.Cl. G06F21/51, G06F21/12

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Published examined utility model applications of Japan	1922-1996
Published unexamined utility model applications of Japan	1971-2017
Registered utility model specifications of Japan	1996-2017
Published registered utility model applications of Japan	1994-2017

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	金井文宏, Androidアプリケーションの自動リパッケージに対する耐性評価, 情報処理学会論文誌（ジャーナル）, (KANEI, Fumihiro, Evaluating Resistance of Android Applications to Automated Repackaging, Transactions of Information Processing Society of Japan) vol. 56, no. 12 [online], 15 December 2015, vol. 56, pp. 2275-2288	1-6
A	JP 2008-84275 A (FUJITSU LTD.) 10 April 2008, abstract (Family: none)	1-6
A	WO 2008/010508 A1 (MATSUSHITA ELECTRIC INDUSTRIAL CO., LTD.) 24 January 2008, paragraphs [0539]-[0546], fig. 16, 18 & US 2009/0177873 A1, paragraphs [0630]-[0637], fig. 16, 18 & CN 101490693 A	1-6

☐ Further documents are listed in the continuation of Box C.

☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search 14 December 2017 (14.12.2017)	Date of mailing of the international search report 26 December 2017 (26.12.2017)
Name and mailing address of the ISA/ Japan Patent Office 3-4-3, Kasumigaseki, Chiyoda-ku, Tokyo 100-8915, Japan	Authorized officer Telephone No.

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ２０１７／０３４３５４

Ａ．　発明の属する分野の分類（国際特許分類（ＩＰＣ））

Int.Cl.　G06F21/51(2013.01)i, G06F21/12(2013.01)i

Ｂ．　調査を行った分野

調査を行った最小限資料（国際特許分類（ＩＰＣ））

Int.Cl.　G06F21/51, G06F21/12

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報	１９２２－１９９６年
日本国公開実用新案公報	１９７１－２０１７年
日本国実用新案登録公報	１９９６－２０１７年
日本国登録実用新案公報	１９９４－２０１７年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

Ｃ．　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求項の番号
A	金井　文宏　FUMIHIRO　KANEI, Ａｎｄｒｏｉｄアプリケーションの自動リパッケージに対する耐性評価　Evaluating Resistance of Android Applications to Automated Repackaging, 情報処理学会論文誌（ジャーナル）　Ｖｏｌ．５６　Ｎｏ．１２　［ｏｎｌｉｎｅ］, 2015.12.15, 第56巻, pp.2275～2288	1-6
A	JP 2008-84275 A (富士通株式会社) 2008.04.10, 要約（ファミリーなし）	1-6

☑　Ｃ欄の続きにも文献が列挙されている。　　☐　パテントファミリーに関する別紙を参照。

＊　引用文献のカテゴリー
「Ａ」特に関連のある文献ではなく、一般的技術水準を示すもの
「Ｅ」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「Ｌ」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「Ｏ」口頭による開示、使用、展示等に言及する文献
「Ｐ」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
「Ｔ」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「Ｘ」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Ｙ」特に関連のある文献であって、当該文献と他の１以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「＆」同一パテントファミリー文献

国際調査を完了した日	国際調査報告の発送日
１４．１２．２０１７	２６．１２．２０１７

国際調査機関の名称及びあて先
日本国特許庁（ＩＳＡ／ＪＰ）
郵便番号１００－８９１５
東京都千代田区霞が関三丁目４番３号

特許庁審査官（権限のある職員）　５Ｓ　３９８３
岸野　徹
電話番号　０３－３５８１－１１０１　内線　３５４６

Ｃ（続き）．　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求項の番号
A	WO 2008/010508 A1 (松下電器産業株式会社) 2008.01.24, [0539]-[0546], 図16, 18 & US 2009/0177873 A1, [0630]-[0637], FIG.16, 18 & CN 101490693 A	1-6